Universal almost optimal compression and Slepian-Wolf coding in probabilistic polynomial time*

BRUNO BAUWENS, National Research University Higher School of Economics, Russia
MARIUS ZIMAND†, Towson University, USA

In a lossless compression system with target lengths, a compressor $\mathcal{C}$ maps an integer m and a binary string x to an m-bit code p, and if m is sufficiently large, a decompressor $\mathcal{D}$ reconstructs x from p. We call a pair (m, x) *achievable* for $(\mathcal{C}, \mathcal{D})$ if this reconstruction is successful. We introduce the notion of an optimal compressor $\mathcal{C}_{\text{opt}}$, by the following universality property: For any compressor-decompressor pair $(\mathcal{C}, \mathcal{D})$, there exists a decompressor $\mathcal{D}'$ such that if (m, x) is achievable for $(\mathcal{C}, \mathcal{D})$, then $(m + \Delta, x)$ is achievable for $(\mathcal{C}_{\text{opt}}, \mathcal{D}')$, where Δ is some small value called the overhead. We show that there exists an optimal compressor that has only polylogarithmic overhead and works in probabilistic polynomial time. Differently said, for any pair $(\mathcal{C}, \mathcal{D})$, no matter how slow $\mathcal{C}$ is, or even if $\mathcal{C}$ is non-computable, $\mathcal{C}_{opt}$ is a fixed compressor that in polynomial time produces codes almost as short as those of $\mathcal{C}$. The cost is that the corresponding decompressor is slower.

We also show that each such optimal compressor can be used for distributed compression, in which case it can achieve optimal compression rates, as given in the Slepian-Wolf theorem, and even for the Kolmogorov complexity variant of this theorem.

ACM Reference Format:
Bruno Bauwens and Marius Zimand. 2019. Universal almost optimal compression and Slepian-Wolf coding in probabilistic polynomial time. *J. ACM* 1, 1 (November 2019), 33 pages. https://doi.org/0000001.0000001

1 INTRODUCTION

Good lossless compression means mapping data to short compressed codes. Ideally, the codes should have minimal or close-to-minimal description length. It is also desirable to have efficient compression and decompression procedures. Unfortunately, it is impossible to obtain close-to-minimal codes from which decompression is even remotely efficient.[1] In contrast, we show that, remarkably, there is no fundamental conflict between having close-to-minimal codes and efficient compression. More precisely, if a good approximation of the length of a minimal description of the data is given, then a probabilistic compressor can generate an almost minimal compressed code in polynomial time. The same

*Preliminary versions of the results in this paper have been presented at the IEEE 29th Conference in Computational Complexity (CCC), Vancouver, June 2014 [Bauwens & Zimand 2014] and at the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, June, 2017 [Zimand 2017]

†Marius Zimand has been supported in part by the National Science Foundation through grant CCF 1811729.

[1] The standard proof that the Kolmogorov complexity function is not computable provides such counter examples. Indeed, for the sake of contradiction, assume that for some computable time bound $t : \mathbb{N} \to \mathbb{N}$, the following holds: For all n and n-bit x with complexity smaller than $2 \log n$, there exists a program of length at most $n/10$ that generates x in time $t(n)$. Then the lexicographically first n-bit string x that does not have such a program, has complexity at most $\log n + O(1)$, which is a contradiction for large n.

Authors' addresses: Bruno Bauwens, National Research University Higher School of Economics, Faculty of Computer Science, Moscow, Russia, brbauwens@gmail.com; Marius Zimand, Towson University, Department of Computer and Information Sciences, Baltimore, MD, USA, mzimand@towson.edu.

Manuscript submitted to ACM

optimal compressor can be used for any description system, as explained in the abstract. Moreover, it can be used for distributed compression as well.

Our study of compression uses a more general approach than what is standard in information theory. In information theory, the underlying assumption is that data is generated by a stochastic process, whose structural properties, such as various types of independence, ergodicity, various statistics of the distribution, etc., are at least partially used in constructions or in the analysis. Instead, we do not assume any generative model, and require that for *each* string, the compressed length is as small as in any other compressor-decompressor pair up to an additive polylogarithmic term. Our main results subsume important results in information theory: the Shannon source coding theorem and the Slepian-Wolf coding theorem. Moreover, they provide corresponding results in algorithmic information theory, where the optimal compression length of a string is measured by its Kolmogorov complexity relative to an optimal Turing machine.

The contributions of this paper concern both single-source compression and multi-source distributed compression.

Single source compression

By direct diagonalization, one may observe that no (probabilistically) computable compressor is optimal: for each such compressor $\mathcal{C}$ mapping binary strings to binary strings, there exists another deterministic compressor $\mathcal{C}'$ and a sequence $x_1, x_2, \ldots$ of strings with length $|x_n| = n$ such that $|\mathcal{C}(x)| \geq n$ (using the maximal length over all randomness if $\mathcal{C}$ is probabilistic) and $|\mathcal{C}'(x)| \leq O(\log n)$. To overcome this barrier, we consider compression functions that have a "target compression size" as an additional input. Our compressors are also probabilistic.

Consequently, a compressor $\mathcal{C}$ is a probabilistic algorithm that has three inputs: the string x being compressed, the target compression length m, and the error probability ε. The output is a random variable, denoted $\mathcal{C}_{\varepsilon,m}(x)$, whose realization is a string of length m.

A *decompressor* $\mathcal{D}$ is a deterministic partial function mapping strings (viewed as compressed codes) to strings. The *Kolmogorov complexity* of x relative to the decompressor $\mathcal{D}$, is given by

$$\mathrm{C}_{\mathcal{D}}(x) \;=\; \min\{|p| : \mathcal{D}(p) = x\}.$$

Note that unlike standard Kolmogorov complexity theory, we do not restrict $\mathcal{D}$ to be partial computable. Our definition of optimal compressors requires that (*i*) the length of the compressed code is equal to the target length, and (*ii*) simultaneously for *all* decompressors $\mathcal{D}$, correct decompression is achieved as soon as m is slightly above $\mathrm{C}_{\mathcal{D}}(x)$.

Definition 1.1. *A* compressor $\mathcal{C}$ *is a probabilistic function that maps* $\varepsilon > 0$, m *and* x, *with probability* 1, *to a string* $\mathcal{C}_{\varepsilon,m}(x)$ *of length exactly* m. *Let* Δ *be a function of* ε, m *and* x, *called* overhead. *A compressor* $\mathcal{C}$ *is* Δ-optimal *if for every decompressor* $\mathcal{D}$ *there exists a decompressor* $\mathcal{D}'$ *such that for all* $\varepsilon > 0$, x *and* $m \geq \mathrm{C}_{\mathcal{D}}(x) + \Delta$:

$$\Pr\left[\mathcal{D}'\big(\mathcal{C}_{\varepsilon,m}(x)\big) = x\right] \;\geq\; 1 - \varepsilon.$$

In our constructions, the mapping from $\mathcal{D}$ to $\mathcal{D}'$ is effective in a certain sense, and this implies that if $\mathcal{D}$ is partial computable, respectively, computable, and computable in polynomial space, then so is $\mathcal{D}'$, see Remark 2.

Our main result for single-source compression states that for some polylogarithmic overhead Δ, an optimal compression function exists, and can even be computed in polynomial time.

Theorem 1.1. *There exists a* $O(\log m \cdot \log \frac{|x|}{\varepsilon})$*-optimal compressor that is computable in time polynomial in* $|x|$.

In other words, any slow compressor in a compressor-decompressor pair can be replaced by a fixed fast compressor and slower decompressor, at the expense of a small infrease of the compression length.

Remark. It is standard in compression to assume that the parties do not share a source of randomness. If they do, the result of Theorem 1.1 can be obtained via simple hashing, as we explain in section 2.

In Kolmogorov complexity, one typically fixes an optimal[2] partial computable decompressor $\mathcal{U}$ and writes $C(x)$ in lieu of $C_{\mathcal{U}}(x)$. $C(x)$ is called the *Kolmogorov complexity* or the *minimal description length* of x. In a similar way, for two strings x and y, we define $C(x \mid y)$, the complexity of x conditional on y, see equation (2). Theorem 1.1 implies polynomial time compression of a string x down to almost its minimal description length, more precisely down to length $m = C(x) + O(\log^2(|x|/\varepsilon))$, provided the compressor works with target length m.

Corollary 1.2. *There exists a polynomial-time computable compressor $\mathcal{C}$ and a constant c, such that for all $\varepsilon > 0$, m and x with $C(x) \le m - c \cdot \log m \cdot \log \frac{|x|}{\varepsilon}$:*

$$\Pr[\mathcal{U}(\mathcal{C}_{\varepsilon,m}(x)) = x] \;\ge\; 1 - \varepsilon.$$

The compressor in Theorem 1.1 also provides a solution to the so-called *document exchange problem*, which can be formulated as follows. Suppose y is the obsolete version of an updated file x. The receiver holds y and the sender holds x. The sender transmits $\mathcal{C}_{\varepsilon,m}(x)$ to the receiver, and if $C(x \mid y) \le m - \Delta$, then the receiver can reconstruct x. This follows from the optimality condition in Definition 1.1 applied for $\mathcal{D}(\cdot) = \mathcal{U}(\cdot, y)$, where $\mathcal{U}$ is the optimal machine in the definition of $C(\cdot \mid \cdot)$. It is remarkable that the sender computes $\mathcal{C}_{\varepsilon,m}(x)$ without knowing y, but only the target length $m = C(x \mid y) + \Delta$.

The overhead Δ can be improved from polylogarithmic to logarithmic at the cost of a slower compressor running in exponential space, and, hence, double exponential time.

Proposition 1.3. *There exists a $O(\log \frac{|x|}{\varepsilon})$-optimal compressor that is computable in exponential space.*

The compressor in Proposition 1.3 is also probabilistic and uses a logarithmic number of random bits. We prove the following lower bounds, which show that the overhead and the randomness in Proposition 1.3 are essentially minimal: Every compressor with a computably bounded running time has overhead $\Delta \ge \Omega(\log \frac{|x|}{\varepsilon})$, and if $\Delta \le O(|x|^{0.99})$, at least a logarithmic amount of randomness is needed. Let $\lceil a \rceil = \mathrm{ceil}(a)$.

Proposition 1.4. *Let Δ and r be functions of ε and $n = |x|$. If there exists a Δ-optimal compressor that can be evaluated with at most r random bits in a computably bounded running time, then*

- $\Delta \ge \log \frac{n}{\varepsilon} - \log\log \frac{n}{\varepsilon} - 8$ *for all n and ε with $2^{-n/4} \le \varepsilon \le 1/4$,*
- $\left\lceil \varepsilon 2^{r+1} \right\rceil \;\ge\; \frac{n - r - \log(2/\varepsilon)}{\Delta + 4}$ *for all n and $\varepsilon \le 1/2$.*

The second item for $r = 0$ and $\varepsilon = 1/2$ implies that a deterministic computable compressor can only be Δ-optimal for $\Delta \ge n - 6$. Essentially the same bounds also apply for a weaker version of Δ-optimality in which we consider a single optimal Turing machine as decompressor, and allow probabilistic decompression, see Lemmas 4.1 and 4.2.

[2] $\mathcal{U}$ is a Turing machine, and its optimality means that for every other Turing machine $\mathcal{D}$, there exists a constant c such for every string x, $C_{\mathcal{U}}(x) \le C_{\mathcal{D}}(x) + c$. Here, we assume that $\mathcal{U}$ is *universal* in the sense that for each other Turing machine $\mathcal{D}$, there exists a string w such that $\mathcal{U}(wp) = \mathcal{D}(p)$ for all strings p for which $\mathcal{D}(p)$ is defined.

Distributed compression

In its most basic form, distributed compression is the task of compressing correlated pieces of data by several parties, each one possessing one piece and acting separately. This task is also known as *Slepian-Wolf coding*.

For illustration, let us consider the following simple example, with two senders, Alice and Bob, and a receiver, Zack. Alice knows a line L of the form $y = ax + b$ in the affine plane over a finite field with 2^n elements, and Bob knows a point P with coordinates (u, v) on this line. Each of L and P has $2n$ bits of information, but, because of the geometrical relation, together they only have $3n$ bits of information. Alice and Bob want to email L and P to Zack, without wasting bandwidth. If they have to compress the message in isolation, how many bits should they send to Zack?

