(De)randomized Construction of Small Sample Spaces in $\mathcal{NC}$

David R. Karger*
AT&T Bell Laboratories
Murray Hill, NJ 07974
karger@lcs.mit.edu

Daphne Koller†
Computer Science Division
University of California
Berkeley, CA 94720
daphne@cs.berkeley.edu

Abstract

Koller and Megiddo introduced the paradigm of constructing compact distributions that satisfy a given set of constraints, and showed how it can be used to efficiently derandomize certain types of algorithm. In this paper, we significantly extend their results in two ways. First, we show how their approach can be applied to deal with expectation oracles *of the type used in the* method of conditional probabilities. *More importantly, we provide the first* parallel *($\mathcal{NC}$) algorithm for constructing a small sample-space distribution that satisfies the given constraints up to a small* relative *error. This algorithm deals with constraints over any event that can be verified by finite automata, including all* independence constraints *as well as constraints over events relating to the parity or sum of a certain set of variables. Our construction relies on a new and independently interesting parallel algorithm for converting a solution to a linear system into an almost basic approximate solution to the same system. We use these techniques in the first $\mathcal{NC}$ derandomization of an algorithm for constructing large independent sets in d-uniform hypergraphs for* arbitrary d. *We also show how the linear programming perspective suggests new proof techniques which might be useful in general probabilistic analysis.*

*Supported by a Hertz Foundation Graduate Fellowship and by NSF Young Investigator Award CCR-9357849, with matching funds from IBM, Schlumberger Foundation, Shell Foundation and Xerox Corporation

†Supported by a University of California President's Postdoctoral Fellowship and by the Air Force Office of Scientific Research (AFSC) under Contract F49620-91-C-0080. Some of this work was done while at Stanford University.

1 Introduction

The probabilistic method of proving existence of combinatorial objects has been very successful (see, for example, Spencer [Spe87]). The underlying idea is as follows. Consider a finite set Ω whose elements are classified as "good" and "bad." Suppose we wish to prove existence of at least one "good" element within Ω. The proof proceeds by constructing a *probability distribution* ζ *over* Ω (i.e., a function $\zeta : \Omega \to [0,1]$ such that $\sum_{\boldsymbol{x}\in\Omega} \zeta(\boldsymbol{x}) = 1$) and showing that if we sample from Ω according to this distribution then the probability of picking a good element is positive. Probabilistic proofs often yield randomized algorithms for constructing a good element. In particular, many randomized algorithms are a special case of this technique, where the sample space Ω contains the various sequences of random choices which could be made by the algorithm, and the "good" elements are those sequences of random choices that make the algorithm work in the desired way (such as running quickly or giving the correct output).

It is often desirable to *derandomize* algorithms. Perhaps the simplest way to do this is to enumerate all of the choice sequences in Ω and try each until we find the good one guaranteed by the analysis. Unfortunately, the size of Ω is typically exponential in the size of the problem; for example, the sample space of n independent random bits contains 2^n points.

A way around this problem is to make more careful use of the distribution ζ from the analysis. Since we have proven that there is a nonzero probability of a good point, there must be some good point to which ζ assigns a nonzero probability. It therefore suffices to enumerate only those points in $S(\zeta) = \{\boldsymbol{x} \in \Omega \mid \zeta(\boldsymbol{x}) > 0\}$; this set is called the *support* or *sample space* of ζ, and its cardinality is called the *size* of the distribution. This approach may let us[1] $\Downarrow$

[1]**Old Text:** It may therefore be possible to $\Leftarrow$

efficiently derandomize an algorithm by replacing the original distribution with one of "small" (polynomial) size. In order for this procedure to work, the new distribution must agree with the original one to the extent that the correctness proof of the algorithm remains valid. The correctness proof often relies on certain assumptions about the distribution; that is, the distribution is assumed to satisfy certain constraints. For example, there may be a constraint on the probability of some *event* $Q \subseteq \Omega$, i.e., an equality of the form $\Pr(Q) \stackrel{\text{def}}{=} \sum_{\mathbf{x} \in Q} \zeta(\mathbf{x}) = \pi$, where $0 \leq \pi \leq 1$. Alternatively, there might be a constraint bounding the expectation or variance of some random variable(function over the sample space). If the new distribution satisfies all the constraints that are relied upon by the correctness proof, then the algorithm remains correct using the new distribution; no new analysis is needed.

What properties of a distribution are typically required by correctness proofs? The original distribution is almost always induced by a set of *independent* random variables $X_1, \ldots, X_n$; the proof of correctness depends on the properties of this independent distribution. In many cases, however, full independence is not necessary. For example, the necessary constraints are often satisfied by a *d-wise independent distribution*—one satisfying all the *independence constraints*, asserting that the events of the form $[X_{i_1} = b_{i_1},\ X_{i_2} = b_{i_2}, \ldots,\ X_{i_d} = b_{i_d}]$ should have the same probability as if the variables were independent.

History

Most previous work in derandomization has focused on constructing small d-wise independent distributions for small d [Jof74, Lub86, ABI86, KM94, NN93, AGHP90, AMN92, EGL+92, BR91, MNN89]. Furthermore, the emphasis has been on constructions in $\mathcal{NC}$, so as to allow a derandomization of parallel algorithms. Most of these works construct a distribution that only approximately satisfies some of the required constraints. The early works [Jof74, Lub86, ABI86] generate distributions that err on the probabilities $\Pr(X_i = b_i)$, but precisely satisfy the actual independence constraints such as those on $\Pr(X_i = b_i,\ X_j = b_j)$. This approach is inherently limited, since it was shown by Chor *et al.* [CGH+85] that any sample space of n d-wise independent random bits has cardinality $\Omega(n^{[d/2]})$. Karloff and Mansour [KM94] extended this result to biased random bits, and showed that, in certain cases, the smallest sample space maintaining d-wise independence has size $\Omega(n^d)$. Therefore this approach can be used only if the desired d is constant.

These bounds can be circumvented by allowing some error in the independence constraints. In [NN93, AGHP90, AMN92, EGL+92], the probabilities of the relevant events (as described above) are required to be within an *additive* factor of $\pm\epsilon$. The size of the resulting distribution is polynomial in $1/\epsilon$, so that we must choose ϵ to be at least $1/\text{poly}(n)$. For $d = O(\log n)$, this approach yields a polynomial-size distribution that is nearly d-wise independent (as compared to the lower bound of $\Omega(n^{\log n})$ for truly d-wise independent distributions). However, the probabilities of events involving independence of more than $O(\log n)$ bits are swamped by the error factor, so the constraints on their probabilities are meaningless. Thus, this approach is not applicable to cases where we are interested in events relating to more than $O(\log n)$ of the variables.

