Splitters and near-optimal derandomization

(Preliminary Version)

Moni Naor[*] Leonard J. Schulman[†] Aravind Srinivasan[‡]

Abstract

We present a fairly general method for finding deterministic constructions obeying what we call k-restrictions; this yields structures of size not much larger than the probabilistic bound. The structures constructed by our method include (n, k)-universal sets (a collection of binary vectors of length n such that for any subset of size k of the indices, all 2^k configurations appear) and families of perfect hash functions. The near-optimal constructions of these objects imply the very efficient derandomization of algorithms in learning, of fixed-subgraph finding algorithms, and of near optimal $\Sigma\Pi\Sigma$ threshold formulae. In addition, they derandomize the reduction showing the hardness of approximation of set cover. They also yield deterministic constructions for a local-coloring protocol, and for exhaustive testing of circuits.

1 Introduction

Research conducted over the last decades has demonstrated the significance of the Probabilistic Method and of probabilistic algorithms and procedures (see [6, 28] for recent reviews of these achievements). However, there are many reasons why one should not be satisfied with a probabilistic construction of an object or with a probabilistic algorithm. This is especially true in cases where there is no procedure for checking the correctness of the result. Also, probabilistic algorithms often behave less satisfactorily than deterministic ones under recursion, since this can require resource-expensive boosting of the success probability. Hence, a lot of effort has been devoted to finding ways of removing randomness from algorithms. Unfortunately, the resulting algorithm is often much less efficient than the original one. Exceptions to this are, *e.g.*, the results of [5, 10, 24], where there is no significant penalty in time (or number of processors, in the case of parallel algorithms).

The goal of this paper is to present a fairly general method for constructing some combinatorial objects which we call k-restriction collections. All k-restriction problems have a probabilistic construction obtained by picking a random collection of vectors. One can show a "union bound" for such a collection (see Section 3.1), and our method achieves deterministic constructions of sizes close to that of the union bound. These constructions in turn allow us to remove the randomness from a large variety of algorithms. A k-restriction problem is, roughly speaking, a collection of vectors of length n over an alphabet of size b such that for *any* k out the n indices, we will find some "nice" configurations; see Section 2.2 for the formal definition.

At the heart of our method are *splitters:* an (n, k, ℓ)-splitter H is a family of functions from $\{1, ..., n\}$ to $\{1, ..., \ell\}$ such that for all $S \subseteq \{1, ..., n\}$ with $|S| = k$, there is a $h \in H$ that splits S perfectly, i.e., into equal-sized parts $(h^{-1}(j)) \cap S$, $j = 1, 2, \ldots \ell$ (or as equal as possible, if ℓ does not divide k). Splitters themselves fall into the category of k-restriction problems for which our construction is applicable: the alphabet size is ℓ and each vector corresponds to a function h, where the ith entry of the vector is $h(i)$. The nice configurations for a specified k-set S are therefore those where each letter in the alphabet appears the same number of times, when restricted to S.

1.1 Method

We give here a brief overview of our method. Starting with a universe of size n, we first reduce our problem to one with a universe of size k^2 by finding a poly-time computable family H of (n, k, k^2)-splitters, i.e., a family H of maps from $\{1, \ldots, n\}$ to $\{1, \ldots, k^2\}$ such that for every k-sized subset S of $\{1, \ldots, n\}$, there is some function in H which is injective on S. A construction for the $[k^2]$-sized universe will then be "pulled back" to one on the $[n]$-universe, at a $poly(k) \cdot \log n$ cost in the size of the family.

Next we find a poly-time computable family of

[*] Incumbent of the Morris and Rose Goldman Career Development Chair, Dept. of Applied Mathematics and Computer Science, Weizmann Institute. Supported by an Alon Fellowship and by a grant from the Israel Science Foundation administered by the Israeli Academy of Sciences. E-mail: naor@wisdom.weizmann.ac.il.

[†] College of Computing, Georgia Inst. Technology, Atlanta GA 30332-0280, USA. Most of this work was done while the author was with the Dept. of Applied Mathematics and Computer Science, Weizmann Institute. E-mail: schulman@cc.gatech.edu.

[‡] Dept. of Information Systems & Computer Science, National University of Singapore, Singapore 0511, Republic of Singapore. Most of this work was done while the author was visiting the Dept. of Applied Mathematics and Computer Science, Weizmann Institute. Part of this work was done while visiting the Dept. of Computer Science, University of Warwick, England, supported in part by the ESPRIT Basic Research Action Programme of the EC under contract No. 7141 (project ALCOM II). Part was done while visiting the Max-Planck-Institut für Informatik, Saarbrücken, Germany. E-mail: aravind@iscs.nus.sg.

(k^2, k, l) splitters, typically for $\ell \approx \log k$. This gives us, for each k-set in $[k^2]$, a function which partitions the k-set into l evenly sized blocks. We then give an application-dependent construction within each block. This construction will be of the same size guaranteed by the existence proof, and its computation will not be poly-time in the size of the construction; yet it will be poly-time in the parameters of the original problem.

Finally the constructions for the different blocks are combined into a construction for the $[k^2]$-universe in an application-specific manner.

1.2 Problems

There are several problems (combinatorial structures) falling into our framework for which our method yields improved and near-optimal bounds. For most of these problems the improvement is most apparent when $k = \Theta(\log n)$. These problems are defined in Section 2.2 and their constructions and applications are described in detail in Section 5. These k-restriction problems include the following.

(i) **Splitters** are both a means (as mentioned above) and an end of our work. They are rather basic combinatorial objects. We use them for constructing near-optimal size depth-3 formulae for threshold functions, in Section 5.4; this constructivizes the probabilistic existential proof of [35]. An important special case of splitters is:

(ii) **Perfect hashing.** Let H be a family of functions mapping a domain of size n into a range of size k. H is an (n, k)*-family of perfect hash functions* if for all subsets S of size k from the domain there is an $h \in H$ that is 1-1 on S. Thus these are (n, k, k)-splitters. The union bound shows the existence of a family H such that $|H| = O(e^k \sqrt{k} \log n)$, while it is known that $|H| \geq \Omega(e^k \log n / \sqrt{k})$ [17, 21, 34, 32]. The previously best-known explicit construction (based on [40] and described in [7]), is of size $\Omega(11^k \log n)$ (this bound was not made explicit in these papers).

In section 4.4 we present a deterministic construction of size $e^k k^{O(\log k)} \log n$, for this problem. Perfect hash functions have many applications, *e.g.* in table look-up and communication complexity [18, 26, 33]. The area where our method is most relevant is in derandomizing the color-coding method of [7], where we obtain deterministic algorithms with performance close to the randomized ones.

(iii) (n, k)**-universal sets.** This problem is to construct a small set of vectors $T \subset \{0, 1\}^n$ such that for any index set $S \subseteq \{1, 2, \ldots, n\}$ with $|S| = k$, the projection of T on S contains all possible 2^k configurations. The problem originated in the testing of circuits, since it allows exhaustive testing of a circuit where each component relies on at most k inputs. The union bound shows the existence of (n, k)-universal sets of size $\lceil k 2^k \ln n \rceil$. A lower bound of $\Omega(2^k \ln n)$ is known [20]. Previously, the best explicit construction was of size $O(\min\{k 2^{3k} \log n, k^2 2^{2k} \log^2 n\})$ [3, 4, 30]. In section 5.2 we present a near-optimal deterministic construction of size $2^k k^{O(\log k)} \log n$ and discuss the applications of this construction for the fault-tolerance of the hypercube, learning algorithms, distributive coloring, and the hardness of the set-cover problem. Another class of structures related to the hardness of set-cover, *anti-universal sets*, is discussed in Section 5.3.

