Derandomizing Algorithms on Product Distributions and Other Applications of Order-Based Extraction

Ariel Gabizon*Avinatan Hassidim†

Abstract

Getting the deterministic complexity closer to the best known randomized complexity is an important goal in algorithms and communication protocols. In this work, we investigate the case where instead of one input, the algorithm/protocol is given multiple inputs sampled independently from an *arbitrary unknown* distribution. We show that in this case a strong and generic derandomization result can be obtained by a simple argument.

Our method relies on extracting randomness from "same-source" product distributions, which are distributions generated from multiple independent samples from the same source. The extraction process succeeds even for arbitrarily low min-entropy, and is based on the order of the values and not on the values themselves (this may be seen as a generalization of the classical method of Von-Neumann [26] extended by Elias [7] for extracting randomness from a biased coin.)

The tools developed in the paper are generic, and can be used in several other problems. We present applications to streaming algorithms, and to *implicit probe search* [8]. We also refine our method to handle product distributions, where the i'th sample comes from one of several arbitrary unknown distributions. This requires creating a new set of tools, which may also be of independent interest.

*Department of Computing Science, Simon Fraser University, Vancouver, Canada. ariel.gabizon@gmail.com.
†MIT, Cambridge, Massachusetts. avinatanh@gmail.com

1 Introduction

A central goal in complexity theory is achieving *derandomization* in as many settings as possible. The object of derandomization is to take computational tasks that can be achieved with the aid of randomness, and find ways to perform them using less randomness, or ideally none at all. We want to achieve derandomization without increasing the use of other resources by much. For example, we would like the amount of time, space, communication, etc. used in the deterministic solution to be similar to the corresponding quantities in the original randomized solution.

In this paper we deal with both algorithms for decision problems and communication complexity protocols. In the first case, a long line of work initiated by [24, 4, 28, 21, 13] shows that, assuming certain circuit lower bounds, any randomized polynomial time algorithm can be converted into a deterministic polynomial time algorithm However, proving such lower bounds seems well beyond reach and in fact, Kabanets and Impagliazzo[16] building on Impagliazzo et. al [12] show that proving lower bounds is necessary for proving such results. For communication complexity, there are exponential separations between deterministic and randomized protocols (see [18]).

It thus seems well motivated to look for relaxed (but still interesting) models where derandomization can be achieved. Consider the case of time-bounded algorithms. A first (very naïve) attempt at such a relaxation might be to require that instead of succeeding on *every* input, we succeed with high probability on any distribution of inputs. Of course, this is no relaxation at all, as we can consider distributions concentrated on one hard input. A natural way to further relax this is to require high-probability of success only on distributions of inputs that can be efficiently sampled. Impagliazzo and Wigderson [14], followed by Trevisan and Vadhan[25], give conditional derandomizations (and unconditional Gap Theorems for BPP) in this model. Another type of relaxation, which we investigate here, is to allow arbitrary distributions on individual inputs, but to require multiple independent samples from the same distribution[1]. In this setting, when receiving k inputs for large enough k, we would like our deterministic algorithm to solve all k inputs correctly, at a running time close to k-times the running time of the randomized algorithm. Note that in the case of a distribution concentrated on one hard input, the running time on this input will be amortized over k instances. Similarly, we would like a deterministic communication protocol that when receiving k inputs from an arbitrary distribution (over the inputs of *both* parties) solves all instances correctly with a number of communication bits that is close to k times the number of bits used by the randomized protocol. We show that such results can be achieved by simple argument. We show that our constructions are almost optimal, in some sense. Here is a concrete example, which gives the feel of the parameters.

1.1 A motivating example and result

Consider the equality problem in communication complexity: Alice and Bob receive n-bit strings x and y, respectively. They want to decide whether $x = y$. The deterministic communication complexity is n, and shared randomness reduces this to $O(1)$. Repeating the randomized protocol we get that for any k, $O(\log k)$ communication bits suffice such that Alice and Bob will have the incorrect answer with probability at most $1/100k$.

Consider the setting discussed above: Alice and Bob are now given k-tuples of instances $(x_1, \ldots, x_k)$ and $(y_1, \ldots, y_k)$ respectively, such that each pair (x_i, y_i) is sampled *independently* from the same arbitrary unknown distribution D. Obviously, we have a deterministic protocol that uses $k \cdot n$ communication bits for solving the entire sequence correctly, and a public coin randomized protocol using $O(k \log k)$

[1] We also consider the case of multiple samples that are *not* from the same distribution. Moreover, one might want to consider the case of multiple samples that are correlated in some way, and this might be a direction for further work.

communication bits solving the entire sequence correctly with high probability. We show that when $k > c \cdot n \log n$ for some universal constant c, there exists a deterministic protocol using $O(k \log k)$ communication bits, which solves all instances correctly with probability $2/3$. This result is almost optimal — if $k < n/\log n$ facing the same hard input k times any deterministic protocol must send more than $k \log k$ bits to succeed.

1.2 Main Results

The parameters presented above are derived from the following theorem

Theorem 1.1. *Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be any function. Let P_R be a public coin randomized protocol with error ϵ for f using c_r communication bits and r random bits. Let P_D be a deterministic protocol for f using c_d communication bits. For every integer $k \geq \min\{10 \cdot r \cdot n, 100 \cdot r^2 \cdot (c_d/c_r)\}$, there exists a deterministic protocol P using at most $k \cdot (c_r + \log r + 6)$ communication bits, such that for any distribution D on $\{0,1\}^n \times \{0,1\}^n$,*

$$\Pr_{((x_1,y_1),\ldots,(x_k,y_k)) \leftarrow D^{\oplus k}}(P((x_1,y_1),\ldots,(x_k,y_k)) = (f(x_1,y_1),\ldots,f(x_k,y_k))) \geq 1 - (\epsilon \cdot k + 2^{-r}),$$

where $D^{\oplus k}$ denotes the distribution consisting of k independent copies of D.

We get a similar theorem in the case of algorithms for decision problems.

Theorem 1.2. *Let $\mathcal{C}$ be the class of product distributions on $(\{0,1\}^n)^k$. Let $f : \{0,1\}^n \to \{0,1\}$ be any function. Let A_R be a randomized, two-sided error, algorithm with error ϵ for f, running in time t_r and using r random bits, and let A_D be a deterministic algorithm for f running in time t_d. For every integer $k \geq 10 \cdot (t_d/t_r) \cdot r$, there exists a deterministic algorithm A that runs in time $k \cdot t_r + \tilde{O}(n \cdot k)$, such that for any distribution D on $\{0,1\}^n$,*

$$\Pr_{(x_1,\ldots,x_k) \leftarrow D^{\oplus k}}(A(x_1,\ldots,x_k) = (f(x_1),\ldots,f(x_k)) \geq 1 - (\epsilon \cdot k + 2^{-r}).$$

Remark 1.3.

- *We state Theorem 1.2 for two-sided error randomized algorithms, but it is easy to see in our proofs that if the original randomized algorithm has one-sided error, so will the resultant deterministic algorithm for multiple inputs. Similarly, if we start with a Las-Vegas randomized algorithm, we get a deterministic algorithm for multiple inputs that either answers correctly on all inputs or answers 'I don't know'. (The same holds when one-sided error or Las-Vegas randomized protocols are used in Theorem 1.1)*

- *The reader may wonder whether Theorem 1.2 is interesting as incase the original deterministic algorithm A_D is exponential, we will require an exponential number of independent inputs to use the theorem, and thus still need exponential time. We note again that nothing better is possible in this model (unless a* worst case *derandomization is achieved). Also one gets more interesting instantations in the case where A_D's running time is a larger polynomial than A_R (this is the case in the currently known algorithms for primiality testing) , and in cases where A_D is polynomial or linear and A_R is* sublinear *- as is the case in many property testing algorithms. That said, we agree that the communication complexity setting of Theorem 1.1 is probably more convincing.*

We also consider the case where the inputs or sampled *several* arbitrary distributions. To formally present our results, we need the following definition.

Definition 1.4. *Let $D_1,\ldots,D_d$ be any distributions on $(\{0,1\}^n \times \{0,1\}^n)$. A d-*part product distribution *(defined using $D_1,\ldots,D_d$) on $(\{0,1\}^n \times \{0,1\}^n)^k$, is a distribution $X = (X_1,\ldots,X_k)$ such that the X_i's are all independent, and for each $1 \leq i \leq k$, X_i is distributed according to D_j, for some $1 \leq j \leq d$.*

Our main theorem for d-part product distributions is as follows.

Theorem 1.5. *Fix any positive integers d, n and k and let $\mathcal{C}$ be the class of d-part product distributions on $(\{0,1\}^n)^k$. Let $f : \{0,1\}^n \to \{0,1\}$ be any function. Let A_R and A_D be algorithms for f, similarly to Theorem 1.2. For any $0 < \gamma < 1$ and any $k \geq (t_d/t_r) \cdot ((r \cdot 8(d+1)) + \frac{16 \cdot (2d)^5}{\gamma})$, there exists a deterministic algorithm A that runs in time at most $k \cdot t_r + O(n \cdot k \cdot d^2)$ that solves f on $\mathcal{C}$ with error $\epsilon \cdot k + \gamma$.*

The equivalent theorem for communication protocols can be found in Appendix A.