The above approaches all generate a d-wise independent distribution, and then apply it uniformly to any algorithm for which d is an upper bound on the degree of independence required ([BR91] and [MNN89] also present a parallel algorithm for searching the support of certain implicitly constructed polylog$n$-wise independent distributions). In a different paradigm, it is possible to directly examine the constraints imposed by a specific algorithm and possibly even a specific input, and then construct a distribution to satisfy precisely those constraints. This approach was initiated by Schulman [Sch92] and investigated in a more general way by Koller and Megiddo [KM93]. Koller and Megiddo viewed constraints as linear equations on variables representing the probabilities of the points in the sample space. This allows them to apply techniques from linear programming to the problem of derandomization. In particular, they show that for any consistent set $\mathcal{C}$ of constraints on event probabilities there exists a size $|\mathcal{C}|$ distribution also satisfying $\mathcal{C}$. Their proof is not constructive. However, they then restrict attention to independence constraints, and present a polynomial time construction of small-size distributions.

The advantage of this approach is that the size of the distribution depends only on the number of constraints actually used. For example, an algorithm may choose edges in a graph by associating a random bit with each edge. An event whose probability may be relevant in analyzing this algorithm is "no edge adjacent to a node v is chosen". Using the other approaches, independence must be enforced among all neighbors of a vertex; the sample space would then grow as 2^Δ where Δ is the maximum node degree. Using [KM93], there is only one event per node, resulting in a sample space of size n (the number of nodes in the graph). In this example, the constraints depend on the edge structure of the input graph. This input-dependence is typical of the constraint-based approach. Therefore, unlike most previous constructions, the distribution cannot be precomputed. Furthermore, the inherently sequential nature of the [KM93] construction prevents their techniques from being applied to the derandomization of parallel algorithms.

New results

We extend the approach of [KM93] in several ways. In Sections 2 and 3, we show that in addition to constraining the probabilities of events, we can also require that certain random variables have the correct expectation (and similarly the correct variance, and other higher moments), and can still construct a small satisfying distribution in polynomial time. All other constructions (including that of [KM93]) deal solely with independence constraints; the additional power can be quite important. For example, if we wish to maintain the distribution of the parity of $X_1 + \cdots + X_n$, an exponential number of independence constraints are required. (Distributions satisfying parity constraints may be useful for cryptographic purposes [Kus93].) Our construction deals with this as a single constraint. Since our sample space is polynomial
so long as
whenever the number of constraints we wish to satisfy is polynomial, we are able to construct polynomial-size distributions where previous approaches could not.

In addition, we are able to parallelize the construction. In Section 4 we show that for a large class of constraints, it is in fact possible to construct a polynomial-size distribution *approximately* satisfying the constraints in $\mathcal{NC}$. Our parallel construction is more limited than our sequential one, but still handles a large (though difficult to characterize) class of constraints. If the elements of Ω are thought of as strings (of bits, for example), the class includes all constraints on the probabilities of events that can be recognized by finite automata of polynomial size.[2] This class contains independence constraints, parity constraints, constraints on the sum of the random variables, and so on.

The key to our parallel construction is *basis crashing*—the process of transforming a given solution to a linear system into a basic solution to the same system. No parallel basis crashing algorithms were known previously. In Section 6 we present the first parallel algorithm for *approximate* basis crashing, and use it to construct small distributions approximately satisfying the constraints. Parallel basis crashing is of independent interest and Section 6 can be read independent of the rest of the paper. Indeed, it has other applications. For example, our algorithm dovetails well with an $\mathcal{NC}$ algorithm of Luby and Nisan [LN93] for finding approximate solutions to positive (packing and covering) linear programs. By applying our algorithm in a postprocessing step, we can arrange for the solution to the linear program to have at most $m \log^{1+o(1)} m$ nonzero components, where m is the number of constraints, while maintaining the same accuracy as the original output of the [LN93] algorithm.

The fact that we can construct a distribution in $\mathcal{NC}$ allows us to derandomize parallel algorithms. We apply our techniques to the problem of finding a large independent set in a d-uniform hypergraph. The underlying randomized algorithm, described by Alon, Babai, and Itai [ABI86], was derandomized in the same paper for fixed values of d. It was later derandomized also for $d = O(\operatorname{poly}\log\ n)$ by Berger and Rompel [BR91] and Motwani, Naor, and Naor [MNN89]. Koller and Megiddo [KM93] gave a derandomization of this algorithm for arbitrary d, but their derandomization is inherently sequential. Our use of more generalized constraints allows us to reduce the number of constraints, and thus the size of the linear systems that must be solved, by a significant factor. More importantly, we show how this algorithm can be derandomized in $\mathcal{NC}$ for any d. To the best of our knowledge, this is the first parallel algorithm for this problem (for large d).

As we mentioned, our parallel construction only approximately satisfies the constraints. However, our approximation is significantly better than that obtained by the ϵ-independence constructions, since it bounds the *relative* rather than the absolute error in the probabilities of events. That is, for any $\epsilon = \Omega(1/n^d)$, we can construct a polynomial size sample space in which the probability of a constrained event is at most $(1 \pm \epsilon)$ *times* its correct value, rather than within an absolute error of ϵ *plus or minus* its correct value. To gain this power, we give up the ability to deal with all $O(\log n)$-wise independence constraints at once; instead, we ap-

[2] The problem of "looking random" to a group of testers has been studied by Nisan [Nis90], who constructs a sample space of size $n^{O(\log(n))}$ that simultaneously satisfies the constraints associated with *all* polynomial size finite automata. It has also been looked at by Blum and Goldreich [BG92], who take a more general look at the relationship between the power of a computational model and the pseudorandom sequences which can "fool" it.

proximately satisfy only a polynomial number of constraints. The relative error bound, on the other hand, allows us to avoid the problem encountered by the ϵ-independent constructions: our technique can construct distributions meaningfully approximating the probabilities of *arbitrarily low probability* events.

The approach described here also suggests a new perspective on probabilistic analysis which might be useful in other circumstances. As an example of the use of the perspective, we consider a "sensitivity analysis" of approximately correct distributions. In general, using an approximately correct distribution typically requires a return to the proof to confirm that it still works. In Section 5, we present a theorem showing that this is unnecessary in many cases, if relative errors are used. We consider a general class of correctness proofs which depend only on the probabilities of certain events being correct, and show that such proofs necessarily remain correct if the probabilities of the important events vary slightly (in *relative* terms) from their correct values. The proof of this theorem uses the duality theory of linear programming, applied to the linear equations induced by constraints. This demonstrates yet again the power inherent in viewing probabilistic constraints in the framework of linear systems.

2 Constraints as linear equations

As we mentioned in the introduction, Koller and Megiddo [KM93] introduced the paradigm of constructing distributions that satisfy only the constraints actually required for the correctness of the algorithm. In this section and the next, we review their results and extend their applicability to a much larger class of constraints. We consider *random variables over* Ω, i.e., functions from Ω to $\Re$. For a distribution ζ and random variable $\mathcal{F}$, $E_\zeta(\mathcal{F})$ denotes the expectation of $\mathcal{F}$ taken over distribution ζ.