1.3 Explicit Constructions: Global vs. Local

There is a distinction to be made, when discussing explicit constructions, between what we call local and global constructions. For instance, if we were asked to construct an undirected graph $G = (V, E)$ on n vertices satisfying a certain property, we could give a deterministic construction which would list the edges in E in $poly(n)$ time; we would call this a *globally explicit* construction. However, a stronger type of construction is possible: given any node $v \in V$, outputting its neighborhood $N(v)$ in $poly(\log n, |N(v)|)$ time; this is what we would call a *locally explicit* construction, and is what is usually called for in the explicit construction of dispersers and constant-degree expanders, for instance. Clearly, local is stronger than global, in analogy to the distinction between log-space and polynomial time.

In our context of, say, universal test sets and perfect hash functions, globally explicit constructions would refer to listing out the corresponding families F in time polynomial in their size. Locally explicit constructions would just ask for $h_i(j)$ to be evaluated in time polynomial in the *representation* of n, i and j, i.e., $O(\log(n + |F|))$. (Here h_i stands for the ith function in the family F, and j is any index in $\{1, ..., n\}$.) When applying the construction for removing randomness, we require only globally explicit constructions and hence, in describing our results above, we referred to globally explicit ones. However, we also provide locally explicit constructions; these too come to within a $2^{o(k)}$ factor of optimal, but the $2^{o(k)}$ term is worse.

1.4 A brief review of derandomization

The random choices made by a probabilistic algorithm naturally define a probability space where each choice corresponds to a random variable. To remove randomness from an algorithm, we need a way of finding a successful assignment to these choices, deterministically.

One such approach, the *method of conditional probabilities* ([39, 36]), is to search the probability space for a good choice by shrinking the probability space at every iteration, by fixing an additional choice. A different approach for finding a good point is to show that if the random choices satisfy only some limited form of independence (in which case we may have a smaller space), the algorithm is successful. This approach is taken in [23, 2, 19, 30, 4].

These two approaches have been combined in two different ways in the past, in [9, 24, 29] and [5]. The framework suggested in this paper is a synthesis of many known techniques. Finding the "right" combination for achieving near-optimality seems to be the main contribution of this work.

2 Tools and definitions

Notation. Let $[n]$ denote the set $\{1, 2, \ldots, n\}$. For any k, $1 \le k \le n$, the family of k-sized subsets (or k-sets) of $[n]$ is denoted by $\binom{[n]}{k}$.

2.1 Limited independence and small-bias probability spaces

Let Ω be a probability space with n random variables $x_1, x_2, \ldots x_n$, each taking values in a finite set A. Recall that Ω is called *k-wise independent* if for any $\{i_1, i_2, \ldots, i_k\} \subseteq [n]$, the random variables $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$ are mutually independent. Often, as will be the case in this paper, it is also assumed that each x_j is uniformly distributed in A.

Fairly tight bounds are known on the size of k-wise independent spaces: there are explicit constructions of k-wise independent probability spaces of size $O(n^{\frac{a-1}{a}k})$ (assuming a is prime and $n+1$ is a power of a), where a $(= |A|)$ is the alphabet size. On the other hand, there is a lower bound of $\sum_{j=0}^{\lfloor k/2 \rfloor} \binom{n}{j}$, which for fixed k is $\Omega(n^{\lfloor k/2 \rfloor})$, for the size of such a sample space (see [6, 2, 15]). An important property of these constructions is that it is possible to list all members of the probability space in linear time.

When $A = \{0, 1\}$, we say that Ω is a *k-wise ϵ-biased* probability space if for any nonempty subset S of $[n]$ of size at most k we have $|Pr[\bigoplus_{i \in S} x_i = 0] - Pr[\bigoplus_{i \in S} x_i = 1]| \le \epsilon$. A key property of any k-wise ϵ-biased probability space is that $\forall s \le k$ $\forall \{i_1, i_2, \ldots, i_s\} \subseteq \binom{[n]}{s}$ $\forall b_1, b_2, \ldots, b_s \in \{0, 1\}$

$$|Pr(\bigwedge_{j=1}^{s} (x_{i_j} = b_j)) - 1/2^s| \le \epsilon.$$

Therefore k-wise ϵ-biased probability spaces are described as "almost k-wise independent" or "k-wise ϵ-dependent". The construction of small-bias spaces due to [30], as optimized in [3], yields a probability space of size $O(\frac{k \log n}{\epsilon^3})$; those of [4] yield probability spaces of size $O(\frac{k^2 \log^2 n}{\epsilon^2})$.

2.2 k-restriction problems

An instance of a k-restriction problem is specified by (i) positive integers b, k, n, m, and (ii) a list $\mathcal{C} = C_1, C_2, ..., C_m$ where each $C_i \subseteq [b]^k$, and with the collection $\mathcal{C}$ being invariant under permutations of $[k]$.

For a vector $v = (v_1, v_2, \ldots, v_n) \in [b]^n$ and a subset $S \in \binom{[n]}{k}$, we say that v satisfies the restriction C_j at S if $v(S) \in C_j$. (Here $v(S)$ is the vector $(v_{i_1}, ..., v_{i_s})$, for $S = \{i_1, ..., i_s\}$ and $i_1 < ... < i_s$.) We say that a collection of vectors $\mathcal{V} \subseteq [b]^n$ *satisfies the constraints* $\mathcal{C}$ if $\forall S \in \binom{[n]}{k}$ and $\forall j : 1 \le j \le m$ there exists $v \in \mathcal{V}$ such that $v(S) \in C_j$. An important parameter of a k-restriction problem is $c = \min_{1 \le j \le m} |C_j|$. We call c/b^k the *density* of the problem.

We now define the problems we deal with in this paper and explain why they fall into the category of k-restriction problems. See the introduction for the definition of splitters, (n, k)-universal sets, and perfect hash families.

(i) The (n, k, ℓ)**-Splitters problem.** (In case ℓ does not divide k, we require the first (k mod ℓ) parts to be of size $\lceil k/\ell \rceil$ and the remaining ones to be of size $\lfloor k/\ell \rfloor$.) To specify splitters as a restriction problem, let $b = \ell$ and let $\mathcal{C}$ consist of one set C_1 containing all vectors from $[b]^k$ such that each value in $[b]$ appears exactly k/ℓ times. Here, $c = \binom{k}{k/\ell, k/\ell, \ldots, k/\ell}$.

(ii) Perfect hashing. In this case, $b = k$ and $\mathcal{C}$ has exactly one element $C_1 \subseteq [k]^k$, which contains precisely all the permutations of $[k]$. Hence, $c = k!$ and the density is $k!/k^k$. Note that perfect hash families are splitters with $\ell = k$.