Obviously, Alice can send L, so $2n$ bits, and then Bob can send n bits, or symmetrically Bob can send P, so $2n$ bits, and then Alice can send n bits. What about other compression lengths? Some necessary requirements for the compression lengths are easy to see. Let k_A be the number of bits to which Alice compresses her point L, and let k_B have the analogous meaning for Bob. It is necessary that $k_A + k_B \geq 3n$, $k_A \geq n$ and $k_B \geq n$. The first inequality holds because Zack needs to acquire $3n$ bits, and the other two hold because if Zack gets somehow either L or P, he still needs n bits of information about the remaining element.

Without any assumption regarding how L and P were generated, the version of Slepian-Wolf coding in this paper implies that any numbers k_A and k_B satisfying the above necessary conditions (such as, for instance, $k_A = k_B = 3n/2$), are also sufficient up to a small polylogarithmic overhead. More precisely, there exists a probabilistic polynomial-time compression algorithm such that if k_A and k_B satisfy these conditions, then Alice can apply the algorithm to compress the line L to a binary string p_A of length $k_A + O(\log^2 n)$, Bob can compress point P to a binary string p_B of length $k_B + O(\log^2 n)$, and Zack can with high probability reconstruct L and P from p_A and p_B.

In the general case (presented here for simplicity for two senders), Alice has a string x_A, Bob has x_B, which they want to send to Zack. Suppose Zack is using the decompressor $\mathcal{D}$. Let k_A, k_B represent the compression lengths as above, and let $C_{\mathcal{D}}(x_A, x_B), C_{\mathcal{D}}(x_A \mid x_B), C_{\mathcal{D}}(x_B \mid x_A)$ denote the Kolmogorov complexities relative to $\mathcal{D}$ of (x_A, x_B), and respectively, of x_A conditioned by x_B and of x_B conditioned by x_A. Similarly to the simple example with the line and the point, for Zack to be able to reconstruct (x_A, x_B), it is necessary (up to a small additive term) that k_A and k_B satisfy the following inequalities, called the Slepian-Wolf constraints:

$$\begin{aligned} k_A &\geq C_{\mathcal{D}}(x_A \mid x_B) \\ k_B &\geq C_{\mathcal{D}}(x_B \mid x_A) \\ k_A + k_B &\geq C_{\mathcal{D}}(x_A, x_B). \end{aligned} \tag{1}$$

Note that these requirements on the compression lengths are necessary even when Alice and Bob are allowed to collaborate. We show that modulo a polylogarithmic overhead, the conditions are also sufficient. Moreover, such compression is achieved by the optimal compressors from Definition 1.1.

Formally, let $\mathcal{D}$ be a decompressor mapping pairs of strings to strings. The *conditional* Kolmogorov complexity of w given z is

$$C_{\mathcal{D}}(w|z) = \min\{|p| : \mathcal{D}(p, z) = w\}. \tag{2}$$

Let $[\ell] = \{1, 2, \ldots, \ell\}$. The parameter ℓ represents the number of senders. Given $J \subseteq [\ell]$, and an ℓ-tuple $\mathrm{x} = (x_1, \ldots, x_\ell)$, let x_J be the set $\{(j, x_j) : j \in J\}$. The Kolmogorov complexity of x_J is defined by encoding this finite set as a string in some canonical way.

Definition 1.2. *An ℓ-tuple* $\mathrm{k} = (k_1, \ldots, k_\ell)$ *of integers satisfies the* $\mathcal{D}$-Slepian-Wolf constraints *for* x *if*

$$\mathrm{C}_{\mathcal{D}}\left(\mathrm{x}_J \mid \mathrm{x}_{[\ell]\setminus J}\right) \;<\; \sum_{j\in J} k_j \quad \textit{for all non-empty } J.$$

Theorem 1.5. *Assume $\mathcal{C}$ is Δ-optimal for some computable Δ. For every decompressor $\mathcal{D}$ there exists a decompressor $\mathcal{D}'$ such that for all $\varepsilon > 0$, ℓ, ℓ-tuples* x, *and ℓ-tuples* m*:*

$$\Pr\left[\mathcal{D}'\big(\varepsilon, \mathcal{C}_{\varepsilon,m_1}(x_1), \ldots, \mathcal{C}_{\varepsilon,m_\ell}(x_\ell)\big) = \mathrm{x}\right] \;\geq\; 1 - 8\ell\varepsilon,$$

provided $(m_1 - \Delta_1 - \log\frac{\ell}{\varepsilon},\ \ldots,\ m_\ell - \Delta_\ell - \log\frac{\ell}{\varepsilon})$ *satisfies the $\mathcal{D}$-Slepian-Wolf constraints for* x, *where Δ_j is the value of Δ corresponding to the inputs ε, m_j and x_j.*

The compressors in this definition work in isolation in a very strong sense: besides their string x_j, they only use the targets ε and m_j. They do not use the decompressor $\mathcal{D}$ or even ℓ.

If we apply this theorem to the compressor in the proof of Theorem 1.1, then a similar remark for the effectivity of the transformation from $\mathcal{D}$ to $\mathcal{D}'$ holds as for Definition 1.1: if $\mathcal{D}$ is partial computable, respectively, computable, and computable in exponential time, then so is $\mathcal{D}'$, see Remark 4. Moreover, for growing ℓ, the overhead $\Delta_j + \log\frac{\ell}{\varepsilon}$ is logarithmic in ℓ and the error is linear in ℓ. Hence, the result can be meaningful even when the number of senders is exponential in the length of the strings.

Related works and comparison with our results

The study of the fundamental limits of data compression has been undertaken in both Information Theory (IT) and Algorithmic Information Theory (AIT). We recall that the major conceptual difference between the two approaches is that in IT data is viewed as being the outcome of a generative process, whereas in AIT the data consists of individual strings without any assumption regarding their provenance. One can say that IT focuses on *random processes*, that is processes whose outcomes have a degree of uncertainty, and AIT focuses on *random strings*, that is strings that do not admit a concise description due to the lack of discernible patterns. Our approach is in the AIT spirit, but is more general, because it is not restricted to computable decompressors.

We start by reviewing basic results in IT. For concreteness, let us first suppose that an n-bit string is produced by a single activation of some generative process (this is called the "one-shot" scenario). The process is represented by a random variable $X^{(n)}$ which takes values in $\{0,1\}^n$. A zero-error compression procedure is an injective mapping $\mathcal{C} : \{0,1\}^n \to \{0,1\}^*$ and if $\mathcal{C}(u) = v$ we think that v is the compressed encoding of u. The goal is to minimize $L_{\mathrm{avg}}(\mathcal{C})$, which is the expected length of $\mathcal{C}(X^{(n)})$. If the range of $\mathcal{C}$ is a prefix-free set (or, more generally, a so-called instantaneous code), which is desirable if we want to extend compression to a "multiple-shot" setting, then the Kraft inequality implies that the average length has to be at least the Shannon entropy of $X^{(n)}$, i.e., $L_{\mathrm{avg}}(\mathcal{C}) \geq H(X^{(n)})$.

Huffman coding [Huffman 1952] is an algorithm that computes a prefix-free coding $\mathcal{C}_{\mathrm{Huffman}}$ with minimal average length satisfying $H(X^{(n)}) \leq L_{\mathrm{avg}}(\mathcal{C}_{\mathrm{Huffman}}) \leq H(X^{(n)}) + 1$. The drawback is that Huffman compression takes time exponential in n and requires the knowledge of the distribution of $X^{(n)}$.

One way to overcome these issues it to depart from the "one-shot" scenario and to consider that the data is produced by some stochastic process $X_1, X_2, \ldots$ which often satisfies simplifying assumptions, and also to allow some error. The standard stochastic process of this type is the *memoryless source*, which means that $X^{(n)} = (X_1, X_2, \ldots, X_n)$ are n i.i.d. draws from some distribution p_X on $\{0,1\}$. A fixed-length code with compression rate R is given by a family of

functions $(\mathcal{C}_n, \mathcal{D}_n)_{n\in\mathbb{N}}$ of type $\mathcal{C}_n : \{0,1\}^n \to \{0,1\}^{Rn}$, $\mathcal{D}_n : \{0,1\}^{Rn} \to \{0,1\}^n$. The error of the code is defined to be $\varepsilon_n = \text{Prob}(\mathcal{D}_n(\mathcal{C}_n(X^{(n)})) \neq X^{(n)})$.

The Shannon Source Coding Theorem [Shannon 1948] shows that if $R > H(p_x)$ then there exists a code $(\mathcal{C}_n, \mathcal{D}_n)_{n\in\mathbb{N}}$ with compression rate R and $\lim_{n\to\infty} \varepsilon_n = 0$, and if $R < H(p_x)$, then every code $(\mathcal{C}_n, \mathcal{D}_n)_{n\in\mathbb{N}}$ with compression rate R has $\inf \varepsilon_n > 0$. The compression in the positive part of the Source Coding theorem takes time exponential in n and, since the error is not zero, the decompression fails for some strings.

Similarly to the Source Coding theorem for single-source compression, the Slepian-Wolf theorem [Slepian & Wolf 1973] characterizes the possible compression lengths for *distributed compression* in the case of memoryless sources. For notational convenience, our short description of the Slepian-Wolf theorem is restricted to the case of two senders (Alice and Bob) and of data represented in the binary alphabet. Recall that in distributed compression, Alice observes $X^{(n)}$, and Bob observes $Y^{(n)}$, where $(X^{(n)}, Y^{(n)})$ have a joint distribution over $\{0,1\}^n \times \{0,1\}^n$.

In the case of a *2-Discrete Memoryless Source* (2-DMS), $(X^{(n)}, Y^{(n)})$ are obtained by n i.i.d. draws from some distribution $p_{x,y}$ on $\{0,1\} \times \{0,1\}$. In other words, a 2-DMS is given by a sequence $(X_1, Y_1), (X_2, Y_2), \ldots$, where (X_i, Y_i) is the i-th independent draw from $p_{x,y}$. Alice observes $X^{(n)} = (X_1, \ldots, X_n)$ and Bob observes $Y^{(n)} = (Y_1, \ldots, Y_n)$. A distributed compression procedure with compression rates (R_1, R_2) consists of a family of functions $(\mathcal{C}_{1,n}, \mathcal{C}_{2,n}, \mathcal{D}_n)_{n\in\mathbb{N}}$ of the type $\mathcal{C}_{1,n} : \{0,1\}^n \to \{0,1\}^{R_1 n}$, $\mathcal{C}_{2,n} : \{0,1\}^n \to \{0,1\}^{R_2 n}$, $D_n : \{0,1\}^{R_1 n} \times \{0,1\}^{R_2 n} \to \{0,1\}^n \times \{0,1\}^n$. The error is defined as $\varepsilon_n = \text{Prob}(\mathcal{D}_n(\mathcal{C}_{1,n}(X^{(n)}), \mathcal{C}_{2,n}(Y^{(n)})) \neq (X^{(n)}, Y^{(n)}))$.

The compression rates (R_1, R_2) are *achievable* if $\lim_{n\to\infty} \varepsilon_n = 0$, and the *achievable rate region* is the closure of the set of achievable rates. The Slepian-Wolf theorem states that for a 2-DMS as above the achievable rate region is given by all (R_1, R_2) satisfying $R_1 \geq H(X_1 \mid Y_1)$, $R_2 \geq H(Y_1 \mid X_1)$, and $R_1 + R_2 \geq H(X_1, Y_1)$ (these inequalities form the Slepian-Wolf constraints).

The Source Coding Theorem has been extended to sources that are stationary and ergodic (using their asymptotic equipartition property [Cover & Thomas 2006, Theorem 16.8.1]), and even to arbitrary sources [Han & Verdú 1993], using the *information-spectrum concepts* initiated by Han and Verdú (see the monograph [Han 2003]).