1.3 Overview of technique — using 'content independent' extraction

We sketch the proof of Theorem 1.2. The proof of Theorem 1.1 is similar but requires additional technical details. We are given a sequence of inputs $(x_1, \ldots, x_k)$ and we want to deterministically compute $f(x_1), \ldots, f(x_k)$ very efficiently. We distinguish between two cases. In the first there are 'few' distinct inputs among $x_1, \ldots, x_k$. In this case we simply run the deterministic algorithm A_D on all these inputs and as there are few of them, it will not take too long.[2] In the second case, we have 'many'[3] distinct inputs among $x_1, \ldots, x_k$. In this case, we extract a random string from the sequence, and use that random string to run the *randomized* algorithm A_R on each input. Let $\{z_1, \ldots, z_s\}$ be the distinct values among $x_1, \ldots, x_k$. A potential problem with this approach would be that the random string we are using *depends on the values* $z_1, \ldots, z_s$ and thus might be a 'bad' string for some z_i with high probability. This does not occur as our extraction method is essentially independent of the actual values of the inputs. More specifically, the random string we extract is simply a function of the *order* in which (the potentially multiple instances of) $z_1, \ldots, z_s$ appear in the sequence. This may be seen as a generalization of the classical method of Von-Neumann [26] extended by Elias [7] on extracting randomness from a biased coin. (see also the work of Peres[22])

Remark 1.6. *As the inputs in the sequence are independent, a more straightforward approach might have been to apply a (deterministic) multi-source extractor on the inputs. However, multi-source extractors require that each input be sampled from a distribution having a certain min-entropy. Thus, to use a multi-source extractor we would have needed to assume the individual inputs come from such a distribution, and would not get results for arbitrary distributions.*

Generalizing to multiple distributions We now sketch the ideas used to prove Theorem 1.5. As in the above, the problem essentially reduces to extracting randomness from d-part product distributions conditioned on seeing 'many' distinct values. Moreover, the extraction procedure should be independent of the actual values and depend only on their order.
Consider the following simple example: We are given 3 independent samples, such that the first and third are sampled from a distribution D_1 distributed on values $a, b \in \{0,1\}^n$. The second sample comes from a distribution D_2 that gives probability one to a value $c \in \{0,1\}^n$ such that $c \neq a, b$. In our terminology, this is a 2-part product distribution D on $(\{0,1\}^n)^3$. Let us look at D conditioned on seeing 3 distinct values. In this case we have a uniform distribution on the sequences (a, c, b) and (b, c, a) (note that we indeed have a *uniform* distribution on these sequences no matter how D_1 is distributed on a and b). This suggests the following method for extracting one bit: Given $x \in (\{0,1\}^n)^3$, for each pair of indices $i < j \in \{1, \ldots, 3\}$, let $z_{i,j}$ be 1 if $x_i < x_j$ by lexicographical ordering, and 0 otherwise. Now output the sum mod 2 of the $z_{i,j}$'s. Let us call this function the 'all-pairs compare' (APC) function.

[2] In the case of Theorem 1.1 there is an additional complication here of having Alice and Bob conclude what indeed are the distinct input pairs (x_i, y_i) with small communication.

[3] 'many' in this sketch roughly corresponds to the number of random bits used by the randomized algorithm for f.

The APC function has the property that if (x_1, x_2, x_3) are all distinct then any substitution of the order of a pair of elements changes the output value. Note that it is essential that *all* the x_i's are distinct. For example, it is easy to check that for any $a < b$, $APC(a, a, b) = APC(b, a, a)$. Thus to extract many random bits, we need many 'blocks' where all inputs are distinct. This suggests the following extraction scheme for d-part product distributions conditioned on seeing many distinct values: Given a sequence of inputs, delete the values that appear 'too many times' in the sequence. Now divide the (possibly trimmed) sequence into blocks of $d + 1$ inputs each. Count the number of blocks such that all inputs in the block are distinct. If there are at least m such blocks - where m is the number of bits we want to extract - output the APC function on each block. It can be shown that if we start out with enough distinct values (where the exact number is a function of d and m) with high probability we will indeed have m blocks of distinct inputs.

1.4 Related work

Goldreich and Wigderson [10], using an observation of Noam Nisan, attain results similar to ours for the case of the uniform distribution[4]. Their technique uses seeded extractors, and their correctness argument is different (and would not work for product distributions of arbitrary distributions). Barak, Braverman, Chen and Rao [**?**] show that randomized communication protocols require about k-times the communication bits to solve k instances with high probability over product distributions. Together with our result this shows that deterministic and randomized protocols have approximately[5] the same computational power in this setting.

1.5 Applications

Beside our main results, we present two applications of extracting randomness based on the order of elements in a sequence.

1.5.1 Implicit probe search

For domain size m and table size n, *implicit probe search* is the problem of searching for an element $x \in [m]$ in a table T containing n elements from $[m]$ using as few queries as possible to T. Arranging T by the regular ordering of $[m]$ and using binary search we can always use at most $\log n$ queries. Yao [27] showed that when m is allowed to be arbitrarily large as a function of n, $\log n$ queries are necessary. Fiat and Naor [8] showed that when $m = poly(n)$, T can be efficiently arranged such that a constant number of queries suffice[6]. The results of [8] are obtained by reducing this problem to the one of explicitly constructing *rainbows*, which may be viewed as a kind of randomness extraction problem (see Appendix B for details). Using this reduction we extend their results, showing that for any $m \leq 2^n$, $O(\frac{\log n}{\log \log n}) = o(\log n)$ queries suffice. Thus, we show that even when the domain is exponentially large there is a scheme significantly better than binary search.

1.5.2 Streaming algorithms

The data stream model was introduced by Munro et al. [19] (see also the seminal work by Alon, Matias and Szegedy [1]). In this model, an algorithm is presented with a sequence of n elements, and its goal is to estimate a function of it, when it is allowed to pass over the data just once. The algorithm runs in bounded space, usually poly-logarithmic in n. We restrict our attention to algorithms which perform in poly-logarithmic space, and compute a frequency moment of the input. For this problem, it is known that even when the order of the appearance of elements in the stream is chosen in an

[4] In fact, for this case their probability of error is smaller than ours.

[5] with the exception that in our result deterministic algorithms only work for large enough k.

[6] Gabizon and Shaltiel[9] showed that for $m = n^{\text{polylog } n}$ a constant number of queries also suffice, although with today's dispersers [8] could have gotten the same result.

adversarial manner, the algorithm can approximate the p'th moment for $0 \leq p \leq 2$ [15, 1], and that this is not possible for moments $p > 2$, see [3, 6].

A relaxation of the problem assumes that the adversary chooses the values of the elements, but they are presented to the algorithm in a random ordering [5, 11, 2]. For a random ordering of the elements, known bounds only imply that one cannot approximate moments larger than 2.5, although it is believed that the right lower bound is 2, as in the adversarial ordering case. We show a strong derandomization result in this model, which enables concentrating on proving lower bounds for deterministic algorithms[7].

We briefly sketch the proof, showing how any randomized algorithm can be simulated by a deterministic one. Let R be a randomized algorithm which approximates (to within any constant) a moment $p > 2$, with any constant success probability. We present a deterministic algorithm D with the same success probability and approximation ratio, up to an $(1 + n^{-\alpha})$ factor, for a constant $\alpha < 0.25$. To simulate R, D first extracts randomness from the beginning of the stream, using the extractors presented later. If the number of elements required to extract enough randomness is small, it uses this randomness to load a pseudo random generator against space bounded machines, and uses this to simulate the random algorithm on the rest of the input; we prove that with high probability this does not change the quality of the approximation by much. If the randomness requires many elements, the deterministic algorithm simply counts the number of appearances of the first polylog n different elements in the stream; we prove that with high probability this is sufficient to approximate the frequency moments over the entire stream [8]. See details in Appendix C.

2 Preliminaries

For background on communication complexity we refer the reader to [18]. The following definitions will be useful for discussing high probability of success on a sequence on inputs.

Definition 2.1. *Let $\mathcal{C}$ be a class of distributions on $(\{0,1\}^n)^k$. Let $f : \{0,1\}^n \to \{0,1\}$ be any function. We say that a deterministic algorithm A solves f on $\mathcal{C}$ with error ϵ, if for any distribution X in $\mathcal{C}$, when sampling a sequence $(x_1, \ldots, x_k)$ according to X, A answers correctly on all inputs in the sequence with probability at least $1 - \epsilon$. That is,*

$$\Pr_{(x_1,\ldots,x_k)\leftarrow X}(A(x_1, \ldots, x_k) = (f(x_1), \ldots, f(x_k))) \geq 1 - \epsilon.$$

An equivalent definition holds for communication protocols; see Appendix A for details. We define extractors for families of distributions. Note that we are talking only about *deterministic* extractors.

Definition 2.2. *Let $\mathcal{C}$ be a class of distributions on a set Ω. A function $E : \Omega \to \{0,1\}^m$ is an* extractor for $\mathcal{C}$ with error γ *(also called a γ-extractor for $\mathcal{C}$), if for every distribution X in $\mathcal{C}$, $E(X)$ is γ-close to uniform.*

3 The main result

A *product distribution* consists of multiple independent samples from an *arbitrary* distribution.

[7] Approximating frequency moments is perhaps the most common studied problem in this model. Our results apply to other problems as well. For adversarial order there are separations between randomized and deterministic algorithms

[8] We note that it is impossible to use PRG's and exhaust over the seeds, as the stream appears just once (in the adversarial order model deterministic algorithm are provably weaker than random ones). Also, the coins used by the random algorithm should be uncorrelated with the ordering of the elements; this is the reason for running it only on part of the stream. Getting a strong result requires some fine tuning of the parameters.

Definition 3.1 (Product Distributions)**.** *A distribution $X = (X_1, \ldots, X_k)$ on $(\{0,1\}^n)^k$ is a* product distribution *if it consists of k independent samples from the same distribution D, where D can be any distribution over $\{0,1\}^n$.*

Our method relies on the fact that product distributions can be written as convex combinations of distributions that just permute a fixed set of values. We now define these distributions.