(iii) The (n, k)**-universal set problem.** In this problem, $b = 2$ and $\mathcal{C}$ consists of 2^k sets $C_x = \{x\}$ for all $x \in \{0, 1\}^k$. In this case, $c = 1$.

(iv) In order to prove improved non-approximability results for the set-cover problem, Feige has recently introduced the following sets [16], which we call (n, k, b) **anti-universal** sets as suggested by Oded Goldreich: a family of functions from $[n]$ to $[b]$ where for every k-set in $[n]$ and every vector $v \in [b]^k$, there is a function that *disagrees* with v in every coordinate. Formally, it is a collection of functions mapping $[n]$ to $[b]$ such that

$$\forall (i_1, i_2, \ldots, i_k) \in [n]^k \;\; \forall (a_1, a_2, \ldots, a_k) \in [b]^k$$

$$\exists h \in H \; \forall j \in [k] \; h(i_j) \ne a_j.$$

In this case $\mathcal{C}$ consists of b^k sets

$$C_x = \{y \in [b]^k : \; y_j \ne x_j \;\; \forall 1 \le j \le k\},$$

for all $x \in [b]^k$. Note that for $b = 2$, anti-universal sets are identical to (n, k)-universal sets. However for general b, we have $c = (b-1)^k$; the density is $((b-1)/b)^k$.

Note that $\mathcal{C}$ is *implicitly presented* in all the above four problems, and hence, we do not need an explicit list of the constraints for any of these problems. Thus by (globally) efficient algorithms for these four problems, we just mean algorithms taking time polynomial in n and in the output size.

3 Probabilistic and exhaustive bounds for k-restriction problems

3.1 The union bound for k-restriction problems

Suppose that for a k-restriction problem specified by $\mathcal{C} = C_1, C_2, ..., C_m \subseteq [b]^k$ such that $|C_j| \ge c$, we attempt a probabilistic construction. If a random vector $v \in [b]^n$ is chosen and we consider a specific $S \in \binom{[n]}{k}$

and some $C_j \in \mathcal{C}$, then the probability that v satisfies C_j at S is $\frac{|C_j|}{b^k} \geq \frac{c}{b^k}$. Therefore, if we choose t random vectors, we get via the union bound that the probability that the collection does not satisfy $\mathcal{C}$ is bounded above by

$$\sum_{S \in \binom{[n]}{k}} \sum_{j=1}^{m} Pr[\text{no } v \text{ satisfies } C_j \text{ at } S]$$

$$= \binom{n}{k} \sum_{i=1}^{m} \left(1 - \frac{|C_i|}{b^k}\right)^t \leq \binom{n}{k} m \left(1 - \frac{c}{b^k}\right)^t (1)$$

Restricting (1) to be less than 1 implies that

$$t \geq \lceil \frac{k \ln n + \ln m}{\ln(b^k/(b^k - c))} \rceil \qquad (2)$$

suffices; thus, for any given k-restriction problem of density c, there exists a solution of at most this size. We will refer to (2) as the *union bound.* For many k-restriction problems the union bound is very close to the best (smallest) possible construction: *e.g.*, for (n, k)-universal sets and perfect hash functions.

3.2 "Smart" exhaustive search

We now show how to get a construction that is of size equaling that given by the union bound, for any k-restriction problem. This phase of the construction is computationally expensive in its own right, i.e., not polynomial time in its parameters; however with the parameters we will be using it, it will take time polynomial in the parameters of the *main problem.* The reason for dubbing this "smart" search is that though it does brute-force search, the search domain is much smaller than that of the class of all functions mapping $[n]$ to $[b]$.

Typically, for a "main problem" with parameters N, K and B we apply this phase with $n = K^2$, $k = K/(\log K \log B)$ or $K/\log K$, and $b = B$. Since we are discussing general k-restriction problems here, we assume that the collection of constraints $\mathcal{C}$ is presented by a *membership oracle*: a procedure that, given any $v \in [b]^n$, $S \in \binom{[n]}{k}$ and $j \in [m]$, says whether or not $v(S) \in C_j$, within some time bound T. For the examples we are interested in, this oracle computation will be easy, usually taking just $O(k)$ time.

Let $H_{n,k,b}$ be a k-wise independent probability space with n random variables taking values in $[b]$, such as the one mentioned in section 2.1. Henceforth, we assume that $b \leq n$ for k-restriction problems for simplicity, since this is the case for all our applications; thus, $|H_{n,k,b}| \leq n^k$. First note that the union bound (2) is applicable even when the vectors are not chosen uniformly at random from $[b]^n$, but chosen uniformly at random from the much smaller space $H_{n,k,b}$ — this follows from the fact that (1) examines only k-sets of $[n]$.

Theorem 1 *For any k-restriction problem with $b \leq n$, there is a deterministic algorithm that outputs a collection obeying the k-restrictions, with the size of the collection equaling the union bound. The time taken to output the collection is*

$$O(\frac{b^k}{c} \cdot \binom{n}{k} \cdot m \cdot T \cdot |H_{n,k,b}|,$$

where T is the time complexity of the membership oracle. There is a parallel algorithm that outputs a collection at most a constant times larger than the union bound in time $poly(T + k \log n + \log m)$, using $O(\binom{n}{k} \cdot m \cdot |H_{n,k,b}|)$ EREW PRAM processors.

Proof. Consider a set-system in which the universe (ground set) is $H_{n,k,b}$. The sets are $T_{S,j}$, indexed by pairs (S, j) such that $S \in \binom{[n]}{k}$ and $1 \leq j \leq m$. $T_{S,j}$ consists of all $h \in H_{n,k,b}$ that satisfy C_j at S. We do not explicitly list out the sets $T_{S,j}$: note that any given h can be tested for membership in $T_{S,j}$ in time T, using the given membership oracle. Any subset of $H_{n,k,b}$ that *hits* (intersects) all subsets $T_{S,j}$ is a good collection (i.e. is a collection satisfying the k-restriction problem). This is the well-known *hitting set* or *transversal*, problem for hypergraphs.

We can find such a collection by a greedy algorithm via a simple observation, which follows fairly easily by inspecting (1) and by using the fact that (1) holds even if we pick vectors at random from $H_{n,k,b}$; the observation is that there must be an $h \in H_{n,k,b}$ such that h hits at least fraction c/b^k of the sets $T_{S,j}$. The obvious idea then is to find such an h using the membership oracle and add it to our current (partial) hitting set, "removing" the sets hit by h from the set-system, and repeating. Finding such an h takes time at most $O(\binom{n}{k} \cdot m \cdot T \cdot |H_{n,k,b}|)$; also, the number of sets in our set-system is effectively "shrunk" to at most $m\binom{n}{k}(1 - c/b^k)$ after picking h.

Therefore the results of a greedy algorithm would produce a construction of size $\lceil \frac{k \ln n + \ln m}{\ln b^k/(b^k - c)} \rceil$, same as that of (2). Also, the total time taken is at most

$$O(\binom{n}{k} \cdot m \cdot T \cdot |H_{n,k,b}| (\sum_{i=0}^{\infty} \left(1 - c/b^k\right)^i)), \ i.e.,$$

$$O(\frac{b^k}{c} \cdot \binom{n}{k} \cdot m \cdot T \cdot |H_{n,k,b}|).$$

This is the same as running the method of conditional probabilities on the *small space* $H_{n,k,b}$; if we were to run this method on the entire $[b]^n$ space, the time taken would be enormous.