Definition 2.1: *An* expectation constraint ρ *over* Ω *has the form* $E(\mathcal{F}) = \gamma$, *where* $\mathcal{F}$ *is an arbitrary random variable on* Ω. *A probability distribution* ζ *over* Ω satisfies ρ *if* $E_\zeta(\mathcal{F}) = \gamma$. ∎

In [KM93], Koller and Megiddo allowed only *probabilistic constraints* of the form $\Pr(Q) = \pi$, for $Q \subseteq \Omega$. This is clearly a special case of our definition, since we can define $\mathcal{F}$ to be an indicator variable: $\mathcal{F}(\boldsymbol{x}) = 1$ for $\boldsymbol{x} \in Q$ and 0 otherwise. In this case, $E(\mathcal{F}) = \Pr(Q)$. However, expectation constraints are clearly more general; we can, for example, require that the variance of a certain random variable X take on a certain value by defining $\mathcal{F}(\boldsymbol{x}) = (\boldsymbol{x} - E(X))^2$.

For the remainder of this section, fix $\mathcal{C}$ to be a set of m expectation constraints $\{[E(\mathcal{F}_i) = \gamma_i] : i = 1, \ldots, m\}$. Furthermore, assume that $[\Pr(\Omega) = 1] \in \mathcal{C}$. Koller and Megiddo introduced the idea of viewing constraints as linear equations on variables representing the probabilities of the points in the sample space. The basic insight is as follows. Let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ denote the points in Ω. If we use variables v_j to represent the probabilities $\zeta(\boldsymbol{x}_j)$, then a constraint $[E(\mathcal{F}_i) = \gamma_i]$ translates into a linear equation over these variables: $\sum_{j=1}^{N} \mathcal{F}_i(\boldsymbol{x}_j) v_j = \gamma_i$. (Note that $\mathcal{F}_i(\boldsymbol{x}_j)$ is a known constant.) We can use the techniques of [KM93] to prove the following theorem:

Theorem 2.2: *If $\mathcal{C}$ is satisfied by some distribution ζ, then there exists a size m distribution ζ' satisfying $\mathcal{C}$.*

The main idea of the proof is that ζ encodes a non-negative solution to the system of linear equations induced by $\mathcal{C}$. A standard result from linear programming is that the existence of such a solution implies the existence of a *basic* non-negative solution to this system—one that has the minimum number of positive entries. This basic solution corresponds to a distribution ζ' whose support is precisely the set of positive entries in the basic solution. The process of constructing ζ' from ζ is known as *basis crashing*. The fastest basis crashing algorithm known is due to Beling and Megiddo [BM93]; it allows us to prove the following result:

Theorem 2.3: *Given a sparse representation of a distribution ζ satisfying $\mathcal{C}$, it is possible to construct a size m distribution ζ' using $|S(\zeta)|$ evaluations of each $\mathcal{F}_i$ in $\mathcal{C}$, and $O(|S(\zeta)| \cdot m^{1.62})$ arithmetic operations.*

3 Sequential constructions

In the previous section, we described an algorithm that takes a distribution satisfying a given set of constraints, and by basis crashing finds a distribution with a smaller support that still satisfies them. But in order to use this algorithm, we must already have a distribution ζ which satisfies the constraints with a polynomial-size support; otherwise, our algorithm does not work in polynomial time. Our goal is to construct a distribution directly from the constraints that it must satisfy. As shown in [KM93], even the problem of deciding whether a given set of constraints is consistent is $\mathcal{NP}$-hard in general. We can circumvent this problem by requiring that the constraints be consistent with a fixed known *independent distribution*. Let $X_1, \dots, X_n$ be discrete random variables with a finite range. We assume for simplicity that $X_1, \dots, X_n$ all have the same range $\{0, \dots, r-1\}$; our results easily extend to the more general case. For the remainder of this paper, fix a set of values $\{p_{kb} : k = 1, \dots, n;\ b = 0, \dots, r-1\}$ where $\sum_{b=0}^{r-1} p_{kb} = 1$ for all k and $p_{kb} \geq 0$ for all k, b. Fix ϑ to be the *independent distribution* induced by $X_1, \dots, X_n$ when these are independent and each X_k is distributed as $\Pr(X_k = b) = p_{kb}$ for all b.

As we said, we wish ζ to satisfy constraints of the form $E(\mathcal{F}) = \gamma$, where these are known to be satisfied by ϑ. But then, γ is necessarily equal to $E_\vartheta(\mathcal{F})$. This motivates the following definition:

Definition 3.1: *A distribution ζ* fools *a function $\mathcal{F}$ if $E_\zeta(\mathcal{F}) = E_\vartheta(\mathcal{F})$. It fools an event Q if it fools the indicator function which is 1 on Q and 0 elsewhere. It fools a set of functions if it fools each one.* ∎

We can view a set of functions $\{\mathcal{F}_1, \dots, \mathcal{F}_m\}$ as a set $\mathcal{C}$ of constraints on ζ. A function $\mathcal{F}$ can also be viewed as an indicator (or measurement) on the sample space, whose value we would like to maintain.

The sequential construction of [KM93] works by "derandomizing" one variable X_i at a time, using the basis crashing approach of the previous section as a subroutine. We take a similar approach for our more general expectation constraints. This requires that we be able to define a distribution over a subset of the variables, so that it satisfies a version of the constraints restricted to that subset.

Definition 3.2: *Given a random variable $\mathcal{F}(X_1, \dots, X_n)$, the* restriction *$\Pi_{X_i, \dots, X_j}\mathcal{F}$ is a random variable (function) on $X_i, \dots, X_j$ with value $E_\vartheta(\mathcal{F} \mid X_i, \dots, X_j)$.* ∎

As a starting point, the distribution $\zeta_0(\langle\rangle) = 1$ clearly fools $\Pi_\emptyset \mathcal{F}_i = E_\vartheta(\mathcal{F}_i)$. Suppose now that we have a distribution ζ_{k-1} on $X_1, \dots, X_{k-1}$ which fools $\Pi_{X_1, \dots, X_{k-1}}\mathcal{F}_i$ for $i = 1, \dots, m$. We use this to define a distribution ζ_k on $X_1, \dots, X_k$ as follows. Define an intermediate distribution

$$\varrho_k(\langle b_1, \dots, b_{k-1}, b_k\rangle) = \zeta_{k-1}(\langle b_1, \dots, b_{k-1}\rangle) \cdot p_{kb_k}.$$