Definition 3.2 (Multinomial distributions)**.** *The class of* multinomial distributions *on $(\{0,1\}^n)^k$ consists of all distributions of the following form:*
*Let $z_1, \ldots, z_s \in \{0,1\}^n$ be distinct strings and let $a_1, \ldots, a_s$ be non-zero positive integers such that $\sum_{i=1}^{s} a_i = k$. The multinomial distribution X on $(\{0,1\}^n)^k$ defined by $z_1, \ldots, z_s, a_1, \ldots, a_s$, is the uniform distribution on sequences of n-bit strings of length k such that for $1 \leq i \leq k$, the string z_i appears a_i times in the sequence. Moreover, we call such a distribution X an s-*valued *multinomial distribution.*

Lemma 3.3. *Any product distribution is a convex combination of multinomial distributions.*

Proof. Let $X = (X_1, \ldots, X_k)$ be a product distribution on $(\{0,1\}^n)^k$. For any distinct $z_1, \ldots, z_s \in \{0,1\}^n$ and positive integers $a_1, \ldots, a_s$ such that $\sum_{i=1}^{s} a_i = k$. Condition X on the event that the distinct strings outputted were $z_1, \ldots, z_s$ and z_i appears a_i times. Given this conditioning X, because of the independence of $X_1, \ldots, X_k$, any sequence where z_i appears a_i times has equal probability, and therefore we get a multinomial distribution. Writing X as a convex combination of such conditional distributions, the lemma follows. □

For positive integers $k, a_1, \ldots, a_s$ such that $\sum_{i=1}^{s} a_i = k$, the multinomial coefficient $\binom{k}{a_1,\ldots,a_s}$ is the number of different sequences of length k consisting of s distinct elements such that the i'th element appears a_i times: $\binom{k}{a_1,\ldots,a_s} = \frac{k!}{a_1! \cdots a_s!}$. We use the following estimate:

Lemma 3.4. *For any integers $s \leq k$ with $k \geq 32$ and $s \geq 4$ we have $\log \binom{k}{a_1,\ldots,a_s} \geq \frac{s \cdot \log k}{4}$.*

The following claim will enable us to convert uniform distributions over arbitrary sized sets into distributions over binary strings that are close to uniform. A proof can be found in [17].

Claim 3.5. *Let $N > M$ be any integers. Suppose that R is uniformly distributed over $\{1, \ldots, N\}$. Then $R \bmod M$ is $\frac{1}{\lfloor N/M \rfloor}$-close to uniform on $\{0, \ldots, M-1\}$.*

Lemma 3.6. *Fix integers $s \leq k$ with $k \geq 32$ and $s \geq 4$, and let $t = \lfloor \frac{s \cdot \log k}{8} \rfloor$. There exists an extractor $E : (\{0,1\}^n)^k \to \{0,1\}^t$ for the class of s-valued multinomial distributions with error $\gamma = 2^{-t}$. E is computable in time $\tilde{O}(k \cdot n)$.*

Proof. Given a sequence $(x_1, \ldots, x_k)$, let $z_1 < z_2 \ldots < z_s$ be the distinct elements that appear in the sequence, where $<$ denotes the lexicographical ordering. For $i = 1, \ldots, s$ denote by a_i the number of times z_i appears in the sequence. Let S be the set of sequences of length k over $\{1, \ldots, s\}$ such that i appears a_i times. Then $|S| = \binom{k}{a_1,\ldots,a_s}$. The work of Ryabko and Matchikina[23] gives a correspondence of S with $\{1, \ldots, \binom{k}{a_1,\ldots,a_s}\}$ computable in time $\tilde{O}(k \cdot n)$. [9] Let r be the image of the sequence $(x_1, \ldots, x_k)$ in $\{1, \ldots, \binom{k}{a_1,\ldots,a_s}\}$ through this correspondence. Define $E(x_1, \ldots, x_k) \triangleq r \ (mod\ 2^t)$. For any s-valued multinomial distribution X, r is uniformly distributed. Thus using Claim 3.5, $E(X)$ will be γ-close to uniform for $\gamma \leq \frac{1}{\lfloor \binom{k}{a_1,\ldots,a_s}/2^t \rfloor} \leq \frac{1}{2^t}$, where we used the definition of t and Claim 3.4 in the second inequality. □

[9] [23] do this for the case s=2, but it is easy to reduce to this case.

A basic principle in this work, is that when we restrict our input distribution to a component that only 'reorders' a fixed set of values, we can use randomness extracted from the input to run our algorithm or protocol. The following definition and two lemmata formalize this.

Definition 3.7. *We say a distribution X on $(\{0,1\}^n)^k$ is* same-valued*, if there is a fixed set of values $\{z_1,\ldots,z_s\} \subseteq \{0,1\}^n$, such that the support of X consists of sequences $x_1,\ldots,x_k$ such that the set of distinct values in each sequence is exactly $\{z_1,\ldots,z_s\}$.*

Lemma 3.8. *Let $\mathcal{C}$ be a class of same-valued distributions on $(\{0,1\}^n)^k$. Let $E : (\{0,1\}^n)^k \to \{0,1\}^m$ be a γ-extractor for $\mathcal{C}$ computable in time t_E. Fix any $f : \{0,1\}^n \to \{0,1\}$ and any $\epsilon > 0$, and let A_R be a randomized algorithm computing f with error ϵ running in time t_r. Then, there exists a deterministic algorithm A running in time $k \cdot t_r + t_E$ that solves f on $\mathcal{C}$ with error $\epsilon \cdot k + \gamma$.*

Proof. Given $x_1,\ldots,x_k \in (\{0,1\}^n)^k$, A computes $r = E(x_1,\ldots,x_k)$ and outputs $A_R(x_i, r)$ for every $i \in [k]$. The probability that a uniformly chosen $r \in \{0,1\}^m$ is bad for some x_i is at most $\epsilon \cdot k$. Thus, the probability that $r = E(X)$ is bad for some x_i is at most $\epsilon \cdot k + \gamma$. □

We are now ready to prove the theorems stated in the introduction.

Proof of Theorem 1.2. Given k instances $x_1,\ldots,x_k$, let $z_1, z_2 \ldots, z_s$ be the distinct elements that appear in $x_1,\ldots,x_k$. The algorithm A works as follows. Denote $t = \lfloor \frac{s \cdot \log k}{8} \rfloor$. If $t \leq r$ or $s \leq 4$, we run A_D on z_i for every $i \in [s]$. This takes time $t_d \cdot s \leq t_d \cdot 10 \cdot r \leq k \cdot t_r$, which satisfies the theorem for the chosen value of k. Thus we can assume $t \geq r$ and $s \geq 4$. In this case, we compute $y = E(x_1,\ldots,x_k)$, where E is the extractor for multinomial distributions from Lemma 3.6.[10] We now apply the randomized algorithm A_R on all inputs using y as randomness. Let X be a product distribution on $(\{0,1\}^n)^k$. By Lemma 3.3, X is a convex combination of multinomial distributions. Thus, it is enough to prove the theorem in the case that X itself is a multinomial distribution. But in this case, as a multinomial distribution is same-valued the claim follows immediately from Lemma 3.8. As the running time of E is $\tilde{O}(n \cdot k)$, the total running time of A is at most $k \cdot t_r + \tilde{O}(n \cdot k)$. □

For communication protocols the proof requires additional details, as Alice and Bob need to communicate to find out what are the distinct input pairs (x, y). The proof appears in Appendix A.

3.1 On the optimality of our scheme

Using the notation of Theorem 1.2, our method works given at least $k = O((t_d/t_r) \cdot r)$ samples. Can we get a similar result for smaller k? It is easy to see that to get running time $k \cdot t_r$ we need $k \geq t_d/t_r$. In Appendix E we prove a stronger lower bound for a restricted type of scheme. We also show that our extraction scheme is almost optimal.

4 Handling multiple distributions

In this section we show that a similar derandomization for can be achieved when the sequence of inputs is sampled independently from *several* distributions. It is convenient to view d-part product distributions (see Definition 1.4) as convex combinations of certain same-valued distributions.

[10] Note that using $t_d, r > 1$ (otherwise the claim is trivial), we get $k \geq 10 \cdot t_d \cdot r \geq 40$, and thus can use Lemma 3.6.

Definition 4.1. *A d-multinomial distribution on $(\{0,1\}^n)^k$ is a distribution $X = (X_1, \ldots, X_k)$ such that there is a partition $C_1 \cup \ldots \cup C_d = [k]$ into disjoint subsets such that for every $i \neq j \in [d]$, $X|_{C_i}$ and $X|_{C_j}$ are independent, and for every $i \in [d]$ X_{C_i} is a multinomial source. It will be convenient to allow some of the C_i's to be empty. Thus, every d'-multinomial distribution for some $1 \leq d' \leq d$ is also a d-multinomial distribution.*

For distinct strings $z_1, \ldots, z_s \subseteq \{0,1\}^n$ and positive integers $a_1, \ldots, a_s$ such that $\sum_{i=1}^{s} a_i = k$ denote by $D^d_{z_1,\ldots,z_s,a_1,\ldots,a_s}$ the set of d-multinomial distributions whose support consists of sequences where z_i appears a_i times.