Alternatively, any approximation algorithm for the hitting set problem is applicable here. (This is relevant in the parallel context, where one cannot use the greedy algorithm directly.) Berger, Rompel and Shor [10] have presented an efficient parallel algorithm for approximating the hitting set problem. This algorithm finds a hitting set that is within a constant factor of the output of the greedy algorithm. □

For two of our main applications, we explicitly state the time complexity of smart search; Theorem 2 follows directly from Theorem 1.

Theorem 2 *(i) An (n,k)-family of perfect hash functions $C(n,k)$ of cardinality $O(e^k\sqrt{k}\log n)$, can be constructed deterministically in time $O(k^{k+1}\binom{n}{k}n^k/k!)$. (ii) For any given n and $k \leq n$, an (n,k)-universal set of cardinality $O(k2^k\log n)$ can be constructed deterministically in time $O(\binom{n}{k}k2^{2k}n^{\lfloor k/2\rfloor})$.*

Note that this is *not* our final construction of perfect hash families and universal sets! The time complexities are too high in theorems 1 and 2, but the advantage offered by them is that the function families constructed are of "small" size (equaling the union bound). Theorems 1 and 2 will be invoked later on, with "small" values for n and k; this will keep the time taken low, while presenting function families of reasonable size.

4 Splitters

We now present a globally efficient construction for (n,k,ℓ)-splitters. Whenever $\ell < k$, we assume for notational convenience that $\ell \mid k$ (the argument for the general case is similar). We first present a probabilistic argument for (n,k,ℓ)-splitters when $\ell < k$, in Section 4.1. Sections 4.2 and 4.3 then provide some simple splitting families, which will be used to solve some basic sub-problems arising in our applications. Section 4.4 presents a near-optimal construction, building on the results of Sections 3.2, 4.2 and 4.3.

4.1 Probabilistic argument for splitters

Suppose $\ell < k$, $\ell \mid k$. If we pick s independent random functions from $[n]$ to $[\ell]$ where

$$s = \lceil \frac{\ell^k((k/\ell)!)^\ell k \ln n}{k!} \rceil,$$

then we see from (1) that we have an (n,k,ℓ)-splitter with positive probability. Using Robbins' formula–$e^{1/(12a+1)}\sqrt{2\pi a}(a/e)^a \leq a! \leq e^{1/(12a)}\sqrt{2\pi a}(a/e)^a$ [38] in the above definition of s and defining

$$\sigma(k,\ell) \doteq (2\pi k/\ell)^{\ell/2}e^{\ell^2/(12k)}$$

for notational convenience, we see that $s = \Theta(\sqrt{k}\sigma(k,\ell)\log n)$. Hence we get

Lemma 1 *If $\ell \mid k$, then for every $n \geq k$, there exists an (n,k,ℓ)-splitter of size $O(\sqrt{k}\sigma(k,\ell)\log n)$.*

4.2 Splitters for size-reduction

In our applications, it will be useful to have the parameter n "small" as a function of k; this would then help us invoke Theorems 1 and 2, while still keeping the time complexity low. The splitter of Lemma 2 shows how to do this "size-reduction", which essentially allows us to replace n by k^2. This makes our upper bounds for the applications have a linear dependence on $\log n$. Lemma 2 involves constructing a family of functions $A : [n] \to [k^2]$ such that for all $S \in \binom{[n]}{k}$, there is some function in A that is injective on S.

Lemma 2 *There is an explicit (n,k,k^2)-splitter $A(n,k)$ of size $O(k^6\log k\log n)$.*

Proof We follow the well-known trick of using an asymptotically good error correcting code with n codewords over the alphabet $[k^2]$, with a minimum relative distance of at least $1 - 2/k^2$ between any pair of codewords. Such explicit codes of length $L = O(k^6\log n\log k)$ exist [3]. There is a natural correspondence between the code and a family of splitters: the splitting family corresponds to the index set $[L]$. By summing the distances, we get that for any subset of k codewords there is an index where they all differ. This index corresponds to the good split. □

Alternatively, if we use the FKS functions (Corollary 2 and Lemma 2 [18]) then we get a family of size $\Theta(k^4\log^2 n/\log(k\log n))$.

4.3 Splitters for decomposition

Our applications will need small splitting families, and here we use a simple "intervals" family of splitters. This family is not very efficient but we will be using it in the range $n = k^2$ and $\ell = k^{o(1)}$ (principally $\ell = O(\log k)$) where its overhead is modest compared to the complexity of the overall construction.

Lemma 3 *For any $k \leq n$ and for all $\ell \leq n$, there is an explicit family $B(n,k,\ell)$ of (n,k,ℓ)-splitters of size $\binom{n}{\ell-1}$.*

Proof For every choice of $1 \leq i_1 < i_2 < \cdots < i_{\ell-1} \leq n$, define a function $h : [n] \mapsto [\ell]$ by $h(s) = j$ iff $i_{j-1} < s \leq i_j$, for all $s \in [n]$ (taking $i_0 \equiv 0$ and $i_\ell \equiv n$). □

4.4 Globally Explicit Splitter Construction

We now describe our best constructions of splitters. The form of the construction depends on the relative sizes of k and l.

First we note a lemma which follows from Theorem 1.

Lemma 4 *For $\ell \leq k$, an (n,k,ℓ)-splitter of cardinality $O(\sigma(k,\ell)\sqrt{k}\log n)$ can be constructed deterministically in time $O(\sqrt{k}\binom{n}{k}n^k\sigma(k,\ell))$.*

Theorem 3

(i) For $\ell = O(\sqrt{k})$, we can produce an (n,k,ℓ)-splitter of size

$$\begin{aligned} s &= O(k^{2\ell+O(1)}\log n/(\ell!)) \\ &= O((\sigma(k,\ell))^{6+o(1)}k^{O(1)}\log n) \end{aligned}$$

in poly(n,s) time, where the $o(1)$ term decreases monotonically and goes to 0, as $\ell/\sqrt{k}$ decreases and goes to 0.

(ii) For $\ell < k$ and $\ell = \omega(\sqrt{k})$, we can produce an (n,k,ℓ)-splitter of size

$$s = O(\sigma(k,\ell)^{1+o(1)}\log n)$$

in poly(n, s) time, where the $o(1)$ term goes to 0 as $\ell/\sqrt{k} \to \infty$.

(iii) **Perfect Hash Functions:** *For $k \leq \ell < k^2$, we can produce an (n, k, ℓ)-splitter of size $e^k k^{O(\log k)} \log n$ in time linear in the output size. (Also, for any $\ell < k$, an (n, k)-perfect family of hash functions of cardinality $e^k k^{O(1)} (\log n) \binom{k^2}{\ell} \left(\ln k \sqrt{2k/(\pi\ell)}\right)^{\ell}$ can be constructed deterministically in time poly$(n)\,(k/\ell)^{k/\ell+1}\,\binom{k^2}{k/\ell} k^{2k/\ell}/(k/\ell)!$.)*

(iv) For $l \geq k^2$, we can produce an (n, k, ℓ)-splitter of size $O(k^6 \log k \log n)$ in time poly(n, k).