Similar extensions exist for the Slepian-Wolf theorem: it has been generalized to sources that are stationary and ergodic [Cover 1975], and, using the information-spectrum framework, a Slepian-Wolf coding theorem has been obtained for general sources [Miyake & Kanaya 1995]. However, these latter results involving non-memoryless sources have a strong asymptotic nature and require that the distribution of $X^{(n)}$ (and, respectively, of $(X^{(n)}, Y^{(n)})$ for Slepian-Wolf compression) is known.

In summary, the IT results mentioned above show that for data produced by a stochastic process:

- the minimal compression length for single-source compression is arbitrarily close to Shannon entropy in case of a memoryless source, or to some asymptotical version of entropy for more general sources; in the case of distributed compression, the optimal compression lengths are dictated by the Slepian-Wolf constraints, and
- optimal compression relies either on the very simple memoryless property, or on complete knowledge of the distribution.

Theorem 1.1 and Theorem 1.5 can be viewed as a strengthening of the Source Coding and Slepian-Wolf theorems. The main merit of our results compared to the classical theorems is that the compressor works in polynomial time, and is universal in the strong sense of Definition 1.1, which, in particular, means that it does not assume any generative model.

Another significant difference is that whereas in the IT results mentioned above (with the exception of Huffman coding) compression fails for some realizations of the sources, the compression in our results works for all input tuples satisfying the promise condition.

Our theorems imply their IT counterparts for memoryless sources and for stationary and ergodic sources using the fact that a sequence of i.i.d. random variables (or, more generally, the outcome of a stationary ergodic source) gives with a high probability a string of letters whose Kolmogorov complexity is close to Shannon's entropy of the random source (this was stated in [Zvonkin & Levin 1970, Proposition 5.1] and a proof appears in [Brudno 1982]; see also [Horibe 2003; Ming & Vitányi 2014]).

Compression procedures for individual inputs (i.e., without using any knowledge regarding the generative process) have been previously designed using the celebrated Lempel-Ziv methods [Lempel & Ziv 1976; Ziv 1978]. This approach has lead to efficient compression algorithms that are used in practice. Such methods have been used for distributed compression as well [Dueck & Wolters 1985; Kuzuoka 2009; Ziv 1984]. For such procedures two kinds of optimality have been established, both valid for infinite sequences and thus having an asymptotic nature.

First, the procedures achieve a compression length that is asymptotically equal to the so-called finite-state complexity, which is the minimum length that can be achieved by finite-state encoding/decoding procedures [Ziv 1978].

Secondly, the compression rates are asymptotically optimal in case the infinite sequences are generated by sources that are stationary and ergodic [Wyner & Ziv 1994].

In contrast, the compression in Theorem 1.1 applies to finite strings and achieves a compression length close to minimal description length. Unfortunately, it cannot be practical, because, as we have explained, when compression is done at this level of optimality, decompression requires time longer than any computable function. Our theoretical results show that compression is not a fundamental obstacle in the design of an efficient compressor-decompressor pair that is optimal in some rigorous sense, and we hope that this principle will inspire and guide future research lines with applicative objectives.

We now move to previous results obtained in the AIT framework. How well can we compress a string x? By a simple counting argument, there is no compressor (even probabilistic, and even incomputable) that compresses all strings x down to length $\mathrm{C}(x) - c$, for some constant c. As already mentioned no computable procedure can compress x down to a string of length $\mathrm{C}(x)$.

On the other hand, an easy argument shows that, if $\mathrm{C}(x)$ is also given (such an additional information regarding the input is known as *help*), then a description of x of length $\mathrm{C}(x)$ can be obtained by exhaustive search, which, unfortunately, is an exceedingly slow procedure whose running time grows faster than any computable function. Corollary 1.2 shows that in fact a description of x of almost minimal length can be found in probabilistic polynomial time.

An interesting problem is the compression of x conditioned by some unavailable information y, also known as, the *asymmetric* version of the Slepian-Wolf coding or *source coding with side information at the receiver* in the information-theory literature, and as *information reconciliation* or the *document exchange problem* in the theoretical computer science literature. This time, a compressor which knows $\mathrm{C}(x \mid y)$, but not y, cannot perform the exhaustive search.

Muchnik's theorem [Muchnik 2002] gives a surprising solution: There exist algorithms $\mathcal{C}$ and $\mathcal{D}$ such that for all n and for all n-bit strings x and y, $\mathcal{C}$ on input x, $\mathrm{C}(x \mid y)$ and $O(\log n)$ help bits outputs a string p of length $\mathrm{C}(x \mid y)$, and $\mathcal{D}$ on input p, y, and $O(\log n)$ help bits reconstructs x. Thus, given $\mathrm{C}(x \mid y)$, Alice can compute from her string x and only $O(\log n)$ additional help bits a string p of minimum description length such that Zack using p, y and $O(\log n)$ help bits can reconstruct x.

Muchnik's theorem has been strengthened in several ways. Musatov, Romashchenko and Shen [Musatov *et al.* 2011] have obtained a version of Muchnik's theorem for space bounded Kolmogorov complexity, in which both compression and decompression are space-efficient. Romashchenko [Romashchenko 2005] has extended Muchnik's theorem to the general (i.e., non-asymmetric) Slepian-Wolf coding. His result is valid for any constant number of senders, but, for simplicity, we present it for the case of two senders: For any two n-bit strings x and y and any two numbers n_1 and n_2 satisfying the Slepian-Wolf constraints (1), there exist two strings p_1 and p_2 such that $|p_1| = n_1 + O(\log n)$, $|p_2| = n_2 + O(\log n)$, $\mathrm{C}(p_1 \mid x) = O(\log n)$, $\mathrm{C}(p_2 \mid y) = O(\log n)$ and $\mathrm{C}(x, y \mid p_1, p_2) = O(\log n)$. In words, for any n_1 and n_2 satisfying the Slepian-Wolf constraints, Alice can compress x to a string p_1 of length just slightly larger than n_1, and Bob can compress y to a string p_2 of length just slightly larger than n_2 such that Zack can reconstruct (x, y) from (p_1, p_2), provided all the parties use $O(\log n)$ help bits.

Bauwens et al. [Bauwens *et al.* 2013], Teutsch [Teutsch 2014] and Zimand [Zimand 2014] have obtained versions of Muchnik's theorem with polynomial-time compression, but in which the help bits are still present. In fact, their results are stronger in that the compression procedure on input x outputs a polynomial-size list of strings guaranteed to contain a short program for x given y. This is called list approximation. Note that using $O(\log n)$ help bits, the compression procedure can pick the right element from the list, re-obtaining Muchnik's theorem. The gain is that this compression procedure halts even with incorrect help bits, albeit in this case the result may not be the desired x, see appendix G for details.

Bauwens and Zimand [Bauwens & Zimand 2014] show the existence of polynomial-time list approximation algorithms for programs of minimal description lengths that do not require help bits, but which, instead, are probabilistic. Zimand [Zimand 2017] obtains Slepian-Wolf coding in the Kolmogorov complexity setting via probabilistic polynomial-time encoding without help bits.

We were inspired by [Bauwens & Zimand 2014] and [Zimand 2017], which represent the starting point for our study. This work contains conceptual and technical ideas that depart significantly from these two papers. The main conceptual novelty is that the optimal compressors in this paper are universal, i.e., they work for any decompressor, computable or not, in the sense explained in Definition 1.1.

On the technical side, we use a new fingerprinting technique, based on condensers and conductors, which leads to smaller overhead ($\log^2 n$, compared to $\log^3 n$ in [Zimand 2017]).

Also, the Slepian-Wolf coding in Theorem 1.5 provides a double exponential improvement of the dependence of the overhead on the number ℓ of senders compared to [Zimand 2017]. The latter paper implicitly assumes that ℓ is constant; otherwise, the overhead obtained there makes the result essentially meaningless. The same is true for the classical Slepian-Wolf theorem: no non-trivial compression can be achieved if the number of senders is significantly larger than the length of the compressed strings.

2 OPTIMAL SINGLE SOURCE COMPRESSION: PROOF OF THEOREM 1.1

The main novel features of the proof are an efficient fingerprinting construction based on condensers and conductors and a method to handle fingerprints that produce many collisions. We present below the basic ideas.

Invertible functions

In Definition 1.1, the optimal compressor and the corresponding decompressor $\mathcal{D}'$ only need to work correctly for strings x in

$$\{x : \mathrm{C}_D(x) < k\}, \tag{*}$$

where $k = m - \Delta + 1$. For different $\mathcal{D}$, these sets are different, however all of them have size less than 2^k. This set contains the initial suspects, from which the decompressor has to find the compressed string x. The optimal compressor provides a *fingerprint*, i.e., a small amount of information about the input string x that allows its identification among these suspects. Because of the universality property, the compressor needs to generate fingerprints without knowing the list of suspects. For any set $\mathcal{X}$, let $\mathcal{X}^{\leq K}$ denote the set of lists of elements in $\mathcal{X}$ with size at most K. The above discussion leads to the following definition.

Definition 2.1. *A probabilistic function* $\mathrm{F}\colon \mathcal{X} \to \mathcal{Y}$ *is* (K, ε)-invertible *if there exists a deterministic partial function* $g\colon \mathcal{X}^{\leq K} \times \mathcal{Y} \to \mathcal{X}$ *such that for all* $S \in \mathcal{X}^{\leq K}$ *and all* $x \in S$*:*

$$\Pr\left[g_S(\mathrm{F}(x)) = x\right] \;\geq\; 1 - \varepsilon\,,$$

where $g_S(y) = g(S, y)$. F *is* online (K, ε)*-invertible if there exists such a function* g *that is* monotone *in* S*: if list* S' *extends* S*, then the function* $y \mapsto g_{S'}(y)$ *is an extension of* $y \mapsto g_S(y)$.

The interpretation is that g, on input a list S of suspects and a random fingerprint $\mathrm{F}(x)$ of x, identifies x with high probability among the suspects. The main technical result, from which Theorem 1.1 follows, is a probabilistic polynomial time algorithm that computes an invertible function.

Theorem 2.1. *There exists a probabilistic algorithm* F *that on input* $\varepsilon > 0$*,* k *and string* x*, outputs in time polynomial in* $|x|$ *a string* $\mathrm{F}_{\varepsilon,k}(x)$ *of length* $k + O(\log k \cdot \log(|x|/\varepsilon))$*, such that for all* $\varepsilon > 0$ *and* k*, the function* $x \mapsto \mathrm{F}_{\varepsilon,k}(x)$ *is online* $(2^k, \varepsilon)$*-invertible.*

Moreover, there exists a family of monotone inverses $g_{\varepsilon,k}$ *of* $\mathrm{F}_{\varepsilon,k}$*, for which the mapping* $(\varepsilon, k, S, y) \mapsto g_{\varepsilon,k,S}(y)$ *can be evaluated in space polynomial in* $\max_{z \in S} |z|$.

Proof of Theorem 1.1 (assuming Theorem 2.1). Without loss of generality, we assume that ε is a negative power of 2, because rounding down ε increases the overhead by less than a constant factor. Hence, we can represent ε in binary using at most $O(\log \frac{1}{\varepsilon})$ bits. The optimal compressor outputs the string $\mathcal{C}_{\varepsilon,m}(x)$ which is an m-bit representation of the triple $(\varepsilon, k(\varepsilon, m, |x|), \mathrm{F}_{\varepsilon,k}(x))$, where the function k is chosen large enough such that $m - k(\varepsilon, m, n) \leq O(\log m \cdot \log(n/\varepsilon))$, but not too large such that the triple can still be represented in binary using precisely m bits. On input a triple (ε, k, y), the decompressor $\mathcal{D}'$ first enumerates the set S of equation (*), and outputs $g_S(y)$, where g is the monotone inverse of $\mathrm{F}_{\varepsilon,k}$. □

Remark 1. The theorem considers inputs x of arbitrary length. However, it is enough to construct for each fixed input size n, an invertible $\mathrm{F}\colon \{0,1\}^n \mapsto \{0,1\}^m$ with inverse g. Indeed, by adding n to the output, we obtain an inverse in $\{0,1\}^*$, simply by evaluating the inverse for n-bit inputs on the n-bit elements in (*).