Lemma 4.2. *A d-part product distribution is a convex combination of d-multinomial sources.*

4.1 The all pairs extractor

In the following definition, for strings $x, y \in \{0,1\}^n$ we denote by $(x < y)$ the value 1 if $x < y$ (by lexicographical ordering of strings) and 0 otherwise. For an integer l, define the *l-string-all-pairs compare function* $APC : (\{0,1\}^n)^l \to \{0,1\}$ by

$$APC(x_1, \ldots, x_l) \triangleq \bigoplus_{1 \leq i < j \leq l} (x_i < x_j),$$

where $\oplus$ denotes addition modulo 2. That is, we take the parity of comparisons between all pairs.

Claim 4.3. *Fix integers l and n. The l-string all-pairs compare function $APC : (\{0,1\}^n)^l \to \{0,1\}$ is an extractor with error $\epsilon = 0$ for the subclass $D^d_{z_1,\ldots,z_l,1,\ldots,1}$ of d-multinomial distributions for any $d < l$.*

Proof. We first prove the following claim. Let $x = (x_1, \ldots, x_l) \in (\{0,1\}^n)^l$ be a sequence such that $x_i \neq x_j$ for all $i < j \in [l]$. Denote by $x_{i \leftrightarrow j}$ the sequence obtained from x by swapping x_i and x_j. We show that for all $i < j \in [l]$, $APC(x) \neq APC(x_{i \leftrightarrow j})$: To see this[11] notice that swapping adjacent values changes the value of E. That is, for every $1 \leq i \leq l-1$, $APC(x) \neq APC(x_{i \leftrightarrow i+1})$. Loosely speaking, this is because one comparison has changed and the rest have stayed the same. Formally,

$$APC(x_{i \leftrightarrow i+1}) = APC(x) \oplus (x_i < x_{i+1}) \oplus (x_{i+1} < x_i) = APC(x) \oplus 1.$$

Now note that $x_{i \leftrightarrow j}$ can obtained from x by an odd number of swap operations performed on adjacent places: $j - (i+1)$ swap operations to move x_i to the $(j-1)$'th position and another $j - i$ operations to move x_j to the i'th position. Thus we have shown that $APC(x) \neq APC(x_{i \leftrightarrow j})$ for all $i < j \in [l]$. Returning to the original claim, let X be a distribution in $D^d_{z_1,\ldots,z_l,1,\ldots,1}$. Recall that this means there are disjoint subsets $C_1 \cup \ldots \cup C_d = [l]$ such that $X|_{C_i}$ is a multinomial distribution. As $d < l$ there must be an i such that $|C_i| > 1$. Assume w.l.g. that $|C_1| > 1$, and fix two indices $i < j \in C_1$. Look at the distribution X conditioned on a fixing of values in all positions except i and j. Under such a conditioning, we are left we two distinct strings z and z' that are to be assigned in these positions, and as $X|_{C_1}$ is a multinomial distribution, each of the two possible assignments has probability half. From our previous argument it follows that the different assignments will lead to different values of E. Thus, under any such conditioning $APC(X)$ is uniform. Viewing X as a convex combination of such conditional distributions finishes the proof. □

[11] Another way to see this is that if $x_1, \ldots, x_l$ are distinct, the APC function just corresponds to the sign of the permutation which sorts the values. Swapping two elements changes the sign.

4.2 Reducing to all pairs

It will be useful to talk about d-multinomial distributions where 'no value appears too frequently'. The following definition formalizes such a notion.

Definition 4.4. *Let X be a d-multinomial distribution on $(\{0,1\}^n)^k$. We say that X is* δ-bounded *if X belongs to a subclass $D^d_{z_1,\ldots,z_s,a_1,\ldots,a_s}$ of d-multinomial distributions such that for every $i \in [s]$ $a_i \leq \delta \cdot k$. That is, no value z_i appears in more than a δ-fraction of the indices.*

The following lemma shows that a general d-multinomial distribution can be converted into a δ-bounded one, provided it has enough distinct values.

Lemma 4.5. *Fix any $0 < \delta < 1$ and integers n and k. There is a deterministic algorithm F such that for any s-valued d-multinomial distribution X on $(\{0,1\}^n)^k$ with $s \geq (1/\delta) \cdot \log k$, the distribution $F(X)$ is a convex combination of s'-valued δ-bounded d-multinomial distributions on $(\{0,1\}^n)^{k'}$, for some $s' \geq s - (1/\delta) \cdot \log k$ and $k' \leq k$.*

Proof. Given $x = (x_1, \ldots, x_k)$, F operates as follows:

1. Check if there exists a value $z \in \{0,1\}^n$ such that $x_i = z$ for more than a δ-fraction of the x_i's. If so, let z be the most common value in the sequence and remove all x_i's with $x_i = z$.
2. If sequence was changed and it is non-empty, repeat first step on the newly obtained sequence.

Each application of the first step on a d-multinomial distribution, results in a convex combination of d-multinomial distributions. After m repetitions of the first step we are left with at most $(1-\delta)^m \cdot k < e^{-\delta \cdot m} \cdot k$ strings. Thus, after $(1/\delta) \cdot \log k$ repetitions we are left with an empty sequence, and therefore the number of repetitions is bounded by $(1/\delta) \cdot \log k$. Since each repetition reduces the number of values by one, the final components are s'-valued for some $s' \geq s - (1/\delta) \cdot \log k$, as required. □

Theorem 4.6 shows how to extract many random bits from δ-bounded d-multinomial distributions.

Theorem 4.6. *Fix any integers n, m, d and k such that $(d+1)|k$ and $m \leq \frac{k}{d+1}$. The exractor $E : (\{0,1\}^n)^k \to \{0,1\}^m$ be defined as follows. Given input $x \in (\{0,1\}^n)^k$, first partition x into $\frac{k}{d+1}$ blocks $x^1, \ldots, x^{\frac{k}{d+1}}$, each containing $(d+1)$ n-bit strings. We say a block is* good *if all $(d+1)$ n-bit strings in the block are* distinct*. If there are at least m good blocks $x^{i_1}, \ldots, x^{i_m}$, E outputs the all-pairs compare function on each one, $E(x) = APC(x^{i_1}), \ldots, APC(x^{i_m})$. Otherwise, E outputs the 0-string.*

Fix any $0 < \gamma < 1$ and let $\delta = \frac{\gamma}{16d^5}$. E is a γ-extractor for the class of s-valued δ-bounded d-multinomial distributions on $(\{0,1\}^n)^k$, whenever $s \geq m \cdot 4(d+1)$.

The theorem will follow easily from the following lemma.

Lemma 4.7. *Fix any integers n, s, d and k such that $(d+1)|k$. Fix any $0 < \gamma < 1$ and let $\delta = \frac{\gamma}{8 \cdot (d+1) \cdot d^4}$. Let X be an s-valued δ-bounded d-multinomial distribution on $(\{0,1\}^n)^k$. Partitioning a string $x \in (\{0,1\}^n)^k$ and defining a good block as in Theorem 4.6, we have*

$$\Pr_{x \leftarrow X}\left(x \text{ has less than } \frac{s}{4 \cdot (d+1)} \text{ good blocks}\right) \leq \gamma.$$

Proof. Let $C_1 \cup \ldots \cup C_d = [k]$ be the subsets defining X. That is, $X|_{C_i}$ is some multinomial distribution. We show that after removing frequent values as in Lemma 4.5, each one of the underlying distributions

$X|_{C_i}$ is either rare or bounded. Note that if for some $0 < \eta < 1$ and $i \in [d]$ $X|_{C_i}$ is not η-bounded, then $|C_i| \leq \frac{\delta}{\eta} \cdot k$. Taking $\eta = 2\delta \cdot d \cdot (d+1)^2$ we get that at most $(\frac{\delta}{2\delta \cdot d \cdot (d+1)^2} \cdot d) \cdot k = \frac{1}{2(d+1)^2} \cdot k$ indices belong to sets C_i such that $X|_{C_i}$ is not η-bounded. Thus, at most $(d+1) \cdot (\frac{1}{2(d+1)^2} \cdot k) = \frac{k}{2(d+1)}$ blocks contain an index $j \in [k]$ belonging to a subset C_i where $X|_{C_i}$ is not η-bounded. Therefore, we have at least $\frac{k}{d+1} - \frac{k}{2(d+1)} = \frac{k}{2(d+1)}$ blocks such that all indices in the block belong to a set C_i such that $X|_{C_i}$ is η-bounded. Assume without loss of generality that the first $\frac{k}{2(d+1)}$ blocks have this property. For each $i = 1, \ldots, \frac{k}{2(d+1)}$ define a random variable Z_i by $Z_i = 1$ if X^i is bad, and 0 otherwise.