Proof. 1. Let $A = A(n, k)$ and $B = B(k^2, k, \ell)$ be the respective function families (splitters) presented by Lemmas 2 and 3. For every $a \in A$ and $b \in B$, consider the function $g_{a,b}(x) = b(a(x))$; our function family F_1 will contain all the $|A||B|$ such functions $g_{a,b}$. To see that F_1 is an (n, k, ℓ)-splitter, consider any $S \in \binom{[n]}{k}$. There exists an $a \in A$ which is 1-1 on S, and a $b \in B$ that splits the image of S under a correctly; hence F_1 is all we need. Now $|F_1| = |A||B| = O(k^{2\ell+O(1)} \log n/(\ell!)) = O((\sigma(k, \ell))^{6+o(1)} k^{O(1)} \log n)$, for families of splitting problems with $\ell = O(\sqrt{k})$. This is not too far from the bound of Lemma 4; this method will, however, lead to huge splitters as ℓ grows further ($\ell = \omega(\sqrt{k})$), and hence we use a different approach in part (2).

2. For this part of the theorem, the $o(\cdot)$ and $\omega(\cdot)$ notation refers to k tending to infinity. We will need an integral parameter $1 < r < \ell$,

$$r = \Theta(k \log \ell/(\ell \log(2k/\ell))).$$

Note that $r = o(\ell)$ since $\ell = \omega(\sqrt{k})$; a similar easily verified fact that we will need is

$$(k/r)^{k/r} = (2k/\ell)^{O(\ell)} \text{ and } k^{O(r)} = O(\sigma(k, \ell)^{1+o(1)}). \tag{3}$$

We will assume for notational convenience that $r \mid \ell \mid k$. Define the function families $A_1 = A(n, k)$, $B = B(k^2, k, r)$ and $C = A(k^2, k/r)$ as presented by Lemmas 2 and 3, and the family D to be the $((k/r)^2, k/r, \ell/r)$-splitter presented by Lemma 4. Fix any $a \in A_1$, $b \in B$, any sequence of elements $c_1, c_2, \ldots, c_r$ of C, and any sequence of elements $d_1, d_2, \ldots, d_r$ of D. Then we define a function $g_{a,b,c_1,\ldots,c_r,d_1,\ldots,d_r} : [n] \mapsto [\ell]$ by

$$\begin{aligned} & g_{a,b,c_1,\ldots,c_r,d_1,\ldots,d_r}(x) \\ = \;& (b(a(x)) - 1)\ell/r + d_{b(a(x))}(c_{b(a(x))}(a(x))). \end{aligned}$$

Our function family F_2 is composed of precisely all such functions $g_{a,b,c_1,\ldots,c_r,d_1,\ldots,d_r}$. We first consider $|F_2|$ and the time to construct F_2, and then prove that F_2 is an (n, k, ℓ)-splitter. Note that

$$\begin{aligned} |F_2| &= |A_1||B|(|C||D|)^r \\ &= O(k^{O(1)}(\log n)\binom{k^2}{r} \\ &\qquad \cdot \left((k/r)^{O(1)}(\log r)\sigma(k/r, \ell/r)\right)^r) \\ &= O(k^{O(r)}\sigma(k, \ell) \log n) \\ &= O(\sigma(k, \ell)^{1+o(1)} \log n), \text{ by (3).} \end{aligned}$$

The total time to construct the families A_1, B, C and D is, by Lemmas 2, 3 and 4,

$$\begin{gathered} \mathit{poly}(|A_1|, |B|, |C|) + \\ O(\sqrt{k/r}\tbinom{(k/r)^2}{k/r})(k/r)^{2k/r}\sigma(k/r, \ell/r)\log(k/r)). \end{gathered}$$

Thus by (3), the time to construct F_2 is *poly*$(|F_2|, n)$.

To see that F_2 is an (n, k, ℓ)-splitter, take any $S \in \binom{[n]}{k}$. By the definition of A_1 and b,

$$\exists a \in A_1 \; \exists b \in B \; \forall i \in [r], \; |S_i| = k/r,$$

where $S_i = S \cap (b \circ a)^{-1}(i)$. Fix such an a and b. Again by the definition of A, there exist $c_1, c_2, \ldots, c_r \in C$ such that c_i is 1-1 on S_i; fix such a sequence $c_1, c_2, \ldots, c_r$. Finally by the definition of D, there is a sequence $d_1, d_2, \ldots, d_r \in D$ such that d_i splits S_i into ℓ/r pieces of size $(k/r)/(\ell/r) = k/\ell$ each. It is now not hard to verify that $g_{a,b,c_1,\ldots,c_r,d_1,\ldots,d_r}$ splits S.

3. Let $\ell = c \log k$ for some constant c (chosen suitably to minimize the running time, or the lower-order terms in the size of the perfect hash family). Let $A = A(n, k)$, $B = B(k^2, k, \ell)$ and $C = C(k^2, k/\ell)$ be the respective function families presented by Lemmas 2 and 3, and by Theorem 2(i).

The intuition is as follows; as mentioned above, we assume for now that $\ell | k$. A generic function f in our desired perfect hash family H is defined by a function $a \in A$, a function $b \in B$, and ℓ functions $c_1, \ldots, c_\ell \in C$; any such choice of the functions a, b, and c_i is allowed. Now f is defined by $f(x) = c_{b(a(x))}(a(x))$.

Observe that for any fixed $S \in \binom{[n]}{k}$, Lemmas 2 and 3 provide some pair a, b so that each c_i will be applied to k/ℓ points $a(x)$, $x \in S$. Now using Theorem 2(i), ranging over all choices for $c_i \in C$, there will be a function $f \in H$ that is 1-1 on S.

More formally, let j be defined (arbitrarily) to be 1 if $\ell | k$, and otherwise to be the integer given by $k = (\ell - j)\lfloor k/\ell \rfloor + j \lceil k/\ell \rceil$. Let $P_1, P_2, \ldots, P_\ell$ be the following function families: (i) P_i, for $1 \leq i \leq j$, is a $(k^2, \lceil k/\ell \rceil)$-perfect family of hash functions, as presented by Theorem 2(i), and (ii) P_i, for $j+1 \leq i \leq \ell$, is a $(k^2, \lfloor k/\ell \rfloor)$-perfect family of hash functions, as presented by Theorem 2(i).

Now for every $f_1 \in A$, $f_2 \in B$ and $g_i \in P_i$, define, $\forall s \in [n]$, $h_{(f_1, f_2, g_1, g_2, \ldots, g_\ell)}(s)$ to be

$$\lceil k/\ell \rceil (f_2(f_1(s)) - 1) + P_{f_2(f_1(s))}(f_1(s))$$

if $f_2(f_1(s)) \leq j+1$, and

$$\lceil k/\ell \rceil j + \lfloor k/\ell \rfloor (f_2(f_1(s)) - j - 1) + P_{f_2(f_1(s))}(f_1(s))$$

otherwise.