That is, ϱ_k is constructed as the cross-product of two distributions: ζ_{k-1} (over $X_1, \dots, X_{k-1}$) and the complete distribution over the single variable X_k. It is easy to prove that if ζ_{k-1} fools $\Pi_{X_1, \dots, X_{k-1}}\mathcal{F}_i$, then ϱ_k fools $\Pi_{X_1, \dots, X_k}\mathcal{F}_i$. By assumption, ζ_{k-1} has size m, so $|S(\varrho_k)| \leq rm$. We now use Theorem 2.3 to construct a size m distribution satisfying the same set of constraints. In order to do so, we must be able to write down the constraints on ζ_k as a system of linear equations. In general, a distribution ζ over $X_1, \dots, X_k$ fools $\Pi_{X_1, \dots, X_k}\mathcal{F}_i$ if and only if

$$\sum_{\boldsymbol{b} \in S(\zeta)} \zeta(\boldsymbol{b})[\Pi_{X_1, \dots, X_k}\mathcal{F}_i](\boldsymbol{b}) = E_\vartheta(\mathcal{F}_i).$$

This induces a system in the required form, which is satisfied by ϱ_k. However, to find a basic solution to this system, we must write it explicitly, so that we need to know the values of the expressions $[\Pi_{X_1, \dots, X_k}\mathcal{F}_i](\boldsymbol{b}) = E_\vartheta(\mathcal{F}_i \mid X_1 = b_1, \dots, X_k = b_k)$. This motivates the following definition:

Definition 3.3: *A* conditional expectation oracle *for a function $\mathcal{F}$ is a procedure that can compute, for any k and any $b_1, \dots, b_k \in \{0, \dots, r-1\}$, the conditional expectation $E_\vartheta(\mathcal{F} \mid X_1 = b_1, \dots, X_k = b_k)$.* ∎

We therefore assume that each function in $\mathcal{C}$ has a conditional expectation oracle. Given this assumption, we can apply Theorem 2.3 to compute the distribution ζ_k required for this inductive step. The distribution ζ_n that terminates the induction will fool $\Pi_{X_1, \dots, X_n}\mathcal{F}_i = \mathcal{F}_i$ for all i.

Theorem 3.4: *Let $\mathcal{C}$ be a set of m functions, each with an accompanying conditional expectation oracle. Then we can compute a size m distribution ζ fooling the functions in $\mathcal{C}$ in strongly polynomial time, using $O(rnm^{2.62})$ arithmetic operations and rnm calls to the conditional expectation oracle of each $\mathcal{F}_i$.*

The assumption that we have a conditional expectation oracle for each constraint is clearly crucial to this construction. A similar assumption appears in the *method of conditional probabilities* [ES73] (see also [Spe87]). The idea there is to perform a binary search of the sample space for a good point. At step k of the search, the current sample space is split into two halves according to the value of X_k (which, for the sake of convenience, is assumed to be binary). The algorithm computes the *conditional probability* that a good point exists in each half, and then restricts the search to the half where this conditional probability is higher. The method thus also requires an oracle for computing conditional probabilities given values for a certain subset of the variables. Since conditional probabilities are a special case of conditional expectations, this is a special case of our approach. That is, if we define $\mathcal{F}(b_1,\ldots,b_n)$ to be 1 if $(b_1,\ldots,b_n)$ is good and 0 otherwise, our algorithm will also essentially conduct a binary search for a good point. However, our method can also be applied to cases where we cannot compute the probability of a good point. In particular, the conditional probability that a good point exists might involve complex computations. We might, however, be able to prove that this probability is sufficiently large, given that certain constraints hold. If the conditional expectations of these constraints are easy to compute, we can use our technique to construct a small sample space guaranteed to contain a good point. Hence, we can view our approach as a hybrid between the method of conditional probabilities and the methods relying on partial independence, described in the introduction. We construct a sample space precisely satisfying only the desired independence constraints; conditional expectations are used to find, not a single good point, but a set of points that form the support for an appropriate distribution.

4 Constructing small distributions in $\mathcal{NC}$

The above construction seems inherently sequential for two reasons. First, the algorithm only adds one variable at a time to the sample space we are constructing, thus requiring n iterations. More importantly, at each iteration it needs to reduce the support of the intermediate distribution. This stage relies intrinsically on the process of basis crashing, a problem for which no parallel algorithms are known. Nevertheless, we show that is possible to circumvent both these obstacles, and construct small distributions in $\mathcal{NC}$. We begin by addressing the first obstacle. For ease of presentation, we assume for now that we have an $\mathcal{NC}$ function `ReduceSupport` that takes a distribution ζ fooling certain functions and outputs a polynomial size distribution that fools the same functions (recall that a function is fooled if it has the same expectation over the new distribution).

There is now an apparently straightforward parallelization of our sequential construction: In our sequential construction of Section 3, the auxiliary distribution ϱ_ℓ over the variables $X_1,\ldots,X_\ell$ is constructed as the cross-product of two distributions: $\zeta_{\ell-1}$ over the variables $X_1,\ldots,X_{\ell-1}$ and the independent distribution over the single variable X_ℓ. Instead we let $g=\lceil\frac{l+h}{2}\rceil$ and construct a distribution ϱ over the variables $X_l,\ldots,X_h$ as the cross-product of a distribution ζ' over $X_l,\ldots,X_g$ and a distribution ζ'' over $X_{g+1},\ldots,X_h$. These smaller distributions ζ',ζ'' are constructed recursively. We will then use the `ReduceSupport` algorithm, as before, to construct a smaller distribution ζ satisfying the same constraints as ϱ. The new distribution ζ will satisfy I.e.,
$\Pi_{X_l,\ldots,X_h}\mathcal{C}=\{\Pi_{X_l,\ldots,X_h}\mathcal{F}\mid\mathcal{F}\in\mathcal{C}\}$. More formally, consider the following algorithm, which we initially call as `BuildDistribution`$(X_1,\ldots,X_n)$: with

Function `BuildDistribution`$(X_l,\ldots,X_h)$
  If $l=h$ then
    For all $b\in\{0,\ldots,r-1\}$:
      $\varrho(b)\leftarrow p_{hb}$
  else $(l<h)$
    $g=\lceil\frac{l+h}{2}\rceil$
    $\zeta'\leftarrow$ `BuildDistribution`$(X_l,\ldots,X_g)$
    $\zeta''\leftarrow$ `BuildDistribution`$(X_{g+1},\ldots,X_h)$
    For all $b_l,\ldots,b_h$:
      $\varrho(b_l,\ldots,b_h)\leftarrow\zeta'(b_l,\ldots,b_g)\cdot\zeta''(b_{g+1},\ldots,b_h)$
  $\zeta\leftarrow$ `ReduceSupport`$(\varrho,\Pi_{X_l,\ldots,X_h}\mathcal{C})$
  Return(ζ).