Note that $E(Z_i) = \Pr(Z_i = 1) \leq \frac{(d+1) \cdot d}{2} \cdot \eta = \delta \cdot (d^2)(d+1)^3$. Define $Z = \sum_{i=1}^{\frac{k}{2 \cdot (d+1)}} Z_i$. Then,

$$E(Z) \leq \delta \cdot (d^2)(d+1)^3 \cdot \frac{k}{2 \cdot (d+1)} = \frac{\delta \cdot d^2 \cdot (d+1)^2}{2} \cdot k \leq \delta \cdot 2d^4 \cdot k.$$

Therefore, using Markov's inequality, for any $0 < \gamma < 1$, $\Pr(Z > \frac{1}{\gamma} \cdot (\delta \cdot 2d^4 \cdot k)) \leq \gamma$. Conversely, with probability at least $1 - \gamma$, we have at least $\frac{k}{2 \cdot (d+1)} - \frac{\delta}{\gamma} \cdot 2d^4 \cdot k$ good blocks. Finally, noting that $k \geq s$ and using the value of δ we get

$$\frac{k}{2 \cdot (d+1)} - \frac{\delta}{\gamma} \cdot 2d^4 \cdot k \geq \frac{s}{2 \cdot (d+1)} - \frac{\delta}{\gamma} \cdot 2d^4 \cdot s \geq \frac{s}{4 \cdot (d+1)},$$

and the lemma follows. □

proof of Theorem 4.6. Let X be a δ-bounded s-valued d-multinomial distribution on $(\{0,1\}^n)^k$. Let $l = \frac{k}{d+1}$. For subsets $Z_1, \ldots, Z_l \subseteq \{0,1\}^n$, with $|Z_i| \leq d+1$, we define the distribution $X_{Z_1,\ldots,Z_l}$ to be X conditioned on the event that for every $1 \leq i \leq l$, the set of distinct values in X^i is exactly Z_i. We can view X as a convex combination of the distributions $X_{Z_1,\ldots,Z_l}$. Note that these distributions are simply concatenations of independent d-multinomial distributions on $(\{0,1\}^n)^{d+1}$. Call a distribution $X_{Z_1,\ldots,Z_l}$ 'good' if for at least m values $i \in [l]$, $|Z_i| = d+1$, i.e., the i'th block contains $d+1$ distinct elements. It follows from Lemma 4.7 that the mass of 'good' distributions in the convex combination representing X, is at least $1 - \gamma$. using Claim 4.3, for a good distribution $X_{Z_1,\ldots,Z_l}$, $E(X_{Z_1,\ldots,Z_l})$ is completely uniform. Thus, $E(X)$ is γ-close to uniform. □

Using our conversion from general d-multinomial distributions to δ-bounded d-multinomial distributions, we get an extractor for general d-multinomial distributions.

Corollary 4.8 (Extractors for d-multinomial distributions)**.** *Fix any integers n, m, d and k and any $0 < \gamma < 1$. There is a γ-extractor $E : (\{0,1\}^n)^k \to \{0,1\}^m$ for the class of s-valued d-multinomial distributions whenever $s \geq m \cdot 8(d+1) + \frac{16 \cdot (2d)^5}{\gamma} \cdot \log k$. E is computable in time $O(nk \cdot d^2)$*

Proof. Let F be the algorithm from Lemma 4.5 for $\delta = \frac{\gamma}{16 \cdot (2d)^5}$ Given $x \in (\{0,1\}^n)^k$, our extractor E works by first applying F on x. We then possibly add at most d n-bit strings to $F(x)$ to make the number of n-bit strings it contains a multiple of $d+1$ (at each step, we add the lexicographically first string that does not yet appear in $F(x)$). We then compute $E'(F(x))$, where E' is the extractor for δ-bounded d-multinomial distributions from Theorem 4.6. Let X be an s-valued d-multinomial distribution. Lemma 4.5 guarantees that $F(X)$ is a convex combination of s'-valued δ-bounded d-multinomial distribution for $s' \geq s - (1/\delta) \cdot \log k \geq m \cdot 8(d+1)$. The possible additions make the components of $F(X)$ δ-bounded $2d$-multinomial distributions. As $s' \geq m \cdot 8(d+1) > m \cdot 4(2d+1)$ it now follows from Theorem 4.6 that $E(X) = E'(F(X))$ is γ-close to uniform. □

Using Corollary 4.8, the proof of Theorem 1.5 is similar to the one of Theorem 1.2.

Acknowledgments

We thank Oded Goldreich, Aram Harrow, Jelani Nelson, Noam Nisan, Krzysztof Onak, Ran Raz, Ronen Shaltiel, Leonard Shculman, Chris Umans and Avi Wigderson for stimulating discussions and helpful comments.

References

[1] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. In *STOC '96: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 20–29, New York, NY, USA, 1996. ACM.

[2] Alexandr Andoni, Andrew McGregor, Krzysztof Onak, and Rina Panigrahy. Better bounds for frequency moments in random-order streams, August 15 http://arxiv.org/abs/0808.2222, 2008.

[3] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *J. Comput. Syst. Sci.*, 68(4):702–732, 2004.

[4] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, 13(4):850–864, November 1984.

[5] A. Chakrabarti, G. Cormode, and A. McGregor. Robust lower bounds for communication and stream computation. In *Proceedings of the fourtieth annual ACM symposium on Theory of computing*, pages 641–650. ACM New York, NY, USA, 2008.

[6] Amit Chakrabarti, Subhash Khot, and Xiaodong Sun. Near-optimal lower bounds on the multi-party communication complexity of set disjointness. In *IEEE Conference on Computational Complexity*, pages 107–117, 2003.

[7] P. Elias. The efficient construction of an unbiased random sequence. *Ann. Math. Statist.*, 43(3):865–870, 1972.

[8] A. Fiat and M.Naor. Implicit O(1) probe search. *SICOMP: SIAM Journal on Computing*, 22, 1993.

[9] A. Gabizon and R. Shaltiel. Increasing the output length of zero-error dispersers. In *RANDOM*, pages 430–443, 2008.

[10] O. Goldreich and A. Wigderson. Derandomization that is rarely wrong from short advice that is typically good. In *RANDOM, see also Lecture Notes in Computer Sceince, 209–223*, 2002.

[11] S. Guha and A. McGregor. Stream order and order statistics: Quantile estimation in random-order streams. *SIAM Journal of Computing*, 2008.

[12] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: Exponential time vs. probabilistic polynomial time. In *Sixteenth Annual IEEE Conference on Computational Complexity*, pages 1–12, 2001.

[13] Russell Impagliazzo and Avi Wigderson. $P = BPP$ if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 220–229, El Paso, Texas, 4–6 May 1997.

[14] Russell Impagliazzo and Avi Wigderson. Randomness vs. time: De-randomization under a uniform assumption. In *39th Annual Symposium on Foundations of Computer Science*. IEEE, 1998.

[15] P. Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *Journal of the ACM (JACM)*, 53(3):307–323, 2006.

[16] V. Kabanets and R. Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *CMPCMPL: Computational Complexity*, 13, 2004.

[17] J. Kamp and D. Zuckerman. Deterministic extractors for bit-fixing sources and exposure-resilient cryptography. *SIAM J. Comput*, 36(5):1231–1247, 2007.

[18] E. Kushilevitz and N. Nisan. *Communication Complexity*. Cambridge University Press New York, 1997.

[19] JI Munro and MS Paterson. Selection and sorting with limited storage. In *Foundations of Computer Science, 1978., 19th Annual Symposium on*, pages 253–258, 1978.

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

[21] N. Nisan and A. Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149–167, October 1994.

[22] Y. Peres. Iterating von neumann's procedure for extracting random bits. *Annal of Statistics*, 20, 1992.

[23] Ryabko and Matchikina. Fast and efficient construction of an unbiased random sequence. *IEEETIT: IEEE Transactions on Information Theory*, 46, 2000.

[24] A. Shamir. On the generation of cryptographically strong pseudorandom sequences. *ACM Trans. on Computer Sys.*, 1(1):38, February 1983.

[25] L. Trevisan and S. Vadhan. Pseudorandomness and average-case complexity via uniform reductions. In *Annual IEEE Conference on Computational Complexity (formerly Annual Conference on Structure in Complexity Theory)*, volume 17, 2002.

[26] J. von Neumann. Various techniques used in connection with random digits. *Applied Math Series*, 12:36–38, 1951.

[27] A. C.-C. Yao. Should tables be sorted? *J. ACM*, 28(3):615–628, 1981.

[28] Andrew C. Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science*, pages 80–91, Chicago, Illinois, 3–5 November 1982. IEEE.

A Communication Complexity

In this appendix we present the proofs for the derandomization results in the communication complexity model. The main difference is that the players need to coordinate their actions. For example, one player may have the same value many times, while the other player has a distribution on many values. In case both distributions do not have many different values we must solve all possible $\binom{s}{2}$ pairs, and thus need to increase k. We begin by defining a good deterministic protocol for multiple instances

Definition A.1. *Let $\mathcal{C}$ be a class of distributions on $(\{0,1\}^n \times \{0,1\}^n)^k$. Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be any function. We say that a deterministic protocol P solves f on $\mathcal{C}$ with error ϵ, if for any distribution Z in $\mathcal{C}$, when sampling a sequence $((x_1, y_1), \ldots, (x_k, y_k))$ according to Z , A answers correctly on all inputs in the sequence with probability at least $1 - \epsilon$. That is,*

$$\Pr_{((x_1,y_1),\ldots,(x_k,y_k))\leftarrow X}(A((x_1, y_1), \ldots, (x_k, y_k)) = (f(x_1, y_1), \ldots, f(x_k, y_k))) \geq 1 - \epsilon.$$

The proof of the following lemma is similar to the one of 3.8

Lemma A.2. *Let $\mathcal{C}'$ be a class of distributions on $(\{0,1\}^n \times \{0,1\}^n)^k$. Let $\mathcal{C}$ be a class of same-valued distributions on $(\{0,1\}^n)^k$ such that for any $Z = ((X_1, Y_1), \ldots, (X_k, Y_k))$ in $\mathcal{C}'$, $X = (X_1, \ldots, X_k)$ is in $\mathcal{C}$. Let $E : (\{0,1\}^n)^k \to \{0,1\}^m$ be a γ-extractor for $\mathcal{C}$. Fix any $f : (\{0,1\}^n \times \{0,1\}^n) \to \{0,1\}$ and any $\epsilon > 0$, and let P_R be a randomized public coin protocol for f using r random bits and c_r communication bits. Then, there exists a deterministic communication protocol P using $k \cdot c_r + r$ communication bits that solves f on $\mathcal{C}'$ with error $\epsilon \cdot k + \gamma$.*

Using this Lemma, we can prove the main derandomization result for the communication complexity model.