As sketched in the intuitive description above, we define our desired family H of functions to be the collection of all such functions $h_{(f_1, f_2, g_1, g_2, \ldots, g_\ell)}$. Now, $|A| = O(k^6 \log k \log n)$, $|B| = \binom{k^2}{\ell - 1} = k^{O(\log k)}$, and $\prod_{i=1}^{\ell} |P_i| = e^k k^{O(\ell)} = e^k k^{O(\log k)}$. Hence,

$$|H| = |A||B| \prod_{i=1}^{\ell} |P_i| = e^k k^{O(\log k)} \log n. \qquad (4)$$

It is not hard to verify that each $h \in H$ maps $[n]$ to $[k]$. We now show that H is indeed (n,k)-perfect. Let S be an arbitrary element of $\binom{[n]}{k}$. By the properties of A and B, we know the existence of $f_1 \in A$ and $f_2 \in B$ such that

$$|(f_2 \circ f_1)^{-1}(i) \cap S| = \lceil k/\ell \rceil \text{ for all } i \in [j], \text{ and}$$

$$|(f_2 \circ f_1)^{-1}(i) \cap S| = \lfloor k/\ell \rfloor \text{ for all } i \in \{j+1, j+2, \ldots, \ell\}.$$

For each $i \in [\ell]$, we are also assured of the existence of a $g_i \in P_i$ which is 1-1 on $(f_2 \circ f_1)^{-1}(i) \cap S$. Thus, $h_{(f_1, f_2, g_1, g_2, \ldots, g_\ell)}$ is 1-1 on S, which is what we set about to show.

By Theorem 2(i), each P_i can be constructed in $k^{O(k/\ell)} = 2^{O(k)}$ time. (This is the reason for our choice of ℓ to be $\Omega(\log k)$; since larger values of ℓ will make the splitter of Lemma 3 inefficient, we settled for $\ell = \Theta(\log k)$.) The other operations take at most $poly(2^k, n)$ time.

In some situations, it would be good to have the dependence on ℓ explicit, rather than fixing $\ell = \Theta(\log k)$. By keeping ℓ unspecified above, we obtained the parameterized version of the statement (using Robbins' formula).

4. From Lemma 2. □

A locally explicit construction for (n,k)-universal sets is presented by Theorem 7; similar locally explicit constructions of splitters and perfect hash families will be presented in the final version of this work.

5 Applications

In this section, we show how to apply our method to achieve good linear-time constructions of the k-restriction problems defined in Section 2.2 (but for splitters and perfect hash families, which we have already discussed). We also discuss the implication of these constructions for efficient derandomization.

5.1 Applications of Perfect Hash functions

Perfect hash functions have been used in [7] to derandomize many algorithms for finding subgraphs, such as paths and cycles of length k. We provide some improvement to these methods.

Theorem 4 ([7]) *A simple (directed) path of length $k-1$ in a (directed) graph $G = (V, E)$ that contains such a path can be found in $(2e)^k \cdot poly(k) \cdot |V|$ expected time in the undirected case and in $(2e)^k \cdot poly(k) \cdot |E|$ expected time in the directed case. A simple (directed) cycle of size k in a (directed) graph $G = (V, E)$ that contains such a path can be found in either $(2e)^k \cdot poly(k) \cdot |V||E|$ or $(2e)^k \cdot poly(k) \cdot |V|^\omega$ expected time, where $\omega < 2.376$ is the exponent of matrix multiplication.*

As pointed out in [7], explicit $(|V|, k)$-perfect hash functions can be used to derandomize their algorithm. Thus, we get near-optimal derandomization using our improved constructions of perfect hash functions. If $k = o(\log |V|)$, Theorem 3(3) can be used as stated; if $k = \Theta(\log |V|)$, we have to appeal to its parameterized version to get the best constant in the exponent. Analogous results hold for finding an isomorphic copy of a fixed directed forest with k vertices in a graph, as shown in [7].

Theorem 5 *A simple (directed) path of length $k-1$ in a (directed) graph $G = (V, E)$ that contains such a path can be found in $(2e)^k \cdot k^{O(\log k)} \cdot |V| \log |V|$ time in the undirected case and in $(2e)^k \cdot k^{O(\log k)} \cdot |E| \log |V|$ time in the directed case. A simple (directed) cycle of size k in a (directed) graph $G = (V, E)$ that contains such a path can be found in either $(2e)^k \cdot k^{O(\log k)} \cdot |V||E| \log |V|$ or $(2e)^k \cdot k^{O(\log k)} \cdot |V|^\omega \log |V|$ time.*

Note that if we are given an arbitrary graph (with no assurance about the existence of the desired object) then the [7] algorithms may err and to reduce the probability of error significantly, to, say, at most $2^{-|V|}$, one may need to run the algorithm many times $(\Omega(|V|))$, incurring a cost larger than our deterministic algorithm!

5.2 (n,k)-universal sets

The idea for (n,k)-universal sets is similar to that behind Theorem 3, with the only modification being that we now need the universal sets guaranteed by Theorem 2(ii). We thus get

Theorem 6 *We have a deterministic construction for (n,k)-universal sets of size $2^k k^{O(\log k)} \log n$. The collection may be listed in linear time.*

We also present locally explicit constructions of (n,k)-universal sets; the proof of Theorem 7 is sketched below. Analogous results hold for perfect hash families and splitters; we shall present the details in the full version.

Theorem 7 *There is a locally explicit construction of (n,k)-universal sets of cardinality $2^{k+o(k)} \log n$.*

Proof Sketch: The reason why the constructions of Theorems 3 and 6 and are not locally explicit lies in their usage of the smart search. This brings an exponential dependence on k to the time complexity. In order to get a local construction we must reduce the problem to parameters that allow "smart"-search in the allowed time (polynomial in $\log n$ and k). Note that this construction is not local in the sense of presenting an explicit structure, since some search is involved; however, it is local in the sense of Section 1.3.

To achieve the above-stated locally explicit construction, we need the following lemma. See, e.g., Lemma 2.3 in [11] for a proof of Lemma 5.

Lemma 5 *Let r, k and n be positive integers such that $r < k \leq n$ and such that $k/r \geq \log k$. Then for any fixed $c > 0$, there exist constants $a, b > 0$ such that the following holds. If a random function $[n] \rightarrow [r]$ is picked so that the random variables $\{h(1), h(2), \ldots, h(n)\}$ are uniform in $[r]$ and $(a \log k)$-wise independent, then $\forall S \in \binom{[n]}{k} \quad \forall i \in [r]$:*

$$Pr\left(\mid |S \cap h^{-1}(i)| - k/r \mid \geq b\sqrt{(k/r)\log r} \right) \leq r^{-c}.$$

We can use the $a \log k$-wise independent sample space H as discussed in Section 2.1. Briefly, our procedure involves a recursion of depth two:

(i) "Reduce" n to k^2, using Lemma 2; hence we may assume that $n = k^2$ henceforth.
(ii) Take $r_1 = \frac{k}{\log^d k}$ for some large enough (but fixed) d. Apply Lemma 5 using the above-seen small probability space H of size $k^{O(\log k)}$, to reduce the problem to r_1 subproblems, each of which is to construct a $(k^2, k_1 = k/r_1 + b\sqrt{(k/r_1)\log r_1})$-universal set.
(iii) Solve one of these subproblems recursively (see below) and take the r_1-fold directed product of the constructed universal test set.
(iv) To solve the subproblem, take $r_2 = 2\log^{d-1} k$; apply the hash functions as in Step (ii) and in order to solve the resulting problem apply Theorem 6 above. Note that the resulting k_2 is $O(\log k)$ and hence that the search needed takes time only polynomial in k.