Unfortunately, this construction is not as powerful as the one from Section 3. It is quite possible that ζ' fools $\Pi_{X_l\ldots X_g}(\mathcal{F})$ and ζ'' fools $\Pi_{X_{g+1}\ldots X_h}(\mathcal{F})$, while ϱ does not fool $\Pi_{X_l\ldots X_h}(\mathcal{F})$. Consider, for example, the sample space on two unbiased independent bits X_1 and X_2, and let $\mathcal{F}$ be the parity function. Let $\zeta'(1)=\zeta''(1)=1$, in other words take two subdistributions (on X_1 and X_2 respectively) which assign value 1 with probability 1. Then ζ' fools $\Pi_{X_1}(\mathcal{F})$, since if X_2 is an unbiased random bit then the parity of X_1X_2 has the right distribution regardless of the distribution of X_1. Similarly, ζ'' fools $\Pi_{X_2}(\mathcal{F})$. However, the cross-product of these two distribution assigns 1 to each of X_1 and X_2 with probability 1 and therefore gives the wrong distribution on $\mathcal{F}$.

In the full paper, we investigate the difficulties further and provide a description of a general class of functions that do compose properly. For the purposes of this abstract, we restrict attention to a simpler class of functions. Consider the case where $X_1,\ldots,X_n$ are random variables taking values over some domain D (above we took D to be $\{0,\ldots,r-1\}$), and the sample space Ω is defined to be D^n. Now, a point in Ω can be thought of as a string over the alphabet D. This allows the following definition.

Definition 4.1: *A* regular function $\mathcal{F}$ *over the sample space is a function induced by a deterministic finite automaton with polynomially many labeled states, such that the label of the final state on input $b_1,\ldots,b_n$ encodes the value of $\mathcal{F}$ on that sample point. A distribution* fools *a regular function $\mathcal{F}$ if the distribution of*

$\mathcal{F}$ is as it would be if the input were drawn ¿from the independent distribution ϑ. ∎

This definition of fooling seems to clash somehwat with the previous one as it constrains the distribution of a function rather than its expectation. However, the two definitions are equivalent: we can constrain the distribution of final states in the automaton by constraining polynomially many indicator functions which identify the final state, and we can have certainly constrained the expectation of a regular function if we succeed in constraining the distribution.

It is important to note that the problem of deciding whether certain variables have taken on certain fixed values can be decided by a finite automaton. Thus, the class of constraints on regular functions contains the class of independence constraints. Hence, our results in this section provide a parallel construction that works in all cases covered by the results of [KM93]. Furthermore, it covers a large number of other interesting cases, such as parity, sum modulo k, and threshold functions.

Theorem 4.2 : *Given an $\mathcal{NC}$ implementation of `ReduceSupport`, it is possible to construct in $\mathcal{NC}$ a polynomial size distribution which fools a set $\mathcal{C}$ of regular functions.*

Proof: Intuitively, our construction fools the events that a random substring causes a transition between any given pair of states. More formally, fix attention on one particular regular function $\mathcal{F}$ with a corresponding automaton $\boldsymbol{M}$ on states $\{s_i\}$. We show how to construct a distribution fooling $\mathcal{F}$; rather than doing this directly, we construct a distribution that fools the *transition probabilities* of the automaton $\boldsymbol{M}$. Let $s_i \xrightarrow{\boldsymbol{b}} s_j$ denote the event that M, starting at state s_i, will upon reading string $\boldsymbol{b}$ arrive at state s_j. These events (over all pairs of states s_i and s_j) are the events which we aim to fool.

Now consider the recursive function `BuildDistribution`. We assume by induction that the two recursive calls return distributions ζ' and ζ'' which respectively fool $s_i \xrightarrow{X_l \cdots X_g} s_j$ and $s_i \xrightarrow{X_{g+1} \cdots X_h} s_j$ for all s_i and s_j. Then it is easy to prove that ϱ fools $s_i \xrightarrow{X_l \cdots X_h} s_j$. This follows from the fact that

$$\Pr[s_i \xrightarrow{X_l \cdots X_h} s_j] = \sum_k \Pr[s_i \xrightarrow{X_l \cdots X_g} s_k] \cdot \Pr[s_k \xrightarrow{X_{g+1} \cdots X_h} s_j].$$

If we then run `ReduceSupport` with this set of constraints, the resulting distribution ζ will also fool these transition events, thus maintaining the inductive hypothesis. It follows that under the final distribution ζ over $X_1, \ldots, X_n$, if s_0 is the start state of the automaton, then the probability that any particular state s_k is the final state, namely $\Pr[s_0 \xrightarrow{X_1 \cdots X_n} s_k]$, has the correct value. Hence, the probability that the regular function takes on any particular value is also correct. ∎

This result rests entirely on the assumption that we have an $\mathcal{NC}$ implementation of `ReduceSupport`. How can we reduce the support of a distribution in parallel? The main technique we use is random sampling. To explain the basic intuition, we give the following simplified algorithm. Suppose we have a distribution ζ satisfying certain constraints $\Pr[Q_i] = \pi_i$. Take k random samples from this distribution, yielding a multiset S. Construct a new distribution ζ' by assigning probability $1/k$ to each element of S. The expected size of $S \cap Q_i$ is $k\pi_i$. Indeed, so long as $k >> 1/(\delta^2 \pi_i)$, the size of $S \cap Q_i$ is in the range $(1 \pm \delta)k\pi_i$ with high probability (by the Chernoff bound). It follows that $\Pr_{\zeta'}(Q_i) \in (1 \pm \delta)\pi_i$.

There are three main barriers to using this naive algorithm as our $\mathcal{NC}$ implementation of `ReduceSupport`. The first is that it requires a very large number of samples if π_i is small. We can compensate for this by sampling more often from points inside low-probability events. The second is that there is no point using a randomized algorithm as a subroutine in a derandomization procedure. We therefore "derandomize" our derandomization routine using the deterministic *lattice approximation* techniques developed by [MNN89]. Finally, the distribution ζ' does not satisfy the constraints precisely, but only approximately. While our techniques do not allow us to solve this problem, it turns out that this is not a barrier to using our approach for derandomization (see Section 5): "almost" satisfying the constraints is usually good enough.

Definition 4.3: *A distribution ζ ϵ-fools a function $\mathcal{F}$ if $E_\zeta(\mathcal{F}) \in (1 \pm \epsilon)E_\vartheta(\mathcal{F})$.* ∎

In Section 6, we describe a derandomized algorithm for approximate basis crashing, based on the ideas described above. As a corollary to Theorem 6 below, we obtain the following parallel but approximate analogue to Theorem 2.3.

Corollary 4.4 : *There exists an $\mathcal{NC}$ algorithm `ReduceSupport` that, given a sparse representation of a distribution ζ fooling a set of nonnegative functions*

$\mathcal{C}$, constructs a distribution ζ' that $(\delta^{0.5-o(1)}\sqrt{\log m})$-fools all the $\mathcal{F}_i$ in $\mathcal{C}$ and such that $|S(\zeta')| \leq m/\delta + o(m/\delta)$. The algorithm requires $|S(\zeta)|$ evaluations of $\mathcal{F}_i$ for each $i = 1, \dots, m$.