Remark 2. The definition of Δ-optimality requires the existence of some mapping $\mathcal{D} \mapsto \mathcal{D}'$, but does not impose any computability requirement on this mapping. However, from the proof of Theorem 1.1 we obtain that if $\mathcal{D}$ is given as an oracle, we can evaluate $\mathcal{D}'$ for an n-bit compressed string in space polynomial in n, and hence in time exponential in n. Indeed, assume the length n is a part of the output, as explained in the previous remark. On input (ε, k, y, n), the

decompressor $\mathcal{D}'$ uses the oracle to enumerate the set S of n-bit strings in (*). Each time such a string is found, the value of $g_S(y)$ is calculated, and the output of $\mathcal{D}'$ is the corresponding value. The monotonicity property guarantees that the output of $\mathcal{D}'$ does not change after new elements appear in S. If $\mathcal{D}$ is computable in polynomial space, then so is the corresponding $\mathcal{D}'$. Similarly, if $\mathcal{D}$ is computable, respectively, partial computable, then so is $\mathcal{D}'$.

It remains to prove Theorem 2.1. To better understand the main difficulty, we first review weaker results based on standard fingerprinting techniques.

Fingerprints from random hashing

Fix $\varepsilon > 0$ and K. Theorem 2.1 provides (K, ε)-invertible functions with output length close to $\log K$. Consider the variant of Definition 2.1 in which the functions F and g are evaluated using shared randomness in their evaluations. In this model, F and g have an extra argument ρ representing a random string. Fingerprints given by random subset parities provide (K, ε)-invertible functions for this easier setting.

The fingerprints. On input x and a sufficiently long ρ, let $\mathrm{F}_\rho(x)$ be a string of size $m = \lceil \log \frac{K}{\varepsilon} \rceil$ evaluated by taking m disjoint segments $\rho_1, \ldots, \rho_m$ of length $|x|$ from ρ, and setting the i-th bit equal to $\sum_{j \le n} \rho_{i,j} x_j \bmod 2$. In other words, $\mathrm{F}_\rho(x) = Rx$, where R is the matrix having the rows $\rho_1, \ldots, \rho_m$, and the ρ_i's and x are viewed as n-vectors with elements in the field GF[2].

The inverse g. On input ρ, a list S of strings, and y, the value $g_{\rho,S}(y)$ is given by the first z in S for which $\mathrm{F}_\rho(z) = y$.

Lemma 2.2. *If S is a list of at most K strings of equal length and $x \in S$, then* $\Pr_\rho \left[g_{\rho,S}(\mathrm{F}_\rho(x)) = x \right] \ge 1 - \varepsilon$.

Proof. For two different strings x and z of equal length, the probability over ρ that $\mathrm{F}_\rho(x) = \mathrm{F}_\rho(z)$ is at most $2^{-m} \le \varepsilon/K$. By the union bound, the probability that some $z \in S$ different from x, has the same subset parities, is at most ε. This bounds the probability that g returns an element different from x. □

How can shared randomness be eliminated? One idea is to attach the random bits to the output of F. However, on input x, the technique above uses $m \cdot |x|$ random bits, and after attaching, the length exceeds the output length of the identity function. We can try another well known hashing technique based on arithmetic modulo prime numbers, which only requires about $\log K$ random bits, and provides output lengths close to $2 \log K$. This hashing technique will be used later, and is based on the following.

Lemma 2.3. *If $x, z_1, \ldots, z_K$ are different nonnegative integers less than 2^n and P is a set of at least Kn/ε prime numbers, then for a fraction $1 - \varepsilon$ of primes p in P: $x \bmod p \notin \{z_1 \bmod p, \ldots, z_K \bmod p\}$.*

Proof. Each number $(x - z_i)$ has at most n different prime factors. Thus all $(x - z_i)$'s together have at most Kn different prime factors. Hence a random element in a set of at least Kn/ε primes, is a prime factor with probability at most ε. □

Interpret strings x as nonnegative integers smaller than $2^{|x|}$. Let the fingerprint $\mathrm{H}_{\varepsilon,K}(x) = (p, x \bmod p)$ be given by selecting p randomly among the primes of bit size at most $s + \lceil \log s \rceil + 1$, where $s = \log(K|x|/\varepsilon)$.

Lemma 2.4. *For all $\varepsilon > 0, K$ and n, the fingerprint $\mathrm{H}_{\varepsilon,K}$ applied to n-bit strings, defines a (K, ε)-invertible function with output size at most $2 \log K + O(\log \frac{n}{\varepsilon})$.*

Proof. The prime number theorem implies that the i-th prime is less than $2i \log i$ for large i. Hence, the set of primes of bit size at most $s + \lceil \log s \rceil + 1$ contains at least $2^s = Kn/\varepsilon$ primes. On input a set S and a pair (p, j), the function g outputs an element x in S for which $x = j \bmod p$. By the lemma above, with probability $1 - \varepsilon$, there exists no such z different from x, and hence, the decompressor outputs the correct value. □

Let us summarize: a $(2^k, \varepsilon)$-invertible function has output size at least k. The fingerprints of Lemma 2.2 have output size $k + O(\log \frac{1}{\varepsilon})$, which is close to optimal, but use shared randomness. Lemma 2.4 does not use shared randomness, but the method cannot achieve compression lengths better than $2k + O(\log \frac{1}{\varepsilon})$.[3]

The technique used in the above lemmas produces inverses g of F of a simple form: $g_S(y)$ searches for the first $x \in S$ such that $\mathrm{F}(x) = y$ has positive probability. We show that this technique has a limitation that precludes optimal compression for all inputs. Let F_x denote the set of all values of $\mathrm{F}(x)$ that appear with positive probability. For the technique to work correctly, we need that with positive probability F_x is not included in $\bigcup\{F_z : z \in S, z \neq x\}$, in other words, the fingerprint $\mathrm{F}(x)$ does not cause x to collide with any other $z \in S$. Unfortunately, such inverses for lists of suspects of size 2^k, always require output sizes $m \geq 2k$, instead of the desired k.

Lemma 2.5. *For all $x \in \mathcal{X}$, let $F_x \subseteq Y$. If $\#\mathcal{Y} < \min\{\frac{1}{4}K^2, \#\mathcal{X} - \frac{1}{2}K\}$, there exists an $x \in \mathcal{X}$ and a set $S \subseteq \mathcal{X}$ of size less than K such that*

$$F_x \subseteq \bigcup_{z \in S, z \neq x} F_z \,.$$

The lemma follows from a more general theorem about union-free sets, see Jukna [Jukna 2011, Th 8.13]. In appendix A we give a simple proof of the lemma.

Fingerprints from condensers and conductors

We bypass the issue from Lemma 2.5 by allowing a limited amount of collisions. By using a second fingerprint and a more complex inverse function g, each input can still be recovered with high probability. The required first fingerprints are obtained from condensers and conductors, which have been studied in the theory of pseudo-randomness. We introduce the specific type of conductor that we use, after which we present an overview of the proof of Theorem 2.1.

Condensers have been introduced in various studies of extractors [Raz *et al.* 1999; Raz & Reingold 1999; Ta-Shma *et al.* 2007], and conductors were explicitly introduced by Capalbo et al. [Capalbo *et al.* 2002], under the name *simple conductors*. The following closely related variant is tailored to our purposes. Let P be the probability measure of a random variable Y with finite range $\mathcal{Y}$. The *γ-excess* of Y is

$$\sum_{y \in \mathcal{Y}} \max\{0, P(y) - \gamma\} \,.$$

For a set S, let U_S be the random variable that is uniformly distributed in S.

Definition 2.2. *A probabilistic function $\mathrm{F}\colon \mathcal{X} \to \mathcal{Y}$ is a $(K \xrightarrow{\varepsilon} K')$-*condenser *if for every set $S \subseteq \mathcal{X}$ of size $\#S = K$, the $(1/K')$-excess of $\mathrm{F}(U_S)$ is at most ε. F is a (K, ε)-*conductor *if it is a $(K' \xrightarrow{\varepsilon} K')$-condenser for all $K' \leq K$.*

[3] An idea would be to improve Lemma 2.2 by computing fingerprints with fewer random bits. A possibility would be to use randomness generated by a pseudo-random generator. In this case, one can prove a weaker variant of Theorem 1.1 where one is restricted to decompressors $\mathcal{D}$ running in polynomial time. Moreover, the statement is valid only conditional on a hardness assumption from computational complexity.
Another possibility would be to use Newman's theorem from communication complexity, that converts protocols with shared randomness into protocols with private randomness. In this way, we can only achieve invertible functions for a fixed set S. This means that we obtain a variant of Theorem 1.1 where the optimality requirement considers only a single decompressor. Even if this decompressor runs in polynomial time, the resulting compressor runs in exponential time. If the decompressor is optimal, as in Corollary 1.2, the resulting compressor is not even computable.

Equivalently, F is a (K, ε)-conductor if for every set S of size at most K, the $(1/\#S)$-excess is at most ε. The equivalence with standard definitions in the literature is discussed in appendix B. The following lemma shows that every invertible function is a conductor. It is used in the proof of Proposition 1.4.

Lemma 2.6. *Every (K, ε)-invertible function is a (K, ε)-conductor.*

Proof. Let $\mathrm{F}\colon \mathcal{X} \to \mathcal{Y}$ be (K, ε)-invertible. For any $S \in \mathcal{X}^{\leq K}$ and $x \in S$, we have $g_S(\mathrm{F}(x)) = x$ with probability at least $1 - \varepsilon$. Thus $g_S(\mathrm{F}(U_S)) \neq U_S$ with probability at most ε.

Let P be the measure of $g_S(\mathrm{F}(U_S))$. We show that this variable has $(1/\#S)$-excess at most ε. If we subtract from $P(x)$ the probability that $g_S(\mathrm{F}(x)) \neq x$, the resulting semimeasure has $(1/\#S)$-excess 0. Hence, $g_S(\mathrm{F}(U_S))$ has $(1/\#S)$-excess at most ε.

Note that after applying a deterministic function, the excess can only increase. Since $y \mapsto g_S(y)$ is deterministic, also $\mathrm{F}(U_S)$ has excess at most ε. □

Remarkably, the relation in the other direction also holds true: from conductors we obtain invertible functions. This is the content of Theorem 2.1. We present a sketch of the proof. The main observation is that conductors produce fingerprints with some relaxed properties, but which are still good enough for decompression. The first relaxation is that we do not need to insist on a fingerprint that produces zero collisions as in Lemma 2.5. It is sufficient if, for each list of suspects S, a string has a random fingerprint that causes at most t collisions with other elements in S, for t polynomial or quasi-polynomial in n. Such a fingerprint is called a *light* fingerprint. This works, because to the light fingerprint, we simply append a second hash code based on prime numbers. By Lemma 2.3, its size is $O(\log(tn/\varepsilon))$, and this constitutes the overhead. Then the string can be isolated from the t collisions, and be reconstructed. The second relaxation is that we do not require that all the elements of S have light fingerprints, but only at least half of them. More precisely, we say that a string x is *deficient* if with probability $\varepsilon/2$, the fingerprint $p = \mathrm{F}(x)$ is not light, where F is the conductor that we use as a hash function that produces fingerprints. Then, as we explain in the next paragraph, it suffices if at most half of the elements in S are deficient, and a conductor F indeed has this property.

Let x be the string that we want to compress and let $p = \mathrm{F}(x)$ be a random fingerprint of x produced by the conductor F. We now sketch the decompression procedure which reconstructs x from p and the prime-based hash code. Assume S is a valid set of suspects, i.e., $x \in S$. Initially, we collect the first t strings in S that have p as a fingerprint. If x is non-deficient, x will be among the collected strings with high probability. But x may very well be deficient, and, in this case, x needs to be reanalyzed at a later stage. So, we form a smaller set $\mathrm{R}(S)$ with all the deficient strings, and the decompressor is applied recursively to $\mathrm{R}(S)$, the new list of suspects. Since each recursive call decreases the set of suspects by half, the recursion has at most $\log(2\#S)$ levels, and we collect at most $t \log(2\#S)$ strings at all the levels of recursion. Now, the second prime-based hash code will allow us to distinguish x among the collected strings and reconstruct it. The details are presented in the next section, where we prove Theorem 2.1.