Proof of Theorem 1.1. Let $z_1, z_2 \ldots, z_s$ be the distinct elements that appear in the sequence $(x_1, \ldots, x_k)$ received by Alice, and let $v_1, \ldots, v_l$ be the distinct elements that appear in the sequence $(y_1, \ldots, y_k)$ received by Bob. The protocol P works as follows:

1. Denote $t_1 = \lfloor \frac{s \cdot \log k}{8} \rfloor$, and $t_2 = \lfloor \frac{l \cdot \log k}{8} \rfloor$. Alice checks whether $t_1 \leq r$ or $s \leq 4$ and Bob checks whether $t_2 \leq r$ or $v \leq 4$, and they communicate the results of these checks.
2. If both answers where positive,
   (a) Bob sends Alice $k \cdot \log l \leq k \cdot (\log r + 4)$ bits indicating for each $1 \leq i \leq k$, for which $j \in [l]$ $x_i = v_j$.
   (b) They run the more efficient of the following two protocols (according to the values r, k, c_d and using (only) that $s, l \leq 10 \cdot r$).
      - Bob sends Alice $v_1, \ldots, v_l$ and Alice computes $f(x_i, y_i)$ for all $i \in [k]$.
      - For $1 \leq i \leq k$, Alice checks whether the instance (x_i, y_i) has appeared previously in the sequence and indicates to Bob whether to solve this instance using P_D or move to the next one (note that Alice can do this using the information sent in Step 2a).
3. Otherwise, assume w.l.g. that $t_1 \geq r$ and $s \geq 4$. In this case Alice computes $y = E(x_1, \ldots, x_k)$ where E is the extractor from Lemma 3.6, and sends y to Bob. Alice and Bob use y as a shared random string to run P_R on all instances.

Let $Z = ((X_1, Y_1), \ldots, (X_k, Y_k))$ be a product distribution on $(\{0,1\}^{n \cdot 2})^k$. By Lemma 3.3, Z is a convex combination of multinomial distributions. Thus, it is enough to prove the theorem in the case that Z itself is a multinomial distribution. In case we have run one of the protocols of Step 2, the correctness is obvious. Note that in this case, the number of communication bits used is at most

$$2 + k \cdot (\log r + 4) + \min\{10 \cdot r \cdot n, 100 \cdot r^2 \cdot c_d + k\} \leq k \cdot (\log r + 6 + c_r),$$

using our assumption on k, which satisfies the claim of the theorem. Otherwise we have used step 3. In this case, note that $X = (X_1, \ldots, X_k)$ and $Y = (Y_1, \ldots Y_k)$ are multinomial distributions one of which

is s-valued for s such that $\lfloor \frac{s \cdot \log k}{8} \rfloor \geq r$. As multinomial distributions are same-valued, the correctness now follows from Lemma A.2. In this case, the number of communication bits used is

$$2 + r + k \cdot c_r \leq k \cdot (c_r + \log r + 6),$$

as required. □

Finally, we present the result for communication protocols with multiple distributions. The proof is similar to the one of Theorem 1.5, with the required adjustments of Theorem 1.1.

Theorem A.3. *Fix any integer d, and let $\mathcal{C}$ be the class of d-part product distributions on $(\{0,1\}^{2 \cdot n})^k$. Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be any function. Let*

- *P_R be a public coin randomized protocol with error ϵ for f using c_r communication bits and r random bits*
- *P_D be a deterministic protocol for f running using c_d communication bits.*

Fix any $0 < \gamma < 1$ and denote $\ell = r \cdot 8(d+1) + \frac{16 \cdot (2d)^5}{\gamma}$. For every integer $k \geq \ell \cdot min\{\ell \cdot (c_d/c_r), n\}$, there is a deterministic protocol P using at most $k \cdot (c_r + \log r + 6)$ communication bits that solves f on $\mathcal{C}$ with error $\epsilon \cdot k + \gamma$.

B Rainbows and Implicit Probe Schemes

In this section we discuss an application of same-source extractors to the problem of *implicit probe search*. Loosely speaking, this is the problem of searching for an element in a table with few probes, when no additional information but the elements themselves is stored.

Definition B.1 (Implicit probe search scheme)**.** *Fix integer parameters n, k and q such that $k < n$. The implicit probe search problem is as follows: Store a subset $S \subseteq \{0,1\}^n$ of size 2^k in a table T of size 2^k, (where every table entry holds only a single element of S), such that given $x \in \{0,1\}^n$ we can determine whether $x \in S$ using q queries to T. A solution to this problem is called an* implicit q-probe scheme with table size 2^k and domain size 2^n.
We will say a scheme is efficient *if given $x \in \{0,1\}^n$ (and given access to the table T storing the elements of S), we can determine whether $x \in S$ in poly(n)-time.*

Fiat and Naor [8] investigated implicit $O(1)$-probe schemes, i.e., schemes where the number of queries is a constant not depending on n and k. They showed that this problem is unsolvable when n is large enough relative to k (this improves a previous bound by Yao [27]). They also gave an efficient implicit $O(1)$-probe scheme whenever $k = \delta \cdot n$ for any constant $\delta > 0$. They did this by reducing the problem to the task of constructing a combinatorial object called a *rainbow*.

Definition B.2. *(Taken from [8])*

- *A* t-sequence *over a set U is a sequence of length t without repetitions, of elements in U.*
- *An (m, n, k, t)-rainbow is a coloring of all t-sequences over $\{0,1\}^n$ with 2^m colors such that for any $S \subseteq \{0,1\}^n$ of size 2^k, the t-sequences over S are colored in all colors.*

We will say an (m, n, k, t)-rainbow is explicit *if it is poly(n)-time computable.*[12]

[12] This definition makes sense as the other parameters m, k and t will always be at most n in this section.

For readers familiar with extractor literature, a rainbow may be viewed as a seedless 'zero-error disperser' where our 'weak random source' consists of multiple independent copies of the same source with the additional restriction that the same sample cannot appear twice.

Fiat and Naor showed that rainbows imply implicit probe schemes.

Theorem B.3. *[8] Fix any n, k with $\log n \leq k \leq n$. Given an explicit (k, n, k, t)-rainbow we can construct an efficient implicit $O(t)$-probe scheme with table size 2^k and domain size 2^n.*

We first present a simple rainbow construction for small k with few colors (which is implicit in [8]). Later on, we show how to get to 2^k colors.

Lemma B.4. *For any n, k and t such that $t \geq 9$ and $\log t \leq k \leq n$, there is an explicit $(t \log t/3, n, k, t)$-rainbow.*

Proof. The rainbow will simply output the 'permutation order' of the input. More precisely, given input $x_1, \ldots, x_t$, where the x_i's are distinct, let $\sigma \in S_t$ be the unique permutation of $\{1, \ldots, t\}$ such that $x_{\sigma(1)} < \ldots < x_{\sigma(t)}$ are in lexicographical order. Output the index of σ in S_t. Since, for $t \geq 9$ (using Stirling's approximation)

$$\log |S_t| = \log t! \geq t \log t/3,$$

we are done. □

The following instantiation of Lemma B.4 will be convenient for us.

Corollary B.5. *Fix any constant $c \geq 1$ and let $d = 6 \cdot c$. For every large enough k and n with $k < n$ and $k > \frac{d \cdot \log n}{\log \log n}$, there is an explicit $(c \cdot \log n, n, k, \frac{d \cdot \log n}{\log \log n})$-rainbow.*

Proof. Taking $t = 6c \cdot \frac{\log n}{\log \log n}$ in Lemma B.4, we get an (m, n, k, t)-rainbow where

$$m = t \log t/3 \geq 2c \cdot \frac{\log n}{\log \log n} \cdot (\log \log n - \log \log \log n) \geq c \cdot \log n,$$

for large enough n. □

To increase the number of colors we use the following lemma from [9].

Lemma B.6. *There exist constants $0 < \eta < 1$ and c such that for every sufficiently large k, n and $m = \eta k$: There is a $\mathrm{poly}(n)$-time computable function $F : (\{0,1\}^n)^2 \times \{0,1\}^{d=c \cdot \log n} \to \{0,1\}^m$ such that for any 2 independent sources X_1, X_2 with min-entropy at least k, for every $b \in \{0,1\}^m$, there exists $y \in \{0,1\}^d$ such that*

$$\Pr(F(X_1, X_2, y) = b) \geq 2^{-m+1}.$$

Using this lemma we get the following rainbow construction

Theorem B.7. *For every large enough k and n with $\log n \leq k < n$, there is an explicit $(k, n, k, O(\log n/ \log \log n))$-rainbow*

Proof. We will show that for some constant $0 < \eta < 1$ there is an explicit $(\eta k, n, k, O(\log n/ \log \log n))$-rainbow. It then follows from [8] (see remark after Theorem 6 in that paper) that there is an explicit $(k, n, k, O(\log n/ \log \log n))$ rainbow.