In order to complete the parallelization of our construction, we combine Corollary 4.4 with the recursive construction on which Theorem 4.2 is based. It is easy to verify that the errors accumulate in a straightforward way, so that by taking $\epsilon = n\sqrt{\delta \lg m}$ we achieve the following result:

Theorem 4.5: *There exists an $\mathcal{NC}$ algorithm that, given a set $\mathcal{C}$ of regular functions, constructs a distribution ζ that ϵ-fools all the $\mathcal{F}_i$ in $\mathcal{C}$ and such that $|S(\zeta)| \leq mn^{2+o(1)}/\epsilon^2$.*

We therefore obtain an $\mathcal{NC}$ algorithm for constructing a polynomial size distribution approximately fooling any polynomial set of regular functions. Furthermore, as we have observed, this is the only construction where the approximation is in the form of a *relative* error rather than an absolute one, even for the simple case of independence constraints.

5 Algorithmic applications

In this section, we demonstrate how the techniques of this paper can be used to derandomize algorithms. For illustration, we use the example of finding large independent sets in sparse hypergraphs. The problem description and the randomized algorithm for its solution are taken from [ABI86]. The analysis of the problem in terms of constraints is taken from [KM93]. We derandomize the algorithm, thus producing what is, to the best of our knowledge, the first $\mathcal{NC}$ algorithm that completely solves the problem. We then show how our linear system formulation can be used to prove interesting results about the applicability of approximate distributions in derandomization, an important issue given the prevalence of approximate distributions in present derandomization techniques.

A *d-uniform hypergraph* is a pair $\mathcal{H} = (V, \mathcal{E})$ where $V = \{v_1, \dots, v_n\}$ is a set of *vertices* and $\mathcal{E} = \{E_1, \dots, E_m\}$ is a collection of subsets of V, each of cardinality d, that are called *edges*. A subset $U \subseteq V$ is said to be *independent* if it contains no edge. The following algorithm, due to Alon, Babai, and Itai, randomly constructs an independent set in $\mathcal{H}$. First, the algorithm constructs a random subset R of V by putting each v_i in R with probability p. In the second phase, it transforms R into an independent set U as follows: For each edge $E_j \in \mathcal{E}$ such that $E_j \subseteq R$ it removes from R some arbitrary vertex $v_i \in E_j$. Alon, Babai, and Itai prove that this algorithm finds a "large" independent set with "high" probability. Intuitively, the proof is as follows. For each vertex $v_i \in V$, let X_i be the random variable that equals 1 if $v_i \in R$ and 0 otherwise. The cardinality of R is $X = \sum_{i=1}^n X_i$, so we can compute $E(X)$. If the X_i's are pairwise independent, then we can also compute the variance of X, and then use Chebychev's inequality to prove that, with high probability, X is near its expectation. We then prove that in the second phase of the algorithm, not too many vertices are removed from R in forming U. That part of the proof is based on the fact that $\Pr(E_j \subseteq R) = \Pr([X_i = 1 \text{ for all } i \in E_j]) = p^d$, so that not too many edges cause the removal of a vertex. These two steps suffice to show that the resulting set U is large with high probability.

Alon, Babai, and Itai provide an $\mathcal{NC}$ algorithm for this problem by constructing a joint distribution of d-wise independent variables X_i. This technique is effective only for constant d. The results of [BR91] and [MNN89] provide an $\mathcal{NC}$ algorithm for $d = O(\text{poly}\log n)$. However, they still maintain the general approach of searching the sample space of an almost d-wise independent distribution. Hence, this approach cannot be pushed any further in $\mathcal{NC}$. The paradigm of looking only at the precise constraints imposed by the algorithm is crucial for making further progress. In [KM93], Koller and Megiddo observe that d-wise independence, although sufficient for the analysis, is almost entirely redundant. Taking a close look, it is easy to see that far fewer constraints are actually required. As we mentioned, pairwise independence suffices for bounding the variance. Using the constraint-based approach, this requirement is equivalent to fooling the $4\binom{n}{2}$ events of the form $[X_{i_1} = b_1, X_{i_2} = b_2]$. For the argument concerning $\Pr(E_j \subseteq R)$, we are interested only in the probability of the event "all the vertices in E_j are in R". Hence, it suffices to fool the event $[X_i = 1 \text{ for all } i \in E_j]$. That is, we only require a single constraint for each edge, as compared to the 2^d imposed by d-wise independence. Overall, this induces $4\binom{n}{2} + m$ constraints, independently of d.

Our new techniques can improve this still further. Using expectation constraints, rather than enforcing pairwise independence, we simply enforce a constraint saying that the variance and mean of X must be correct. Since it is easy to construct expectation oracles for X and $(X - E(X))^2$, we can reduce the number of constraints needed to $m + 2$. This in turn shrinks the linear systems which we must solve and thus improves the efficiency of the sequential derandomization of [KM93]. We can also apply our parallel techniques to the problem. Using Theorem 4.5, we can construct a distribution with a small support that approximately satisfies these constraints. It is easy to see that the approximation does not affect the correctness of the algorithm. (In fact, most previous derandomizations of this algorithm were also based on approximate distributions.) Thus, we can prove the following result:

Theorem 5.1 : *There is an $\mathcal{NC}$ algorithm that finds independent sets of size $\Omega((n^d/m)^{1/(d-1)})$ in d-uniform hypergraphs with n vertices and m edges, for all values of d, n, m.*

In the sequential derandomization of the hypergraph algorithm, we constructed a distribution that exactly satisfies the desired constraints. In the parallel version, however, we could only construct a distribution that *approximately* satisfies the constraints. This is a typical phenomenon with many of the derandomization techniques, especially the parallel ones. In such a situation, it is necessary to reexamine the correctness proof of the algorithm under consideration and verify that it remains true for approximate distributions. In the case of the hypergraph algorithm, it is easy to verify that approximate distributions are indeed sufficient. However, in general this is work we would like to avoid. As we now show, when the approximate distribution has a small *relative* error, this is often possible. More precisely, a correctness proof for exact distributions immediately implies correctness for approximate distributions with relative error. We consider the following common form of correctness proof. A certain family of "bad" events $\{B_i\}$ is analyzed, and it is proved, based on the probabilities of these bad events, that the algorithm works with probability π. We also need to assume that the bad events B_i do not cover the entire sample space. Note that, even with these restrictions, this result is stronger than the claim that $\Pr(\cup_i B_i)$ changes by only a small factor. It is not necessarily the case that the algorithm succeeds iff no bad event occurs. For example, it may suffice that a majority of the bad events do not happen.