We next present the condensers and conductors that we use in the proofs. In our construction of $(2^k, \varepsilon)$-invertible functions, the difference between the output size and k is proportional to the length of the second hash code, which is in turn proportional to the number of random bits used in the evaluation of the conductor. In the pseudo-randomness literature, this is called the *seed length*, and a $\Omega(\log \frac{n}{\varepsilon})$ lower bound has been proven, see [Nisan & Zuckerman 1996; Radhakrishnan & Ta-Shma 2000] and Proposition 4.3 below. By a standard construction using the probabilistic method, there exist condensers that can be evaluated with $O(\log \frac{n}{\varepsilon})$ random bits. They are computable, but unfortunately not computable in polynomial time.

Proposition 2.7. *For all ε, n and k, there exists a $(2^k, \varepsilon)$-conductor $\mathrm{F}\colon \{0,1\}^n \to \{0,1\}^{k+2}$ that uses $\log \frac{4n}{\varepsilon}$ bits of randomness.*

Remark. Because conductors are finite objects, they can be computably constructed by exhaustive search. On input ε, k and n, this search can be done in space exponential in n.

The result of Theorem 2.1 considers a polynomial time construction. A family of condensers or conductors is *explicit* if there exists a probabilistic algorithm that on every input, consisting of ε, k and an n-bit x, outputs $\mathrm{F}^{(n)}_{\varepsilon,k}(x)$ in polynomial time.

We obtain explicit families of conductors from known constructions of extractors. Based on a result of Raz, Reingold, and Vadhan [Raz *et al.* 2002], Capalbo et al. [Capalbo *et al.* 2002] show that there exists an explicit conductor that uses $r = O(\log k \cdot \log^2 \frac{n}{\varepsilon})$ random bits. In section D, we obtain from explicit extractors given by Guruswami, Umans, and Vadhan [Guruswami *et al.* 2009], an improved explicit conductor that uses less randomness.

Proposition 2.8. *There exists an explicit family $\mathrm{F}^{(n)}_{\varepsilon,k}$ of $(2^k, \varepsilon)$-conductors $\mathrm{F}^{(n)}_{\varepsilon,k}\colon \{0,1\}^n \to \{0,1\}^k$ that uses $O(\log k \cdot \log \frac{n}{\varepsilon})$ random bits, for all ε, k and n.*

We also present a closely related, but technically less cumbersome approach, which does not use conductors, but instead obtains invertible functions by combining a few condensers given by the following result.

Theorem 2.9 ([Guruswami *et al.* 2009], Theorem 1.5 or 4.17). *For all n, κ and ε, there is an explicit family of $(2^{2\kappa} \xrightarrow{\varepsilon} 2^\kappa)$-condensers $\mathrm{F}\colon \{0,1\}^n \to \{0,1\}^\kappa$ whose evaluation requires $O(\log \frac{n}{\varepsilon})$ bits of randomness.*

Remark. The cited theorem in [Guruswami *et al.* 2009] gives an extractor with output length $m \geq k/2$. We can set $m = \mathrm{ceil}(k/2)$, because in extractors, we can reduce the output size by merging equal numbers of outputs, and this does not increase the statistical distance to the uniform measure. Finally, we obtain a condenser, since a function is a $(2^{2\kappa}, \varepsilon)$-extractor if and only if it is a $(2^{2\kappa} \xrightarrow{\varepsilon} 2^\kappa)$-condenser, see Remark 5 in appendix B.

Proof of Proposition 1.3 and Theorem 2.1

The following corresponds to the weak invertibility requirement in the above proof sketch.

Definition. $\mathrm{F}\colon \mathcal{X} \to \mathcal{Y}$ *is online (K, t, ε)-*list invertible *with T* rejections *if there exist deterministic monotone functions $\mathrm{R}\colon \mathcal{X}^{\leq K} \to \mathcal{X}^{\leq T}$ and $\mathrm{G}\colon \mathcal{X}^{\leq K} \times \mathcal{Y} \to \mathcal{X}^{\leq t}$ such that for all $S \in \mathcal{X}^{\leq K}$ and all $x \in S \setminus \mathrm{R}(S)$:*

$$\Pr\left[x \in \mathrm{G}(S, \mathrm{F}(x))\right] \;\geq\; 1 - \varepsilon.$$

The interpretation is that G is the "pruning" function, which on inputs S and y, reduces the list of K suspects to a smaller list of at most t elements. For $y = F(x)$, this short list should contain x, provided that $x \in S$ and x does not belong to the set $\mathrm{R}(S)$ of at most T "deficient elements." R is the "reanalyze" function that determines the elements in S that can be incorrectly lost in the pruning, and need to be reanalyzed again.

Lemma 2.10. *Let a be an integer. Every $(K \xrightarrow{\varepsilon} \frac{1}{a}K)$-condenser that can be evaluated with r random bits is online $(K, a2^r, 2\varepsilon)$-list invertible with $K/2$ rejections.*

In this section we use the lemma with $a = 1$.

Proof. For $S \in \mathcal{X}^{\leq K}$ and $y \in \mathcal{Y}$, let $\mathrm{G}(S, y)$ be the list containing the first $a2^r$ appearances of elements x in S for which $\mathrm{F}(x) = y$ has positive probability (or all appearances, if there are less than $a2^r$ of them).

Note that G is monotone in S. We assume that S has size exactly K, since the invertibility conditions only become easier to prove for smaller S. For a list L, let U_L denote the random variable obtained by selecting a random element in L. Observation:

> *If L' is the sublist of some $L \in \mathcal{Y}^B$ obtained by retaining the first b appearances of each element $y \in L$, then $\Pr[U_L \notin L']$ is at most the (b/B)-excess of U_L.*

We show that

$$\Pr\left[U_S \notin \mathrm{G}(S, \mathrm{F}(U_S))\right] \;\leq\; \varepsilon.$$

Let L be the list of length $2^r K$ of the values $\mathrm{F}_\rho(x)$ for all $x \in S$ and all assignments ρ of r random bits in $\mathrm{F}(x)$. More precisely, for $S = [x_1, \ldots, x_K]$, concatenate $[\mathrm{F}_{0^r}(x_i), \ldots, \mathrm{F}_{1^r}(x_i)]$ for increasing values of i. Note that U_L has the same distribution as $\mathrm{F}(U_S)$, which by definition of condenser, has $\frac{a}{K}$-excess at most ε. The procedure for G defines a sublist L' of L, containing all first $a2^r$ appearances of some $y \in \mathcal{Y}$. By choice of L and L', the events $U_L \notin L'$ and $U_S \notin \mathrm{G}(S, \mathrm{F}(U_S))$ have precisely the same probability. The inequality follows by applying the above observation, with $b = a2^r, B = K2^r$.

The function R is obtained by selecting the elements $x \in S$ for which $\Pr\left[x \notin \mathrm{G}(S, \mathrm{F}(x))\right] > 2\varepsilon$. R is monotone, because after adding an element to S, these probabilities for its other elements do not change. The proof finishes by showing that $\mathrm{R}(S)$ contains at most $K/2$ elements, i.e., $\Pr[U_S \in \mathrm{R}(S)] \leq 1/2$. This follows from

$$2\varepsilon \cdot \Pr[U_S \in \mathrm{R}(S)] \;\leq\; \Pr[U_S \notin \mathrm{G}(S, \mathrm{F}(U_S))] \;\leq\; \varepsilon. \qquad \square$$

Remark. If the function $\mathrm{F}\colon \{0,1\}^n \to \mathcal{Y}$ can be evaluated in space polynomial in n, then also the functions R and G constructed in the proof above can be evaluated in space polynomial in n.

Corollary 2.11. *If* F *is a (M, ε)-conductor that can be evaluated with r random bits, then* F *is online $(M, 2^r \log(2M), 2\varepsilon)$-list invertible (with 0 rejections).*

Proof. Let G and R be the functions defined above for $a = 1$. Note that we have $\#\mathrm{R}(S) \leq \frac{1}{2}\#S$ for all S, since the algorithms do not depend on K, and the assumption holds for all $K \leq M$, by definition of conductors.

For $S \in \mathcal{X}^{\leq M}$ and $y \in \mathcal{Y}$, the inverse G′ that satisfies the conditions is defined recursively. If S is empty, then $G'(S, y)$ is empty. Otherwise, $G'(S, y)$ is the concatenation of $\mathrm{G}(S, y)$ and $\mathrm{G}'(\mathrm{R}(S), y)$.

Each recursive call adds at most 2^r elements, and at most $\log(2M)$ recursive calls are made. The probability that $x \notin \mathrm{G}'(S, \mathrm{F}(x))$ is at most 2ε. Indeed, if $x \notin \mathrm{R}(S)$, this follows from Lemma 2.10, and otherwise, this follows by an inductive argument on the size of S. $\square$

Remark 3. If the function $\mathrm{F}\colon \{0,1\}^n \to \mathcal{Y}$ can be evaluated in space polynomial in n, then also the functions R′ and G′ constructed in the proof above can be evaluated in space polynomial in n. Indeed, these functions operate on sets of n-bit strings and can be of exponential size. But, to evaluate the functions $\mathrm{R}(S)$ and $\mathrm{G}(S, y)$, one does not need to store the full set S, but only needs to iterate over elements in S. This implies that the space needed to evaluate $\mathrm{G}'(S, y)$ equals the recursion depth times the space needed to iterate over $\mathrm{R}(\cdot)$ and $\mathrm{G}(\cdot, y)$, and this is polynomial in n.

The next result follows almost directly from the definitions.

Lemma 2.12. *If* F *is (K, t, ε)-list invertible with T rejections,* F′ *is $(t + T, \varepsilon')$-invertible, and both functions have the same domain, then $x \mapsto (\mathrm{F}(x), \mathrm{F}'(x))$ is $(K, \varepsilon + \varepsilon')$-invertible.*

The next result follows directly from this lemma and Corollary 2.11.

Corollary 2.13. *If* $\mathrm{F}\colon \{0,1\}^n \mapsto \mathcal{Y}$ *is a* (K,ε)*-conductor that can be evaluated using at most* r *random bits, then the function*

$$x \longmapsto (\mathrm{F}(x), \mathrm{H}_{\varepsilon,s}(x)) \quad \text{with} \quad s = 2^r \log(2K)$$

is $(K, 3\varepsilon)$*-invertible, where* $\mathrm{H}_{\varepsilon,s}$ *is the prime-based hash function from Lemma 2.4.*

Proof of Theorem 2.1. We apply Corollary 2.13 to the conductor given in Proposition 2.8. The result is a family of online $(2^k, \varepsilon)$-invertible functions where $\mathrm{F}\colon \{0,1\}^n \to \{0,1\}^m$ with $m - k \le O(\log k \cdot \log \frac{n}{\varepsilon})$. By Remark 3, the inverse can be evaluated in space polynomial in n. Finally, by Remark 1, we obtain an inverse function over the set of strings of all lengths. □

Proof of Proposition 1.3. We apply Corollary 2.13 to the conductor given in Proposition 2.7. The result is a family of online (K,ε)-invertible functions where $\#\mathcal{Y}/K$ is polynomial in n/ε. Proposition 1.3 follows by converting invertible functions to compressors, as explained in the proof of Theorem 1.1 assuming Theorem 2.1 presented at the beginning of section 2. □

Proof of Theorem 2.1 based on condensers

The proof of Theorem 2.1 given in the previous section requires the explicit conductor from Proposition 2.8, whose proof is rather technical. In this subsection, we give an easier and more direct proof that is based only on condensers, bypassing conductors. The two proofs are similar, still, they each have their own advantage: if explicit conductors are discovered that use less randomness, then this leads to improved compressors through the first proof. On the other hand, if explicit condensers are found that use logarithmic randomness and extract more of their minentropy, then the approach of this section leads to improved compressors. The following is a consequence of Lemma 2.10.