Let F be the function from Lemma B.6. Let D_1 be the $(c \cdot \log n, n, k, t)$-rainbow from Corollary B.5 taking c to be the coefficient of $\log n$ in the seed length of F, where $t = O(\log n / \log\log n)$. We define a function D by

$$D(x_1, \ldots, x_{t+2}) \triangleq F(x_{t+1}, x_{t+2}, D_1(x_1, \ldots, x_t)).$$

Let $S \subseteq \{0,1\}^n$ be a subset of size 2^k and let (X_{t+1}, X_{t+2}) be the distribution consisting of 2 independent copies of the uniform distribution on S. Fix any $b \in \{0,1\}^m$. Fix a y such that

$$\Pr(F(X_{t+1}, X_{t+2}, y) = b) \geq 2^{-m+1}$$

Since $Pr(X_{t+1} = X_{t+2}) = 2^{-k}$ and $2^{-k} < 2^{-m+1}$, assuming say $\eta \leq 1/2$, then there must be *distinct* $x_{t+1}, x_{t+2} \in S$ such that $F(x_{t+1}, x_{t+2}) = b$. Fixing a t-sequence in $x_1, \ldots, x_t$ over S such that $D_1(x_1, \ldots, x_t) = y$, we get

$$D(x_1, \ldots, x_{t+2}) = F(x_{t+1}, x_{t+2}, y) = b.$$

Thus, for any $b \in \{0,1\}^m$ we have found a $(t+2)$-sequence over S mapped to b and the claim follows. □

Corollary B.8. *For every large enough k and n with $\log n \leq k < n$, there exists an efficient implicit $O(\log n / \log\log n)$-probe scheme for table size 2^k and domain size 2^n*

We phrase the result, perhaps more naturally, with query complexity and domain size as a function of table size.

Corollary B.9. *For every large enough m and n with $m \leq 2^n$, there exists an efficient implicit $O(\log\log m / \log\log\log m)$-probe scheme for table size n. In particular, for $m = 2^n$ there is an efficient implicit $O(\log n / \log\log n)$-probe scheme.*

C Data Stream

Let $x_1, \ldots x_n$ be the stream of elements, and assume that we are trying to compute the p'th moment, for a constant $p > 2$. Let R be a randomized algorithm which computes the moment up to multiplicative factor $(1+\epsilon)$ with success probability $1-\delta$, where the probability is taken over the ordering of the elements as well as the coin tosses of the algorithm. Let S be a bound on the space required by R, where S is poly logarithmic in n, and let r be a bound on the number of coins R uses. Assume r is at most polynomial in n (having r quasi polynomial does not change the result). We present a deterministic algorithm D, which approximates the p'th moment[13] up to a multiplicative factor of $(1+\epsilon+\epsilon_D)$ with success probability $1-\delta-\epsilon_D$, with a space bound of $O(\log^2 n \cdot S \cdot \log r)$ where $\epsilon_D = n^{-\alpha}$ for some $\alpha > 0$ (the analysis here requires $\alpha < 0.25$; this can be improved by refining the algorithm.)

C.1 Preliminaries

Let $f_p(\vec{y})$ denote the p'th moment; that is, if the distinct elements of $\vec{y}$ are $y_1, \ldots y_m$ where y_i appears c_i times then $f_p(\vec{y}) = (\sum_i c_i^p)^{1/p}$. The algorithm will use a Pseudo Random Generator (PRG) against bounded space. Nisan showed in [20], that

Theorem 1 from [20] *there exists a constant $c > 0$ such that there exists a deterministic pseudo random generator which converts a random seed of length $cS\log r$ bits to r bits, such that if R is a deterministic algorithm which runs under space S, the probability that R distinguishes the truly random bits from the output of the PRG is $O(1/n)$.*

[13] similar techniques work for other functions such as Reiny entropy, entropy etc.

We ignore the errors induced by this construction, by playing with the constants. The generator itself runs in space $O(S \log r)$.

The common way to use this PRG is to exhaust over all possible random strings, and to simulate the random algorithm on all possible outputs. However, this approach is not possible in the data streaming model, as the input appears just once. Therefore we use the randomness of the ordering, to load the PRG. In general, this may not be possible, as the coin flips of the random algorithm need to be independent of the ordering. To solve this, we later prove a continuity result on frequency moments. Results of similar flavor hold for other problems as well.

C.2 Algorithm

The deterministic algorithm begins by compressing the beginning of the stream. If this builds up entropy quickly, it uses this entropy to load the PRG, and simulate the random algorithm on the rest of the stream. If the entropy rise is slow, it goes over the stream, counting the number of appearances of the elements which appeared in the beginning. Using these numbers, it produces an estimate to the total p'th moment. Formally

1. Let k be the first element such that the multinomial coefficient describing $x_1, \ldots x_k$ is at least $2^{d \cdot \log n}$ where $d = c \cdot S \cdot \log r$, and c is the constant from the PRG. Store $x_1, \ldots, x_k$, by storing the multinomial coefficient, as well as the distinct values.
2. If k is small (at most $n^{3/4}$):
   (a) D applies E from 3.6 on the multinomial, to extract r bits which are $1/n$ close to uniform.
   (b) D uses the r random bits to load Nisan's PRG from [20], and uses it to simulate R on the rest of the stream.
   (c) Output $R(x_{k+1}, \ldots, x_n)$.
3. If $k > n^{3/4}$
   (a) Let s denote the number of distinct values of $x_1, \ldots, x_k$. Assume without loss of generality that the distinct elements are $x_1, \ldots, x_s$.
   (b) D goes over the series, counting the number of appearances of $x_1, \ldots, x_s$. Let a_i denote the number of appearances of x_i.
   (c) Output $(\sum_i a_i^p)^{1/p}$.

C.3 Analysis

We first show that D runs in polylogarithmic space.

Theorem C.1. *D runs in space at most $O(\log^2 n \cdot c \cdot S \cdot \log r)$*

Proof. Remember $d = c \cdot S \cdot \log r$. Computing the the multinomial coefficient of $x_1, \ldots, x_k$ is done using at most $O(d \cdot \log n \log \log n)$ bits. Storing the distinct elements of $x_1, \ldots, x_k$ requires at most $d \log^2 n$ bits, as there are at most $d \cdot \log n$ such elements. If k is small, D proceeds by simulating R; this is done in space $S < d$. If k is large, one needs to compute $a_1, \ldots, a_s$. to bound s, note that if there are s different values than the multinomial coefficient is at least $s! > 2^s$. Therefore, $s < d \log n$, and this can also be done in space $d \log^2 n$, which gives the required bound. □

The correctness proof consists of two different cases.

Theorem C.2. *For any constant $p > 2$ and $\alpha < 0.25$, if $k < n^{3/4}$, then $f_p(x_{k+1}, \ldots, x_n) \leq f_p(x_1, \ldots, x_n) \leq (1 + n^{-\alpha}) f_p(x_{k+1}, \ldots, x_n)$*

Proof. The first inequality is trivial. To show the second inequality, we need to differentiate between elements which appeared many times in $x_1, \ldots, x_k$ (and will therefore probably appear again), and between elements which appeared few times (which will change the approximation ratio only slightly). Assume without loss of generality that $x_1, \ldots, x_l$ appeared more than $2 \log n + 1$ times in $x_1, \ldots, x_k$, and that $x_{l+1}, \ldots, x_s$ appeared less than $2 \log n + 1$ times. Let β_i denote the number of appearances of x_i in $x_1, \ldots, x_k$. We need to prove

$$f_p(x_1, \ldots, x_n) < (1 + n^{-\alpha}) f_p(x_{k+1}, \ldots, x_n)$$

or equivalently $f_p^p(x_1, \ldots, x_n) < (1 + n^{-\alpha})^p f_p^p(x_{k+1}, \ldots, x_n)$. Using the bound $(1 + n^{-\alpha})^p > 1 + pn^{\alpha}$ from the Binomial Theorem, we prove a stronger statement, namely that

$$f_p^p(x_1, \ldots, x_n) - f_p^p(x_{k+1}, \ldots, x_n) < pn^{-\alpha} f_p^p(x_1, \ldots, x_n) \tag{1}$$

We analyze the LHS of the inequality (1).

$$f_p^p(x_1, \ldots, x_n) - f_p^p(x_{k+1}, \ldots, x_n) = \sum_{i=1}^{s} (\gamma_i + \beta_i)^p - \gamma_i^p$$

Where γ_i denotes the number of appearances of the i'th element in $x_{k+1}, \ldots, x_n$.

The following lemma helps us consider heavy elements and light ones differently

Lemma C.3. *If $\beta_i > 2 \log n + 1$ then $\forall \epsilon > 0$ and large enough n, $\Pr(\gamma_i < (1 - \epsilon) n/k \cdot \beta_i) < O(1/n)$*

Note that $(1 - \epsilon) n/k > n^{\alpha} > n^{\alpha/p}$. Applying a union bound over at most s heavy elements, with probability $1 - O(\text{polylog}\, n/n)$

$$\sum_{i;\ \gamma_i > n^{\alpha/p} \beta_i} (\gamma_i + \beta_i)^p - \gamma_i^p < \frac{p}{2} n^{-\alpha} \sum_{i;\ \gamma_i > n^{\alpha/p} \beta_i} \gamma_i^p$$

When β_i is small, assuming that we are in the probable even and γ_i is also small:

$$\sum_{i;\ \gamma_i < n^{\alpha/p} \beta_i} (\gamma_i + \beta_i)^p - \gamma_i^p < \frac{p}{2} n^{1-\alpha}$$

To finish the proof, we look at the RHS of (1):

$$pn^{-\alpha} f_p^p(x_1, \ldots, x_n) \geq pn^{-\alpha} \max n, \sum_i \gamma_i^p \geq \frac{p}{2} (n^{1-\alpha} + \sum_i \gamma_i^p)$$

□

The theorem gives that if k is small D has almost the same success probability as R (up to a $O(\log n/n)$ term which comes from the PRG and from the density of the heavy elements), and outputs an approximation which is almost as good (up to $(1 + n^{-\alpha})$).

If $k > n^{3/4}$, assume without loss of generality that $x_1, \ldots, x_s$ are all the unique elements of $x_1, \ldots, x_k$, and note that $s < d \cdot \log n$. for $1 \leq i \leq s$, Let a_i denote the number of appearances of x_i in $x_1, \ldots, x_n$. D computes $a_1, \ldots, a_s$, and outputs $(\sum_i a_i^p)^{1/p}$.