Theorem 5.2: *Consider any algorithm, and suppose that for any distribution over Ω which assigns probability at most b_i to each of several "bad" events B_i, the algorithm fails with probability at most π. Suppose further that Ω contains a good point $g \notin \cup B_i$. Then for any distribution which assigns probability at most $(1+\epsilon)b_i$ to each B_i, the probability that the algorithm fails is at most $(1+\epsilon)\pi$.*

Proof: Consider the sample space Ω. Some of the points in Ω cause the algorithm to fail when they are selected. This subset is not changed by the choice of distribution on Ω. We can define an indicator vector $\vec{c}$ in $\Re^\Omega$ describing these failure points: a particular coordinate has a 1 when the corresponding point causes failure and a 0 when it does not. We also define a matrix A and a vector $\vec{b}$ that correspond to the events B_i and their probabilities (as well as a constraint asserting that $\sum_i x_i \leq 1$). Using the hypotheses of the theorem we observe that whenever $A\vec{x} \leq \vec{b}$, it is the case that $\vec{c}^T\vec{x} \leq \pi$. This can be reformulated as a statement about a linear program: $\max\{\vec{c}^T\vec{x} \mid B\vec{x} \leq \vec{b},\ \vec{x} \geq 0\} \leq \pi$. The dual to this linear program is $\min\{\vec{b}^T\vec{y} \mid B^T\vec{y} \geq \vec{c},\ \vec{y} \geq 0\}$, and we know by duality that it has the same value of at most π. Suppose now that we replace the vector $\vec{b}$ by a vector $\vec{b}'$, where $b'_i \in (1\pm\epsilon)b_i$, and ask for the probability that the algorithm fails based on these changed constraints. This results in new primal and dual systems, with $\vec{b}'$ replacing $\vec{b}$ in both. But now consider the value $\vec{y}$ which minimizes the first dual linear program: it satisfies $B^T\vec{y} \geq \vec{c}$ and $\vec{y} \geq 0$, and hence is a feasible solution to the second dual problem. Since $\vec{b}^T\vec{y} \leq \pi$, it follows that $(\vec{b}')^T\vec{y} \leq (1+\epsilon)\vec{b}^T\vec{y} \leq (1+\epsilon)\pi$, so that the value of the second dual (and also of the second primal) is at most $(1+\epsilon)\pi$. But this is precisely the failure probability of the algorithm under the new, approximated constraints. ∎

We observe that there are alternative ways to prove this theorem. In particular, it is possible to give a proof based on standard probabilistic arguments.[3] However, the linear-system view of derandomization provides a completely different perspective on this problem, one which allows us to utilize general and powerful techniques such as linear programming duality and sensitivity analysis. We hope that these techniques will turn out to be useful in proving other results in this domain.

6 Approximate basis crashing

As we mentioned, the problem of reducing the support of a distribution is closely related to the problem of basis crashing. Unfortunately, the latter problem seems to be inherently sequential, and may well be $\mathcal{P}$-complete (the problem is currently open). In this section, we introduce the problem of *approximate* basis crashing, and present a (de)randomized parallel algo-

[3] We thank Yishay Mansour for pointing this out to us.

rithm for solving it. To the best of our knowledge, this is the first known parallel algorithm for any variant of basis crashing.

Consider an $m \times h$ matrix A that contains only nonnegative coefficients, a vector $\vec{y} \in I\!R^h_+$, and a vector $\vec{b} \in I\!R^m_+$, such that $A\vec{y} = \vec{b}$. Given these inputs, our algorithm constructs another nonnegative solution $\vec{z}$ to the same linear system which is approximate in two ways. First, it is not a true solution to the system, in that it only approximately satisfies the constraints. Given an error parameter δ, it achieves a *relative* error of $O(\delta^{0.5-o(1)}\sqrt{\log m})$ times the correct value. Picking δ to be $1/m^c$, we obtain an arbitrarily small polynomial relative error. Second, our solution is not actually "basic": its support has size $m/\delta + o(m/\delta)$, which makes it small (polynomial size for $\delta = 1/m^c$) but not basic. Our technique extends to basis crashing for packing problems—positive linear programs with an objective function. We use our approximate basis crashing algorithm on an optimal but nonbasic solution. The result will be approximately basic and approximately feasible, and can be made feasible by rounding down in such a way that the objective value remains almost optimal (see Theorem 6.2).

Our approach uses the same idea of random sampling described in Section 4. For each variable in the support, we randomly select whether to eliminate it from the support (by giving it a value of 0) or to keep it in. More precisely, for each column j, we set z_j to be y_j/q_j with probability q_j and 0 otherwise. Each z_j is a random variable whose expected value is y_j. Hence, the expected value of $B_i = \sum_j a_{ij}z_j$ is b_i for each i. However, this does not suffice for our purposes. In order for this construction to be useful, and particularly to derandomize it, we actually need B_i to be close to b_i with high probability. We guarantee this by an appropriate choice of the values q_j. Our goal is to make each q_j as small as possible since this minimizes the expected number of selected indices. However, if q_j is very small, then z_j, which can take a value of y_j/q_j, can become large enough to cause large deviations of B_i from its expected value b_i. This is prevented by ensuring that $a_{ij}y_j/q_j$ does not affect B_i by more than a fraction of δ of its expectation: i.e., $a_{ij}\frac{y_j}{q_j} \leq \delta b_i$ for all i. Hence, we define q_j to be $\max_i \frac{a_{ij}y_j}{\delta b_i}$ if this is at most 1, and 1 otherwise.

For a sampling procedure based on this choice of the q_j's, what is the expected number of z_j's chosen to receive nonzero values? Let the *owner* of y_j be any i for which $\frac{a_{ij}y_j}{\delta b_i}$ is maximal. For each constraint $i = 1, \ldots, m$, we count the expected number of y_j's owned by i that were selected in the construction:

$$\sum_{y_j \text{ owned by } i} q_j \leq \sum_{y_j \text{ owned by } i} \frac{a_{ij}y_j}{\delta b_i} \leq \frac{1}{\delta b_i}\sum_{y_j} a_{ij}y_j = \frac{1}{\delta}.$$

Since every point is owned by some i, the expected number of variables selected is at most m/δ. It is fairly straightforward to verify that the number of variables selected is close to its expectation with high probability. Similarly, by a Chernoff-type bound, since the maximum value of each z_i is only a δ-fraction of the expected value of B_i, the constraints are approximately satisfied with high probability. We omit these proofs ¿from this abstract, since we are actually interested in the derandomized version of this construction.

The derandomization of this construction is based on the *lattice approximation* techniques of Motwani, Naor, and Naor [MNN89]. The lattice approximation algorithm takes as input an $m \times h$ matrix C, with each entry $c_{ij} \in [0,1]$; a vector $\vec{r} \in [0,1]^h$; and a vector $\vec{d} \in I\!R^m$ such that $C\vec{r} = \vec{d}$. It produces as output a vector $\vec{s} \in \{0,1\}^h$ such that $\|C\vec{s} - \vec{d}\|_\infty$ is "small". If we assign $c_{ij} = a_{ij}y_j/q_j$ and $\vec{d} = \vec{b}$, the solution to the corresponding lattice approximation problem is precisely a derandomization of our construction above. The $\mathcal{NC}$ lattice approximation algorithm of [MNN89] allows us to derandomize our construction in $\mathcal{NC}$ (after appropriate transformation and rescaling of C).