Lemma 2.14. *Let* a *and* b *be integers. If* F *is a* $(M \xrightarrow{\varepsilon} \frac{1}{a}M)$*-condenser that can be evaluated with* r *random bits, then* F *is online* $(bM, ab2^r \log(2b), 2\varepsilon)$*-list invertible with* $M/2$ *rejections.*

Proof. The pruning and reanalyze functions G' and R' are defined recursively in b. Given $S \in \mathcal{X}^{\le bM}$, partition S in b sublists S' of length at most M, and apply the functions G and R from Lemma 2.10. Collect the to-be-reanalyzed strings from all $\mathrm{R}(S')$'s in a set S_{rec}, and the selected strings from all sets $\mathrm{G}(S', y)$ in a set S_{sel}. If $\#S_{\mathrm{rec}} \le M/2$ we are done, and we let $\mathrm{R}'(S) = S_{\mathrm{rec}}$ and $\mathrm{G}'(S, y) = S_{\mathrm{sel}}$. Otherwise, we continue recursively by letting $\mathrm{G}'(S, y)$ be the concatenation of S_{sel} and $\mathrm{G}'(S_{\mathrm{rec}}, y)$, and letting $\mathrm{R}'(S) = \mathrm{R}'(S_{\mathrm{rec}})$.

We show that the algorithm works correctly. By construction, R' outputs the required number of elements. In each recursive call, $\#S_{\mathrm{rec}}$ is at least halved. Thus, the number of recursive calls is at most $\log(2b)$. In each call, the size of S_{sel} is at most $b \cdot (a2^r)$, and hence this number of elements is appended to G'. Thus, this function outputs a list of size at most $ab2^r \log(2b)$. □

Using Lemma 2.12, this implies the following for $a = b = K$ and $M = K^2$.

Lemma 2.15. *If* F *is a* $(K^2 \xrightarrow{\varepsilon} K)$*-condenser that can be evaluated with* r *random bits,* F' *is* $(K^2 2^r \log(4K), \varepsilon')$*-invertible, and both functions have the same domain, then* $x \mapsto (\mathrm{F}(x), \mathrm{F}'(x))$ *is* $(K^3, 2\varepsilon + \varepsilon')$*-invertible.*

We apply this lemma to the following condensers given by Guruswami, Umans, and Vadhan [Guruswami *et al.* 2009].

Theorem (Restated). *For all n, κ and ε, there is an explicit family of $(2^{2\kappa} \xrightarrow{\varepsilon} 2^{\kappa})$-condensers $\mathrm{F}\colon \{0,1\}^n \to \{0,1\}^\kappa$ whose evaluation requires $O(\log \frac{n}{\varepsilon})$ bits of randomness.*

Proof of Theorem 2.1. By Remark 1, it suffices to present the construction for inputs of a fixed length n. If $k \geq n$, then let F be the identity function. Otherwise, we apply Lemma 2.15 recursively for decreasing k. In other words, we concatenate $O(\log k)$ condensers with geometrically decreasing values of κ.

We present the details. Let r denote the $O(\log \frac{n}{\varepsilon})$ bound on the randomness in the condenser of Theorem 2.9. Let b be such that $2^b \geq 2^r \log(4K)$ for all $K \leq 2^n$, but still satisfies $b \leq O(\log \frac{n}{\varepsilon})$.

If $k < 100 \cdot b$, we obtain an $(2^k, \varepsilon)$-invertible function using prime hashing from Lemma 2.4. Otherwise, the invertible function is obtained by concatenating the condenser for $\kappa = \mathrm{ceil}(k/3)$ and the recursive application of the construction for $k \leftarrow 2\kappa + b$.

At most $d \leq O(\log k)$ recursive calls are made, because if $k \geq 100 \cdot b$, the recursive value for k is close to $\frac{2}{3}k$, say at most $\frac{5}{6}k$. We verify that if the recursion depth is d, then we obtain a $(2^k, 2d\varepsilon)$-invertible function with output length $k + bd$. Indeed, in each concatenation, the error given by Lemma 2.15 increases with 2ε, and the output length increases with κ, while the value of k increases by $\kappa - b$. This proves the first part of the theorem. The moreover part follows from similar observations as in Remark 3. □

3 OPTIMAL DISTRIBUTED COMPRESSION: PROOF OF THEOREM 1.5

Observe that if $\mathrm{F}_1\colon \mathcal{X} \to \mathcal{Y}_1$ and $\mathrm{F}_2\colon \mathcal{X} \to \mathcal{Y}_2$ are (K_1, ε_1) and (K_2, ε_2)-invertible, then $x \mapsto (\mathrm{F}_1(x), \mathrm{F}_2(x))$ is $(K_1K_2, \varepsilon_1 + \varepsilon_2)$-invertible. We can not apply this observation to the setting of distributed compression. Indeed, if we choose $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2$, and consider a function F_1 acting on the left and F_2 on the right coordinate, then F_1 can not be invertible on $\mathcal{X}$, because it can not distinguish tuples with the same right coordinate. The following proposition obtains invertible functions suitable for distributed compression with 2 sources.

Proposition 3.1. *If $\mathrm{F}_1\colon \mathcal{X}_1 \to \mathcal{Y}_1$ is (K_1, ε)-invertible and $\mathrm{F}_2\colon \mathcal{X}_2 \to \mathcal{Y}_2$ is (K_2, ε)-invertible, then $(x_1, x_2) \mapsto (\mathrm{F}_1(x_1), \mathrm{F}_2(x_2))$ is (3ε)-invertible*[4] *in sets $S \subseteq \mathcal{X}_1 \times \mathcal{X}_2$ for which*

$$\begin{aligned} \#S &\leq K_1K_2, \\ \#\{(z_1, z_2) \in S : z_2 = x_2\} &\leq K_1 \qquad \forall x_2 \in \mathcal{X}_2, \\ \#\{(z_1, z_2) \in S : z_1 = x_1\} &\leq K_2 \qquad \forall x_1 \in \mathcal{X}_1. \end{aligned}$$

The proof is given in appendix E. For the proof of Theorem 1.5, we need an online variant of this proposition for an arbitrary number of invertible functions. Its statement requires some more definitions.

Definition 3.1. *Let $S \subseteq \mathcal{X}_1 \cdots \mathcal{X}_\ell$ and $\mathrm{K} = (K_1 \ldots K_\ell)$ be an ℓ-tuple of integers. S has* K-small slices *if for all $\mathrm{x} \in S$ and for all $J \subseteq [\ell]$:*

$$\#\{\mathrm{z} \in S : \mathrm{z}_J = \mathrm{x}_J\} \leq \prod_{j \in [\ell] \setminus J} K_j.$$

Let $\mathcal{X}^{<\omega}$ denote the set of finite sequences of elements in $\mathcal{X}$.

[4] We say that F is invertible in a set S if there exists a function h such that $\Pr[h(\mathrm{F}(x)) = x] \geq 1 - \varepsilon$ for all $x \in S$.

Definition 3.2. *Let $\mathcal{S}$ be a collection of subsets of $\mathcal{X}$.* F *is ε-invertible* on $\mathcal{S}$ *if there exists a mapping $g\colon \mathcal{X}^{<\omega} \times \mathcal{Y} \to \mathcal{X}$, such that for all S, whose elements form a set in $\mathcal{S}$*

$$\Pr\left[g_S(\mathrm{F}(x)) = x\right] \;\geq\; 1 - \varepsilon\,,$$

F *is* online *ε-invertible on $\mathcal{S}$ if there exists a monotone such g, as in Definition 2.1.*

Theorem 1.5 follows from the proof of the following.

Theorem 3.2. *If $\mathrm{F}_j\colon \mathcal{X}_j \to \mathcal{Y}_j$ is online $(\frac{\ell}{\varepsilon}K_j, \varepsilon)$-invertible for all $j \in [\ell]$, then the function*

$$(x_1, \ldots, x_\ell) \;\longmapsto\; (\mathrm{F}_1(x_1), \ldots, \mathrm{F}_\ell(x_\ell))$$

is online $(8\ell\varepsilon)$-invertible on the collection of sets with K*-small slices.*

Remark. The offline variant also holds, in which we start with a weaker assumption and obtain a weaker conclusion.

The proof of Proposition 3.1 can be modified for the online version and extended to arbitrary ℓ. Moreover, compared to Theorem 3.2 it is easier, and provides slightly better parameters when ℓ and ε are small. Unfortunately, the obtained error is $\ell 2^\ell \varepsilon$, which can be much larger than $8\ell\varepsilon$.

The difficulty in proving both the proposition and the theorem is to partition the large set S of initial suspects into smaller subsets on which the inverses of the functions F_j can be applied. For the proposition, this partition is deterministic and rather intuitive. For the theorem, we use a different and novel technique, called *randomized-tree partition*, and this allows to reduce the error to $O(\varepsilon\ell)$.

Proof of Theorem 3.2

We construct a probabilistic algorithm g that on input $\mathrm{y} \in \mathcal{Y}$ and a list S of tuples in $\mathcal{X}_1 \times \cdots \times \mathcal{X}_\ell$ with K-small slices, selects an element $g(S, \mathrm{y})$ from S such that for each fixed $\mathrm{x} \in S$:

$$\Pr\left[g(S, \mathrm{F}(\mathrm{x})) \neq \mathrm{x}\right] \;\leq\; (\exp(1) + \ell)\varepsilon, \qquad (*)$$

in which the probability is over the random choices of both g and F. The algorithm operates in an *online* way for the input S. This means that g starts by executing an initialization procedure that does not depend on S. Next, for each subsequent element x in S, an update procedure is run that manipulates some variables. This update procedure might decide to assign the output $g(S, \mathrm{y}) = \mathrm{x}$. Once, committed to an output, this output may no longer be changed. In this way, the obtained function g is monotone.

A deterministic online inverse is obtained by taking majority votes of the probabilistic outcome of g. This at most doubles the error. Indeed, given a set S, a value y is *bad* for some $\mathrm{x} \in S$ if $g(S, \mathrm{y}) = \mathrm{x}$ has probability at most $1/2$. By the inequality above, the output $\mathrm{F}(\mathrm{x})$ is bad for x with probability at most $2 \cdot (\exp(1) + \ell)\varepsilon \leq 8\ell\varepsilon$, and if $\mathrm{F}(\mathrm{x})$ is not bad, then the majority vote returns x. Hence, the derandomized g satisfies the invertibility conditions of the theorem, where the correspondence with the notation from the definition is obtained by $g_S(\mathrm{y}) = g(S, \mathrm{y})$.

The initialization algorithm for g creates a random tree. It starts by creating the root, which has depth 0. Afterwards, for $j = 1, \ldots, \ell$ and for each node at depth $j - 1$, it creates $\frac{\ell}{\varepsilon}K_j$ children and associates each element in $\mathcal{X}_j$ to a random child. After the random tree is created, the initialization procedure finishes.

Recall that the update procedure for $g(S, \mathrm{y})$ is run when a new element x is added to S. This procedure is deterministic and places copies of x on some nodes of the tree. We can view the copies of x as pebbles labeled by x. Elements are

never removed from nodes, and we ensure that at each moment, there is at most one element on a node. The update procedure proceeds in two steps, which we call *top-down percolation* and *bubble up*.

The top-down percolation step.
Place x on a leaf of the tree by descending it in the natural way: for moving down from depth $j-1$ to depth j, select the child node associated to x_j. If the leaf already contains an element, then x is not placed on any leaf, and the update procedure terminates without selecting x for the output.

Before presenting the second step, we first bound the probability that a fixed x in S is not placed on a leaf during the percolation step.

Lemma 3.3. *Assume S has* K*-small slices and that* x *belongs to S. The probability that* x *is not placed on a leaf is at most* $\varepsilon \exp(1)$.

Proof. We bound the probability that some tuple z in S, different from x, descends to the same leaf as x. If this is not the case, than x is indeed placed on a leaf.