Theorem C.4. *For any constant $p > 2$ and $\alpha < 0.25$, if $k > n^{3/4}$, then with probability $1 - n^{-\alpha}$ we have $(\sum_i a_i^p)^{1/p} \leq f_p(x_1, \ldots, x_n) \leq (1 + n^{-\alpha})(\sum_i a_i^p)^{1/p}$*

Proof. As in Theorem C.2, the first inequality is trivial, and holds with probability 1. As for the second, denote $x = n - \sum_i a_i$. Given that $x_{s+1}, \ldots, x_k$ contain only the elements of $x_1, \ldots, x_s$, the probability that $x \gg n^{0.25}$ is exponentially small. We only require a weaker bound, for example

$$\Pr(x > \sqrt{n}) < O(1/n)$$

To prove a bound (which is stronger than the one we need), go over $x_1, \ldots x_n$, putting red balls in all the places where an element of $x_1, \ldots, x_s$ appeared, and blue balls elsewhere. Given that the first s balls are red, if there are $x > \sqrt{n}$ blue balls, the probability that none of them appeared in $x_{s+1}, \ldots, x_k$ is

$$(1 - x/n - s + 1)(1 - x/n - s + 2) \ldots (1 - x/n - k) < (1 - x/n - k)^{k-s}$$

and this probability is exponentially small.

Assume that $x < \sqrt{n}$. The pigeonhole principle gives that there are heavy elements among the s first elements. These elements dominate the moment, and the difference between estimates is bounded $1 + n^{-\alpha}$. □

D Proofs of Technical Lemmas

This appendix presents the calculations done in the paper.

Lemma 3.4 *For any integers $s \leq k$ with $k \geq 32$ and $s \geq 4$ we have*

$$\log \binom{k}{a_1, \ldots, a_s} \geq \frac{s \cdot \log k}{4}$$

Proof. It follows from stirling's formula that for any integer k

$$\sqrt{2\pi k}(k/e)^k \leq k! \leq 3\sqrt{2\pi k}(k/e)^k.$$

Thus,

$$\binom{k}{a_1, \ldots, a_s} \geq \frac{k!}{(k-s+1)!} \geq \frac{\sqrt{2\pi k}(k/e)^k}{3 \cdot \sqrt{2\pi(k-s+1)}((k-s+1)/e)^{k-s+1}}$$

$$\geq \frac{(k/e)^k}{3 \cdot (k/e)^{k-s+1}} = (1/3) \cdot (k/e)^{s-1}$$

Thus,

$$\log \binom{k}{a_1, \ldots, a_s} \geq (s-1)\log k - (s-1)\log e - \log 3$$

Using $s \geq 4$

$$\geq (3/4)s \log k - 2.5s \geq (1/4)s \log k$$

The last inequality follows as

$$(s \log k)/2 \geq 2.5s \leftrightarrow \log k \geq 5s \leftrightarrow k \geq 32$$

□

E Lower bounds

How large does k have to be to get a deterministic scheme for product distributions running in time that is almost k-times the randomized running time? We have given an upper bound of $k = O(t_d \cdot r/t_r)$. In the theorem below we give a somewhat weaker lower bound for algorithms of the following form.

Given $(x_1, \ldots, x_k)$ **either**

1. for all i run $A_R(x_i, E(x_1, \ldots, x_k))$ for some function E.
2. for all i run $A_D(x_i)$.

Theorem E.1. *Fix deterministic and randomized algorithms A_D, A_R respectively for a language L. Denote the running times of A_D and A_R by t_d and t_r respectively. Let k and n be any large enough integers. Assume that there is an algorithm A of the above form running in time $k \cdot t_r + f(n,k)$, such that $f(n,k) < k \cdot t_r$, that answers correctly on all instances $x_1, \ldots, x_k \in \{0,1\}^n$ with probability $4/5$. Let r^* be the minimal number of random bits used by an algorithm for L running in time $t_r + f(n,k) + k \cdot \log^2 k$ with error $1/3$. Then ,*

$$k \cdot \log k = \Omega\left(\frac{r^* \cdot t_d}{t_r}\right).$$

Thus, either k is super-polynomial in r^* or

$$k = \Omega\left(\frac{r^* \cdot t_d}{\log r^* \cdot t_r}\right).$$

Proof. Assume we have such an A for $k = \frac{r^* \cdot t_d}{c \cdot \log k \cdot t_r}$ for a large enough constant c to be determined later. We will construct a randomized algorithm A'_R for f with error $1/3$ running in time $t_r + f(n,k) + k \cdot \log^2 k$ using less than r^* random bits, leading to a contradiction. Define a distribution D on $\{0,1\}^n$ as follows: Let $s = 8 \cdot r^*/(c \cdot \log k)$, and fix distinct elements $z_0, \ldots, z_s \in \{0,1\}^n$. D will give z_0 probability $1 - \frac{10 \cdot s}{k}$ and, for $1 \le i \le s$, D gives z_i probability $10/k$. By Chernoff, with probability at least $99/100$, for large enough[14] s we have between $5s$ to $15s$ appearances of the elements $z_1, \ldots, z_s$ in a sequence of k independent samples from D (while the rest of the sequence is equal to z_0). Also, with probability at least $99/100$ there are at least $s/4$ *distinct* elements in the sequence.[15] Denote by X the distribution $D^{\oplus k}$ conditioned on having $5s - 15s$ appearances of $z_1, \ldots, z_s$ and having at least $s/4$ distinct elements. As X has mass at least $98/100$ in $D^{\oplus k}$, A must be correct on sequences from X with probability at least $4/5 - 2/100 > 3/4$. Note also that as sequences from X contain at least $s/4$ distinct elements A must invoke A_R on all of them: Otherwise it would have running time at least

$$s/4 \cdot t_d = 2 \cdot \frac{r^* \cdot t_d}{c \cdot \log k} \ge 2k \cdot t_r > k \cdot t_r + f(n,k).$$

The algorithm A'_R will work as follows: Given input $z = z_0$ it will produce attempt to sample $(x_1, \ldots, x_k)$ from the distribution X(substituting z as z_0) as described below. If it succeeds, then it will run $A_R(z, E(x_1, \ldots, x_k))$ and answer accordingly. We attempt to sample from X as follows: Choose a subset of $[k]$ of size $15 \cdot s$. For each index in the subset, for $1 \le i \le s$, choose z_i with probability $10/k$,

[14] if s is bounded by a constant so is $r^*/\log k$ in which case the theorem is correct by the bound $k \ge t_d/t_r$.

[15] The probability of a certain z_i appearing is $1 - (1 - 10/k)^k \ge 1 - e^{-10} \ge 999/1000$. Define a random variable Y to be the number of distinct elements from $\{z_1, \ldots, z_s\}$ appearing. Then, $E(Y) \ge 999/1000 \cdot s$. Denoting $\delta = \Pr(Y \le s/4)$ we have $999/1000 \cdot s \le \delta \cdot s/4 + (1 - \delta) \cdot s$, which implies $\delta \le 1/100$.

and choose z_0 with probability $1 - 10 \cdot s/k$. Put z_0 in the rest of the indices. As noted before with probability at least 98/100 we will have $s/4$ distinct elements and at least $5s$ appearances of $z_1, \ldots, z_s$. Conditioned on this event we have a uniform element from X.

The sampling process will take at most $(c/16) \cdot s \cdot \log k \leq r^*/2$ random bits for some constant c, and time $k \cdot \log^2 k$. The algorithm A'_r will be correct with probability 3/4 conditioned on succeeding to generate a sample from X, and thus it's total error will be at most $1/4 + 2/100 < 1/3$.

Altogether, we have obtained a randomized algorithm using less than r^* random bits running in time at most $t_r + f(n,k) + k \cdot \log^2 k$ with error at most 1/3, which is a contradiction to the minimality of r^*. □

In the theorem below we show our extraction scheme is almost optimal.

Theorem E.2. *Let $\mathcal{C}$ be the class of product distributions on $(\{0,1\}^n)^k$ conditioned on having at least s distinct values. Then $\mathcal{C}$ contains a distribution X with $H_\infty(X) \leq O(s \cdot \log k)$. Thus any extractor for $\mathcal{C}$ with error $\epsilon \leq 1/2$ can extract at most $O(s \cdot \log k)$ bits.*

Proof. As a first step to construct X, we define a distribution D on $\{0,1\}^n$ as follows: Fix distinct elements $z_0, \ldots, z_{2s} \in \{0,1\}^n$. D will give z_0 probability $1 - \frac{2s}{k}$ and, for $1 \leq i \leq 2s$, D gives z_i probability $1/k$. Let us denote by X the product distribution $D^{\oplus k}$ conditioned on seeing at least s distinct elements. Denote by X' the distribution X conditioned on having between s and $3s$ appearances of $z_1, \ldots, z_{2s}$. As the min-entropy of a distribution is at most the log of its support size, using the bound of Lemma 3.4 , we have $H_\infty(X') \leq O(s \cdot \log k + \log s) = O(s \cdot \log k)$. Note that the $\log s$ term came from having $2s + 1$ options as to how many appearances of $z_1, \ldots, z_{2s}$ we have.

By Chebychev, with probability at least $1 - 2/s$ we have between s to $3s$ appearances of the elements $z_1, \ldots, z_{2s}$ in a sequence of k independent samples from D. Thus, X' has mass at least $1 - 2/s$ in X and therefore $H_\infty(X) = H_\infty(X') + O(\log s) = O(s \cdot \log k)$. □