Theorem 6.1: *There exists an $\mathcal{NC}$ algorithm such that, given A, $\vec{y}$, and $\vec{b}$ as above, constructs a nonnegative vector $\vec{z}$ such that $(A\vec{z})_i - b_i \in (1 \pm \delta^{0.5-o(1)}\sqrt{\log m})b_i$, and the number of positive entries in $\vec{z}$ is at most $m/\delta + o(m/\delta)$.*

This theorem is the foundation for the parallel support reduction procedure used in Section 4, and hence for our entire parallel construction of small distribution. However, it also has additional applications. Luby and Nisan [LN93] give an $\mathcal{NC}$ algorithm for approximately solving a positive (packing or covering) linear program. By applying our techniques to the solution they find, we can transform it into an approximately basic solution to the problem:

Theorem 6.2: *There exists an $\mathcal{NC}$ algorithm which, given any constant ϵ and an $m \times n$ positive linear program, finds a solution with at most $m\log^{1+o(1)} m$ nonzero coefficients and value within $(1 \pm \epsilon)$ times optimal.*

7 Conclusion

Building on the work of [KM93], we have further explored the problem of constructing small-size distributions by tailoring them to the specific needs of the algorithm and input at hand. We have extended the kinds of constraints which can be satisfied and have also given the first *parallel* algorithm for such a construction. One natural goal here is to further extend the class of constraints which can be maintained. We have already given methods for fooling finite automata; can we extend these techniques to fooling larger language classes such as *logspace* (two way finite automata)? Further investigation of the connection between our results and those of [Nis90, BG92] seems warranted. Another goal is to examine those randomized parallel algorithms which have so far resisted derandomization, and attempt to apply the techniques developed here. Since present proofs of correctness for parallel solutions to maximum matching [KUW86, MVV87] appear to rely on exponentially many constraints, this will of course require a reexamination of the proofs used or a development of new randomized algorithms for the problem.

The approach of Koller and Megiddo utilized basis crashing for reducing the support of the distribution. Motivated by the desire to parallelize their construction, we have developed a new *approximate basis crashing* algorithm. This is a significant contribution of independent interest, since, to our knowledge, there are no parallel algorithms for any type of significant support reduction. On the other hand, Megiddo [Meg94] has observed that the "base case" of basis crashing is as hard as the entire problem. More precisely, he provides an $\mathcal{NC}$-reduction of the general support-reduction problem to the problem of constructing a truly basic solution from one whose support has twice the number of basic variables. "Real" basis crashing is known to have numerous applications in optimization problems such as scheduling and matching. It is quite possible that approximate basis crashing will also be useful in applications other than constructing small distributions.

Finally, this work illustrates yet again the power of viewing probabilistic constraints as linear equations over the probability space. In particular, this paradigm allows us to appeal to powerful techniques such a basis crashing and linear programming duality in constructing small distributions. For example, the use of duality has let us prove a "metatheorem" about the situations in which one can use distributions only *approximately* satisfying the probabilistic constraints used by a proof and still be sure that the proof remains correct. We hope that this different perspective will prove useful in solving other, similar problems.

Acknowledgements

Thanks to Yishay Mansour, Nimrod Megiddo, and Rajeev Motwani for helpful comments and discussions, and to Eyal Kushilevitz for asking whether the approach of [KM93] can be applied to parity constraints.

References

[ABI86] N. Alon, L. Babai, and A. Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *Journal of Algorithms*, 7:567–583, 1986.

[AGHP90] N. Alon, O. Goldreich, J. Hastad, and R. Peralta. Simple constructions of almost k-wise independent random variables. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, pages 544–553, 1990.

[AMN92] Y. Azar, R. Motwani, and J. Naor. Approximating arbitrary probability distributions using small sample spaces. Unpublished manuscript, 1992.

[BG92] Manuel Blum and Oded Goldreich. Towards a computational theory of statistical tests. In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science*, pages 406–416, 1992.

[BM93] P. A. Beling and N. Megiddo. Using fast matrix multiplication to find basic solutions. Technical Report RJ 9234, IBM Research Division, 1993.

[BR91] B. Berger and J. Rompel. Simulating $(\log^c n)$-wise independence in NC. *Journal of the ACM*, 38(4):1026–1046, 1991.

[CGH+85] B. Chor, O. Goldreich, J. Hastad, J. Friedman, S. Rudich, and R. Smolensky. t-resilient functions. In *Proceedings of the 26th Annual Symposium on Foundations of Computer Science*, pages 396–407, 1985.

[EGL+92] G. Even, O. Goldreich, M. Luby, N. Nisan, and B. Veličković. Approximations of general independent distributions. In *Pro-*

ceedings of the 24th Annual ACM Symposium on Theory of Computing, pages 10–16, 1992.

[ES73] P. Erdös and J. Selfridge. On a combinatorial game. *Journal of Combinatorial Theory, series B*, 14:298–301, 1973.

[Jof74] A. Joffe. On a set of almost deterministic k-independent random variables. *Annals of Probability*, 2:161–162, 1974.

[KM93] D. Koller and N. Megiddo. Finding small sample spaces satisfying given constraints. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, pages 268–277, 1993. To appear in SIAM Journal on Discrete Mathematics.

[KM94] Howard Karloff and Yishay Mansour. On construction of k-wise independent random variables. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, 1994. To appear.

[Kus93] E. Kushilevitz, 1993. private communication.

[KUW86] Richard M. Karp, Eli Upfal, and Avi Wigderson. Constructing a perfect matching is in random $\mathcal{NC}$. *Combinatorica*, 6(1):35–48, 1986.

[LN93] M. Luby and N. Nisan. A parallel approximation algorithm for positive linear programming. In *Proceedings of the 25^{th} Annual ACM Symposium on the Theory of Computing*, pages 448–457, 1993.

[Lub86] M. Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM Journal on Computing*, 15(4):1036–1053, 1986.

[Meg94] Nimrod Megiddo, 1994. Private Communication.

[MNN89] R. Motwani, J. Naor, and M. Naor. The probabilistic method yields deterministic parallel algorithms. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 8–13, 1989. To appear in *Journal of Computer and System Sciences. Also available as: IBM Technical Report RJ 7173 (1989).*

[MVV87] Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987.

[Nis90] Noam Nisan. Pseudorandom generators for space-bounded computation. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computating*, pages 204–212, 1990.

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

[Sch92] L. J. Schulman. Sample spaces uniform on neighborhoods. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 17–25, 1992.

[Spe87] J. Spencer. *Ten Lectures on the Probabilistic Method.* Society for Industrial and Applied Mathematics, 1987.