A different approach toward a local construction which may yield improved results is to use a refined version of Lemma 5 that will allow a very large fraction of the elements of S to be mapped into bins that do not contain much more than the expected number (at most a standard deviation above the mean). The good bins are handled recursively, as above. Those elements that were not mapped into empty bins are then taken care of via a separate construction of a (n, k')-universal set, where $k' \ll k$. The two collections are merged by adding (bit-wise Xor) all possible pairs of vectors from the two constructions.

We shall present the detailed proofs in the full version, both because of lack of space and also since our applications require just globally efficient constructions. □

The problem of constructing (n, k)-universal sets has received much attention in the fault-diagnostic literature. See [41] for a bibliography on the problem. We can also phrase the problem as follows: for any $S \subseteq [n]$, let $S^0 \doteq S$ and $S^1 \doteq [n] - S$. Then, our problem is equivalent to finding a collection $\mathcal{C}$ of n subsets of a ground set X that is as small as possible, such that for all $S_1, S_2, \ldots, S_k \in \mathcal{C}$ and for all $b_1, b_2, \ldots, b_k \in \{0, 1\}$, $\bigcap_{i=1}^{k} S_i^{b_i} \neq \phi$. This was called k-independent by Kleitman and Spencer [20]. An equivalent formulation of this problem is related to a fault-tolerance property of the hypercube [13].

Discussion of Krichevsky's result. Krichevsky [22] claims a deterministic construction of size $(1 + o(1))k2^k \ln n$. However, this appears to be wrong and his construction is of size at least $2^{k+\Omega(k/\log\log\log k)} \log n$, as shown in the full version [31].

Application to learning algorithms: Blum and Rudich [9] have applied the construction of (n, k)-universal sets to obtain a deterministic algorithm to learn k-term DNFs. Bshouty [14] provided a learning algorithm for k-CNF running in time polynomial in the DNF size. The time in both algorithms is proportional to the size of the (n, k)-universal sets. Our construction for (n, k)-universal sets yields improved algorithms; the improvement is most significant when $k = \Theta(\log n)$.

Application to the hardness of approximating the set cover problem: Under the assumption that $NP \not\subseteq DTIME[n^{polylog(n)}]$, Lund & Yannakakis showed that for any fixed $c < 1/4$, one cannot approximate set cover to within a factor of $c \log_2 N$ in polynomial time, where N is the size of the ground set of the set cover instance [25]. What is striking about this result is the existence of well-known polynomial-time algorithms achieving a performance ratio of $\ln N + O(1)$. One of the main ingredients of this result was an explicit construction of (n, k)-universal sets. Since the randomized construction of (n, k)-universal sets is better than the previously known deterministic constructions, they also used the randomized construction to show that if $NP \not\subseteq ZTIME[n^{polylog(n)}]$, then for any fixed $c < 1/2$, one cannot approximate set cover to within a factor of $c \log_2 N$ in random polynomial time ($ZTIME$ denotes zero-error probabilistic polynomial time, corresponding to Las Vegas algorithms). Bellare et al. [8] improved the time bounds of Lund & Yannakakis above while losing a bit in the constants. However, taken together with the recent result of Raz [37] we get that that the time complexity can be reduced to $n^{O(\log\log n)}$. The constants of $\log n$ are still $1/4$ and $1/2$ for the deterministic and randomized cases respectively. Our explicit construction improves these non-approximability results by making the deterministic and randomized case have the same performance; we thus conclude that

Theorem 8 *If $NP \not\subseteq DTIME[n^{O(\log\log(n))}]$, then for any fixed $c < 1/2$, one cannot approximate set cover to within a factor of $c \log_2 N$ in deterministic polynomial time.*

Application to distributed coloring: Szegedy and Vishwanathan [42] considered the local coloring problem (see their paper for definition). They showed, via a non-constructive argument, the existence of recoloring protocols that starting with a graph of max degree d colored with c colors go (in a single step) to $O(d2^d \log\log c)$ colors. Mayer et al [27] showed that the key property used by [42] is (n, d)-universal sets. Using the improved construction of this paper we get a constructive version of [42], to within a factor $d^{O(\log d)}$ of the color bound of $O(d2^d \log\log c)$.

5.3 Anti-Universal Sets

In this section we show the construction of anti-universal sets (see definition in section 2.2). The union bound guarantees such a family of size $s = \lceil (b/(b-1))^k k \ln(bn) \rceil$ and as before, we have a simple deterministic construction of such a family, running in time $(nb)^{O(k)} s$. To make this more efficient, we first reduce n to k^2, as before. Next if $b = O(1)$, then we can split to $\ell = \log k \log b$ parts, and get a final output of size $sk^{O(\log k \log b)}$. On the other hand, if b increases with the parameters of the problem, we can split to $\ell = b \log k$ parts, and get a final output of size $sk^{O(b \log k)}$.

Theorem 9 *For any fixed b, there is a globally explicit construction for (n, k, b) anti-universal sets of size $(b/b-1)^k k^{O(\log k)} \log n$. The collection may be listed in linear time.*

The hardness result of [25] was very recently improved by Feige [16] to get the best possible result for the approximation ratio, $(1 - o(1)) \ln N$. The type of family he requires is what we termed an (n, k, b) anti-universal set. Thus he obtains:

Theorem 10 *[16] If $NP \not\subseteq DTIME[n^{O(\log\log(n))}]$, then for any fixed $c < 1$, one cannot approximate set cover to within a factor of $c \ln N$ in polynomial time.*

5.4 Depth-3 formulae for threshold functions

We now show an application of our splitter construction. The kth threshold function T_k^n of n Boolean variables is a Boolean function which is 1 iff at least k of the variables are 1. In a formula, each non-output gate has fanout exactly one, and a $\Sigma\Pi\Sigma$ formula has the form $\bigvee_u \bigwedge_v \bigvee_w L_{uvw}$, where each L is either a variable or a negated variable. The *size* of a formula is the total number of literals in it. The work of [35] shows that for k and n large and $k \leq n/2$, every $\Sigma\Pi\Sigma$ formula for T_k^n has size at least $\exp(\Omega(\sqrt{k/\ln k})) n \log n$, where $exp(x)$ denotes e^x. For k and n large and $k \leq n^{2/3}$, it is also shown in [35] that there exist $\Sigma\Pi\Sigma$ formulas for computing T_k^n with size at most $\exp(2\sqrt{k} \ln k) n \log n$; this proof is probabilistic. By using some ideas from [35] and by invoking some of our ideas from above, we present an explicit version of this upper bound, with a little loss. The proof of Theorem 11 is given in the full version.