Let J be a non-empty subset of $[\ell]$ and let z be an ℓ-tuple for which $z_j \neq x_j$ for all $j \in J$, and $z_j = x_j$ for all $j \in [\ell] \setminus J$. The probability that z is placed in the same leaf as x is at most

$$\prod_{j\in J} \frac{\varepsilon}{\ell K_j}.$$

By the slice assumption, the set S contains at most $\prod_{j\in J} K_j$ such elements z, and hence, the probability that this happens for such a z is at most

$$(\varepsilon/\ell)^{\#J} \;\leq\; \varepsilon(1/\ell)^{\#J},$$

since J is nonempty. By the union bound over all nonempty subsets $J \subseteq [\ell]$, the probability that some element of S is placed in x'es leaf is at most

$$\leq \quad \varepsilon \sum_J \left(\frac{1}{\ell}\right)^{\#J} \quad \leq \quad \varepsilon \left(1+\frac{1}{\ell}\right)^{\ell} \quad \leq \quad \varepsilon \exp(1). \qquad \square$$

The second step of the update procedure is the *bubble up* step. In this step, the element x assigned to some leaf, attempts to be copied upwards to the root. There is a "competition," and out of the elements placed on the children of a node, at most one progresses up. If x reaches the root, then it is the output of $g(S, \mathrm{y})$. Let g_j be the online function that inverts F_j.

The bubble up step.
Repeat the rounds that we describe next until termination. At the current round, execute the following four steps:

- Consider the node with minimal depth that contains x and let j be this depth (so, at the first round, j is ℓ).
- Let B_j be the set of j-th coordinates of tuples placed on the siblings of this node. We view these tuples as competing to create a copy of themselves in the parent.
- If $g_j(B_j, y_j) = x_j$, place a copy of x on the parent node (in other words, x is the winner). Otherwise, terminate without selecting x.
- If x is placed in the root, terminate and select x for the output.

Remarks.

The above loop always terminates, because if termination does not happen in the third step of a round, x is raised up in the tree and the algorithm continues with the next round. Therefore, if termination does not happen in a third step of any round, x reaches the root and the algorithm terminates in the fourth step.

As promised above, in every node, at most one tuple is placed. Indeed, for leaves this follows by the construction in the percolation step. For the other nodes, this follows by the definition of online inverses g_j, because if $g_j(B_j, y_j)$ has committed to an output x_j, then further additions of elements to B_j will not change this output. By the construction of the tree, all siblings in the third step contain tuples with different j-th coordinates, and therefore only at most one can win.

When the algorithm commits to an output, this answers remains. Indeed, by the previous step, at most one tuple is placed in the root, and after being placed, it is never removed.

We prove (*). For a tuple $\mathrm{x} \in S$, consider the following events:
- $\mathcal{E}_0$ is the event that x is assigned to a leaf of the tree at the percolation step.
- For each $j \in [\ell]$, consider the node at depth j on the branch of x. Let B_j be the set of j-th coordinates of tuples placed in the node's siblings after processing all elements of S. $\mathcal{E}_j$ is the event that $g_j(B_j, F_j(x_j)) = x_j$.

Conditioned on all events $\mathcal{E}_0$ and $\mathcal{E}_\ell, \dots, \mathcal{E}_1$ being true, x will appear in the root and, thus, is the output of $g(S, \mathrm{F}(\mathrm{x}))$, in other words, g has correctly inverted F(x). Indeed, by $\mathcal{E}_0$, x is placed to some leaf during the percolation step. And by $\mathcal{E}_j$, at each depth j in the bubble up step, the tuple x wins the competition and is placed on its parent, since, no other sibling has the same j-th coordinate.

We now bound the error probability that at least one of the above events does not happen. By Lemma 3.3, the probability that x is not placed on a leaf at the percolation phase is at most $\varepsilon \exp(1)$. Otherwise, x is placed to some leaf. Consider its branch. Note that the size of B_ℓ is at most $\frac{\ell}{\varepsilon} K_\ell$, because this is the number of sibling nodes, and each node contains at most one tuple. By the $(\frac{\ell}{\varepsilon} K_\ell, \varepsilon)$-invertibility of F_ℓ, the value of $g_\ell(B_\ell, \mathrm{F}_\ell(\mathrm{x}_\ell))$ differs from x_ℓ with probability at most ε. Otherwise, if this does not happen, then x is placed in the branch at depth $\ell - 1$, and $x_{\ell-1}$ belongs to $B_{\ell-1}$. By induction, the same argument is valid for all levels, and thus every event $\mathcal{E}_j$ fails with probability at most ε.

The probability that one of the events does not happen is bounded by the sum of all mentioned probabilities, which is precisely the right-hand side of (*). The theorem is proven.

Proof of Theorem 1.5.

We first connect the Slepian-Wolf constraints from Definition 1.2 to K-small slices.

Lemma 3.4. *The set S of tuples* x *such that* k *satisfies the $\mathcal{D}$-Slepian-Wolf constraints for* x *has $(2^{k_1}, \dots, 2^{k_\ell})$-small slices.*

Proof. If $J = [\ell]$, the set in Definition 3.1 equals $\{\mathrm{x}\}$ and the inequality is true. Fix a strict subset $J \subset [\ell]$. All elements in $\mathrm{z} \in S$ satisfy

$$\mathrm{C}_{\mathcal{D}}(\mathrm{z}_{[\ell]\setminus J} \mid \mathrm{z}_J) \;<\; \sum_{j \in [\ell]\setminus J} k_j.$$

The number of such ℓ-tuples z with $\mathrm{z}_J = \mathrm{x}_J$ is therefore bounded by the exponent of the right hand side. Hence, this set has the required size. □

Recall that in the proof of Theorem 1.1, we constructed a compressor using a family $\mathrm{F}_{\varepsilon,\kappa}$ of $(2^\kappa, \varepsilon)$-invertible functions on $\{0,1\}^*$. For this, we simply appended the parameters ε and κ to the output. Note that for the obtained overhead Δ, the output length is equal to $m = \kappa + \Delta - 1$. For such compressors, Theorem 1.5 follows easily from Theorem 3.2.

Proof of Theorem 1.5 for compressors that have the parameter κ in their output. Let ε and $y = (y_1, \dots, y_\ell)$ be inputs for the decompressor $\mathcal{D}'$ that we need to construct. By the assumption, each y_j provides us with the value κ_j, and therefore specifies a monotone $(2^{\kappa_j}, \varepsilon)$-inverse g_j. Let g be the inverse constructed in the proof of Theorem 3.2. This construction only uses objects that are specified by the inputs ε and y of $\mathcal{D}'$: ε, ℓ, κ_j and g_j for all j. Let S be the set of strings for which $(\kappa_1 - \log \frac{\ell}{\varepsilon}, \dots, \kappa_\ell - \log \frac{\ell}{\varepsilon})$ satisfies the $\mathcal{D}$-Slepian-Wolf constraints. Define $\mathcal{D}'(\varepsilon, y) = g(S, y)$.

If x satisfies the $(m_1 - \Delta_1 - \log \frac{\ell}{\varepsilon}, \dots, m_\ell - \Delta_\ell - \log \frac{\ell}{\varepsilon})$-Slepian-Wolf conditions, then $x \in S$ by choice of S, (there is even 1 bit surplus). By Lemma 3.4, S has $(\frac{\varepsilon}{\ell} 2^{\kappa_1}, \dots, \frac{\varepsilon}{\ell} 2^{\kappa_\ell})$-small slices. By choice of $\mathcal{D}'$ and by Theorem 2.1, the event $\mathcal{D}'(\varepsilon, F(x)) = x$ has probability at least $1 - 8\varepsilon\ell$. This implies the theorem. □

For the general case, we need to adapt the proof of Theorem 3.2.

Proof of Theorem 1.5. Assume the compressor is Δ-optimal, and for inputs (ε, m, x), let $\Delta(\varepsilon, m, x)$ be the value of the overhead. Given a decompressor $\mathcal{D}$ and an ℓ-tuple m, we define the set S_m of tuples x that satisfy the conditions of the theorem:

$$C_{\mathcal{D}}(x_J \mid x_{[\ell]\setminus J}) \;\le\; \sum_{j \in J} \left(m_j - \Delta(\varepsilon, m_j, x_j) - \log \frac{\ell}{\varepsilon} \right) \qquad \text{for all nonempty } J \subseteq [\ell].$$

Let d be an ℓ-tuple of integers. The set of elements x in S_m for which $\Delta(\varepsilon, m_j, x_j) = d_j$ for all $j \in [\ell]$ has $(\frac{\varepsilon}{\ell} 2^{m_1 - d_1}, \dots, \frac{\varepsilon}{\ell} 2^{m_\ell - d_\ell})$-small slices.

Given the set S_m as a list, we need to specify an output for a decompressor $\mathcal{D}'$ for some inputs ε and y. We modify the tree-partitioning in the online algorithm of Theorem 3.2 as follows. For each depth $j \in [\ell]$, for each node at depth j, for each $d_j \le m_j$, we create $2^{m_j - d_j}$ children, and associate each string w for which $\Delta(\varepsilon, m_j, w) = d_j$ to a random such child.

We now consider the update procedure. The first step, the top-down percolation step, does not change. In the bubble up step, we only change the third instruction. For this instruction, we have no inverse function available. But instead, we construct a decompressor $\mathcal{E}$, and apply the decompressor $\mathcal{E}'$ obtained from the definition of Δ-optimality. This decompressor $\mathcal{E}$ assigns programs of length $m_j - d_j$ to all j-th coordinates of tuples of children with associated value d_j. (Recall that there are precisely $2^{m_j - d_j}$ such children.) Let $\mathcal{E}'$ be the corresponding decompressor obtained from Definition 1.1. If $\mathcal{E}'(y_j) = x_j$ then x is copied from the child to the parent.

The analysis of this algorithm follows the analysis above: by the same reasoning as in Lemma 3.3, for each fixed tuple d, there are no collisions in leafs of d-branches, i.e., branches whose nodes at depth j have corresponding values for d_j. Indeed, each factor $\frac{\varepsilon}{\ell} 2^{m_j - d_j}$ in the condition of $(\frac{\varepsilon}{\ell} 2^{m-d})$-small slices, is a factor $\frac{\varepsilon}{\ell}$ smaller than the number of branches, which is $2^{m_j - d_j}$. Finally, the probability that for a fixed $x \in S_m$, on input $\varepsilon, \mathcal{C}_{\varepsilon, m_1}(x_1), \dots, \mathcal{C}_{\varepsilon, m_\ell}(x_\ell)$, the tuple x bubbles up to the root, also follows the same analysis. □

Remark 4. Recall that in the single source case, the optimal compressor in the proof of Theorem 1.1, provides a modified decompressor that can be evaluated in space polynomial in $n_S = \max\{|z| : z \in S\}$, when given oracle access to the original decompressor. Given oracle access to a decompressor $\mathcal{D}$ acting on tuples, the above procedure defines a decompressor $\mathcal{D}'$ that can be evaluated in time exponential in ℓ and $n_S = \max\{|z_j| : z \in S \text{ and } j \in [\ell]\}$, and space polynomial in $n_S \cdot 2^\ell$. This implies that if $\mathcal{D}$ is partial computable, respectively, computable and computable in exponential time, then so is the corresponding $\mathcal{D}'$.

4 LOWER BOUNDS

In this section we prove the lower bounds on the overhead Δ and on the number of random bits r used by the compressors given in Proposition 1.4. To strengthen these lower bounds, we first show that Δ-optimality is equivalent to a weaker notion of optimality.

Robustness of the definition of optimal compressor

Probabilistic decompressors. We show that the definition of Δ-optimality does not change if we also consider probabilistic decompressors, except for a small rescaling of ε.

For a probabilistic partial function $\mathcal{D}$ let $\mathcal{D}_{\text{maj}}$ be the partial function that for each argument of $\mathcal{D}$ is undefined if every value appears with probability at most $1/2$, and otherwise, it equals the unique value that appears with probability strictly more than $1/2$.