Theorem 11 *For any pair of positive integers n and k with $k \leq n$, a monotone $\Sigma\Pi\Sigma$ formula G of size $e^{\sqrt{k} + 2\sqrt{k} \ln k} k^{O(1)} n \log n$ exists, to compute the function T_k^n; G can be constructed deterministically in time polynomial in its size.*

Acknowledgments. We thank Oded Goldreich and Ravi Sundaram for explaining the implication of the construction of (n, k)-universal sets to the non-approximability of set cover; and we thank Mike Luby for a discussion. We thank Uri Feige for explaining his results.

References

[1] N. Alon, Explicit constructions of exponential sized families of k-independent sets, Discrete Math, 58, pp. 191-193 (1986).

[2] N. Alon, L. Babai and A. Itai, *A fast and simple randomized parallel algorithm for the maximal independent set problem*, Journal of Algorithms 7 (1986), pp. 567–583.

[3] N. Alon, J. Bruck, J. Naor, M. Naor and R. Roth, *Construction of asymptotically good, low-rate error-correcting codes through pseudo-random graphs*, IEEE Transactions on Information Theory, 38 (1992), 509–516.

[4] N. Alon, O. Goldreich, J. Hastad and R. Peralta, *Simple constructions of almost k-wise independent random variables*, Random Structures and Algorithms 3 (1992), 289-304. (Addendum: Random Structures and Algorithms 4 (1993), 119-120.)

[5] N. Alon and M. Naor, *Derandomization, witnesses for Boolean matrix multiplication and construction of perfect hash functions*, to appear, *Algorithmica*. Available as Weizmann Institute TR CS94-11.

[6] N. Alon and J. H. Spencer, **The probabilistic method**, John Wiley and Sons Inc., New York, 1991.

[7] N. Alon, R. Yuster and U. Zwick, *Color-coding: a new method for finding simple paths, cycles and other small subgraphs within large graphs*, Proc. 26th ACM Symposium on Theory of Computing (1994), 326–355.

[8] M. Bellare, S. Goldwasser, C. Lund and A. Russell, *Efficient probabilistically checkable proofs and application to approximation*, Proc. 25th ACM Symposium on Theory of Computing (1993), 294–304.

[9] B. Berger and J. Rompel, *Simulating* $(\log^c n)$-*wise independence in NC*, J. of the ACM 38 (1991), pp. 1026–1046.

[10] B. Berger, J. Rompel and P. Shor, *Efficient NC Algorithms for Set Cover with Applications to Learning and Geometry*, Journal of Computer and System Sciences 49, 1994, 454–477.

[11] M. Bellare and J. Rompel, *Randomness-efficient oblivious sampling*, Proc. 35th IEEE Symposium on Foundations of Computer Science (1994), 276–287.

[12] A. Blum and S. Rudich, *Fast learning of k-term DNF formulas with queries*, Proc. ACM Symposium on Theory of Computing (1992), 382–389.

[13] B. Becker and H. U. Simon, *How robust is the n-cube?*, Information and Computation 77, 1988, pp. 162–178.

[14] N. Bshouty, *Exact Learning via the Monotone Theory* Proc. 34th IEEE Symposium on Foundations of Computer Science (1993).

[15] B. Chor, O. Goldreich, J. Hastad, J. Friedman, S. Rudich and R. Smolensky, *The Bit Extraction Problem or t-Resilient Functions*, Proceedings of the 26th Annual Symposium on Foundations of Computer Science, 1985, 396-407.

[16] U. Feige, *A threshold of* $\ln n$ *for approximating set cover*, manuscript, 1995.

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

[18] M. L. Fredman, J. Komlós and E. Szemerédi, *Storing a Sparse Table with $O(1)$ Worst Case Access Time*, J. of the ACM 31 (1984), 538–544.

[19] R. M. Karp and A. Wigderson, *A Fast Parallel Algorithm for the Maximal Independent Set Problem*, J. of the ACM 32 (1985), 762–773.

[20] D. J. Kleitman and J. H. Spencer, *Families of k-independent sets*, Discrete Math. 6 (1973), pp. 255–262.

[21] J. Körner, *Fredman-Komlós bounds and information theory*, SIAM. J. Alg. Disc. Meth. 7 (1986), pp. 560–570.

[22] R. E. Krichevsky, *Occam's razor, partially specified Boolean functions, String matching and independent sets*, Information and Computation 108, 1994, pp. 158–174.

[23] M. Luby, *A simple parallel algorithm for the maximal independent set problem*, SIAM J. Comp. 15 (1986), pp. 1036–1053.

[24] M. Luby, *Removing randomness in parallel computation without a processor penalty*, Journal of Computer and System Sciences 47 (1993), pp. 250–286.

[25] C. Lund and M. Yannakakis, *On the hardness of approximating minimization problems*, J. of the ACM 41 (1994), pp. 960–981.

[26] K. Mehlhorn, **Data Structures and Algorithms 1: Sorting and Searching**, Springer-Verlag, Berlin, 1984.

[27] A. Mayer, M. Naor and L. Stockmeyer, *Local computations on static and dynamic graphs*, Proceeding of the third ISTCS, 1995, pp. 268–278. Full version available as Weizmann Institute TR CS95-05.

[28] R. Motwani and P. Raghavan, **Randomized Algorithms**, Cambridge University Press, 1995.

[29] R. Motwani, J. Naor and M. Naor, *The probabilistic method yields deterministic parallel algorithms*, Journal of Computer and System Sciences (1994), pp. 130–143.

[30] J. Naor and M. Naor, *Small-bias probability spaces: efficient constructions and applications*, SIAM J. on Computing 22, 1993, pp. 838–856.

[31] M. Naor, L. J. Schulman and A. Srinivasan, *Splitters and near-optimal derandomization*, In preparation.

[32] A. Nilli, *Perfect hashing and probability*, Combinatorics, Probability and Computing 3, 1994, pp. 407–409.

[33] A. Orlitsky, *Worst-case interactive communication I: two messages are almost optimal* IEEE Trans. Infor. Theory 36, pp. 1111-1126, 1990.

[34] J. Radhakrishnan, *Improved Bounds for Covering Complete Uniform Hypergraphs*, Information Processing Letters 41 (1992), pp. 203–207.

[35] J. Radhakrishnan, $\Sigma\Pi\Sigma$ *threshold formulas*, Combinatorica 14 (1994), pp. 345–374.

[36] P. Raghavan, *Probabilistic construction of deterministic algorithms: approximating packing integer programs*, Journal of Computer and System Sciences 37 (1988), pp. 130–143.

[37] R. Raz, *A parallel repetition theorem*, Proc. 27th ACM Symposium on Theory of Computing (1995), pp. 447–456.

[38] H. Robbins, *A remark on Stirling's formula*, Amer. Math. Monthly, 62 (1955), pp. 26–29.

[39] J. H. Spencer, **Ten Lectures on the Probabilistic Method**, SIAM, Philadelphia, 1987.

[40] J. P. Schmidt and A. Siegel, *The Spatial Complexity of oblivious k-probe hash functions*, SIAM J. Comp. 19 (1990), pp. 775–786.

[41] G. Seroussi and N. Bshouty, *Vector sets for exhaustive testing of logic circuits*, IEEE Transactions on Information Theory 34 (1988), pp. 513-522.

[42] M. Szegedy and S. Vishwanathan, Locality based graph coloring, Proc. 25th ACM Symposium on Theory of Computing (1993), pp. 201–207.