Lemma 4.1. *For every probabilistic function* F*:*

$$\Pr[\mathcal{D}(\mathrm{F}(x)) \neq z] \;\geq\; \tfrac{1}{2} \cdot \Pr[\mathcal{D}_{\text{maj}}(\mathrm{F}(x)) \neq z].$$

Proof. We call a value $p = \mathrm{F}(x)$ *bad* if $\mathcal{D}(p) = z$ has probability at most $1/2$. The right-hand side is equal to the probability that $\mathrm{F}(x)$ generates a bad value. The inequality follows directly. □

A single optimal decompressor. Recall that a Turing machine $\mathcal{U}$ is *optimal* if for every other Turing machine M there exists a constant $c_{\mathcal{D}}$ such that for all strings x: $\mathrm{C}_{\mathcal{U}}(x) \leq \mathrm{C}_{\mathcal{D}}(x) + c_{\mathcal{D}}$. We show that if in stead of considering any decompressor in the definition of Δ-optimality, we only consider a single optimal decompressor, the definition does not change after a constant shift of Δ, provided Δ is a computable expression.

Lemma 4.2. *Assume that Δ is a computable function of the compressors inputs, $\mathcal{U}$ is an optimal Turing machine, and there exists a Turing machine $\mathcal{U}'$ such that for all ε, x and $m \geq C_{\mathcal{U}}(x) + \Delta$ we have*

$$\Pr\left[\mathcal{U}'(\mathcal{C}_{\varepsilon,m}(x)) = x\right] \geq 1 - \varepsilon.$$

Then $\mathcal{C}$ is $(\Delta + O(1))$-optimal.

Proof. We show the contrapositive: If for large c, compressor $\mathcal{C}$ is not $(\Delta + 2c)$-optimal, then no machine $\mathcal{U}'$ satisfies the condition of the lemma.

For each positive integer c, consider a decompressor $\mathcal{D}_c$ for which the $(\Delta + 2c)$-optimality condition does not hold, i.e., there exists no $\mathcal{D}'$ that satisfies the condition of Definition 1.1. By a compactness argument, there exists such a $\mathcal{D}_c$ with finite domain, and hence, on input c, such a decompressor can be found by exhaustive search. Note that this search requires the evaluating of Δ.

We construct a decompressor $\mathcal{D}$ by combining all these $\mathcal{D}_c$. More precisely, given an input of the form $0^c 1p$, the output of $\mathcal{D}$ is given by $\mathcal{D}_c(p)$. By optimality of $\mathcal{U}$, we have

$$C_{\mathcal{U}}(x) \;\leq\; C_{\mathcal{D}}(x) + c_{\mathcal{D}} \;\leq\; C_{\mathcal{D}_c}(x) + c_{\mathcal{D}} + c.$$

Let $c = c_{\mathcal{D}}$. The assumption $m \geq C_{\mathcal{U}}(x) + \Delta$ in the lemma implies the assumption $m \geq C_{\mathcal{D}}(x) + (\Delta + 2c_{\mathcal{D}})$ in the $(\Delta + 2c)$-optimality criterion for $\mathcal{D}_c$. Thus if there exists a machine $\mathcal{U}'$ satisfying the conditions of the lemma, then it gives us a machine $\mathcal{D}'$ satisfying the $(\Delta + 2c)$-optimality criterion. By choice of $\mathcal{D}_c$, such $\mathcal{D}'$ does not exist, and hence, $\mathcal{U}'$ also does not exist. The lemma is proven. □

Lower bound for the randomness used by optimal compressors

Proposition (Second part of Proposition 1.4). *Let Δ and r be functions of ε and $n = |x|$. If there exists a Δ-optimal compressor that can be evaluated with at most r random bits in a computably bounded running time, then for all $\varepsilon \le 1/2$:*

$$\lceil \varepsilon 2^{r+1} \rceil \;\ge\; \frac{n - r - \log(2/\varepsilon)}{\Delta + 4}$$

Recall that for a deterministic compressor we have $r = 0$. If such a compressor is Δ-optimal and has a computably bounded running time, then for $\varepsilon = 1/2$, this implies $\Delta \ge n - 6$. As a warm-up, let us prove in a direct way, a slightly stronger bound for deterministic compressors.

Lemma. *If a Δ-optimal compressor is deterministic, then $\Delta \ge n - 1$.*

Proof. We show the lemma by contraposition. Assume there exists an $(n-2)$-optimal compressor. There are less than 2^n different outputs of length $m = n - 1$. Thus given target m, there exist 2 different n-bit x and x' that are mapped to the same string. Consider a decompressor $\mathcal{D}$ for which $\mathrm{C}_{\mathcal{D}}(x), \mathrm{C}_{\mathcal{D}}(x') \le 1$, and hence, $m \ge \mathrm{C}_{\mathcal{D}}(x) + (n-2)$. By $(n-2)$-optimality, both x and x' must be decompressed correctly, but this is impossible since they are mapped to the same output. Hence, the compressor is not Δ-optimal. □

We now prove the second item of Proposition 1.4. Recall that Δ-optimal compressors define conductors, and hence condensers, with small output size, see Lemma 2.6.

In the pseudorandomness literature, condensers are usually defined as 2-argument functions $f : \mathcal{X} \times \mathcal{D} \to \mathcal{Y}$, where the size of $\mathcal{D}$ is called the *degree*, see also appendix B. For such f, the function $x \mapsto f(x, U_{\mathcal{D}})$ is a probabilistic 1-argument function as above. The number of random bits needed for its evaluation is $\log \#\mathcal{D}$, provided $\#\mathcal{D}$ is a power of 2.

We obtain lower bounds for the amount of randomness from degree lower bounds of condensers, such as given in [Nisan & Zuckerman 1996; Radhakrishnan & Ta-Shma 2000]. In particular, the second item of Proposition 1.4 follows by applying the next lower bound for a suitable value of K.

Proposition 4.3. *If $f : \mathcal{X} \times [D] \to \mathcal{Y}$ is a (K, ε)-condenser with $\#\mathcal{Y} \ge 2K \ge 4D/\varepsilon$ and $\varepsilon \le 1/2$, then*

$$\lceil 2\varepsilon D \rceil \;\ge\; \frac{\log(\#\mathcal{X}/K)}{3 + \log(\#\mathcal{Y}/K)}.$$

The proof is given in appendix F.

Proof of the second item of Proposition 1.4. If $\mathcal{C}$ is a Δ-optimal compressor, then for all n, m and ε the function $\mathrm{F} : \{0,1\}^n \to \{0,1\}^m$ defined by $\mathrm{F}(x) = \mathcal{C}_{\varepsilon,m}(x)$ is a $(2^{m-\Delta}, \varepsilon)$-conductor, and hence, a $(2^{m-\Delta}, \varepsilon)$-condenser. We apply the lower bound of Proposition 4.3 with $D = 2^r$ and $K = 2D/\varepsilon$, for the target size given by $m = \lceil \log K \rceil + \Delta$. Thus, we have $\mathcal{Y} = \{0,1\}^m$ and $\log(\#\mathcal{Y}/K) \le \Delta + 1$. The formula of Proposition 4.3 implies the inequality of the second item. □

Lower bound for the overhead of optimal compressors

The proof of the first item of Theorem 1.4, uses the second item and Lemma 4.5 below. This lemma essentially states that a simple modification of any invertible function can reduce its use of randomness to $r \le \Delta + O(1)$. In the proof we use another lemma.

Lemma 4.4. *Let b be a nonnegative integer. If a measure μ over $\mathcal{Y}$ is supported on a set of size at most 2^d, then there exists a sampling algorithm that uses $d + b$ random bits, and generates samples from $\mathcal{Y}$ such that each element $y \in \mathcal{Y}$ appears with probability at most $(1 + 3 \cdot 2^{-b})\mu(y)$.*

Proof. Let $D = 2^d$. The proof for arbitrary b is similar to the proof for $b = 0$, which we present here. We show that there exists a measure ν whose values are multiples of $1/D$ such that $\nu(y) \le 4\mu(y)$ for all $y \in \mathcal{Y}$. This is done by rounding. The sampling procedure satisfying the conditions of the lemma, simply outputs a random sample of ν, and we can generate such a sample using d random bits.[5]

Let $\mu'(y)$ be 0 if $\mu(y) < 1/2D$, and $\mu(y)$ otherwise. Note that $Z = \sum_y \mu'(y) \ge 1/2$ since μ is supported on at most D elements. μ'/Z is a measure. We obtain the measure ν by rounding all values of μ'/Z to multiples of $1/D$. By rounding up a suitable set of elements and rounding down the others, we indeed obtain a measure, because:
- if we round up all values, the sum of $\sum_y \nu(y)$ is at least 1,
- if we round down, the sum is at most 1, and
- changing the choice for one element y, changes the sum by a multiple of $1/D$.

We verify that $\nu(y) \le 4\mu(y)$. Recall that $1/Z \le 2$. If $\mu(y) \ge 1/(2D)$, then

$$\nu(y) \;\le\; \frac{\mu'(y)}{Z} + \frac{1}{D} \;\le\; 2 \cdot \mu(y) + 2 \cdot \mu(y). \qquad \square$$

Lemma 4.5. *Assume $\varepsilon \le 1/4$. For every (K, ε)-invertible function with at most M values, there exists a $(K, 2\varepsilon)$-invertible function with at most $M + K$ values that can be evaluated using at most $\lceil \log(M/K) \rceil + 3$ bits of randomness.*

Proof. Let F be (K, ε)-invertible with inverse g. For a given input x, consider the list of all possible outputs that appear with positive probability. Let P_x be the set of these outputs after removing a set of measure at most ε with maximal cardinality. We first prove the following 2 properties.

- Let S be a list of K inputs. For all $x \in S$: $\#P_x$ is at least the number of values y for which $g_S(y) = x$.
- There are less than K inputs x for which $\#P_x > M/K$.

For the first property, note that by definition of (K, ε)-inverse, for a given x, the measure of outcomes $y = \mathrm{F}(x)$ with $g_S(y) \ne x$ is at most ε. Thus, the property follows by choice of P_x.

For the second property, consider the set of inputs x for which the property holds. Assume there are at least K such inputs, and let S be a list of precisely K such inputs. By the first property, for each such element there are more than M/K outputs y with $g_S(y) = x$. But since there are only M outputs and g_S is deterministic, this is impossible. Our assumption must be false, and hence there are less than K such inputs. The 2 properties are proven.

Construction of the $(K, 2\varepsilon)$-invertible function F'.
Let $y_1, \ldots, y_K$ be new values, different from any value of F. On input x consider the set P_x. If $\#P_x > M/K$, compute the index K' of x in a list of all such inputs, and let the output be $y_{K'}$. (The second property implies $K' \le K$.) Otherwise, let μ_x be the measure obtained by normalizing the probabilities of P_x. Sample an output y from $\mathrm{F}(x)$ using a sample procedure satisfying the conditions of Lemma 4.4 with $b = 3$.

By construction, this algorithm uses at most $\lceil \log(M/K) \rceil + 3$ bits of randomness.

[5] The following sample procedure can be used: pick a random integer i in the interval $[2^d]$, and output the largest y in the support of ν with $\sum_{y' \prec y} \nu(y') \le i/2^d$, where $\prec$ is some fixed order on $\mathcal{Y}$.

Construction of the inverse g' of F'.
For values $y = y_{K'}$, the value of $g'_S(y)$ is the K'-th input for which $\#P_x > M/K$, (thus in this case, the value does not depend on S). For the other values of y, we simply output $g_S(y)$.

It remains to bound the probability that $g'_S(\mathrm{F}'(x)) \neq x$ by 2ε. For x such that $\#P_x > M/K$, this probability is 0, by construction. For the other x, the rescaling in the definition of μ_x amplifies the probability of $g_S(\mathrm{F}(x))$ by a factor at most $1/(1-\varepsilon) \leq 4/3$. The sampling procedure from Lemma 4.4 with $b = 3$ can amplify this probability by at most a factor $1 + 3/8$. Together, the amplification is at most a factor 2. The invertibility condition is satisfied. □

Proposition (First part of Proposition 1.4, restated). $\Delta \geq \log \frac{n}{\epsilon} - \log\log \frac{n}{\epsilon} - 8$ *if* $2^{-n/4} \leq \epsilon \leq 1/4$.

Proof. For a fixed target size m, the compressor provides us with a $(2^{m-\Delta}, \epsilon)$-invertible function. From Lemma 4.5 we obtain a $(2^{m-\Delta}, 2\epsilon)$-invertible function that can be evaluated with $r = \Delta + 3$ random bits, and whose output is 1 bit longer. To this function we apply the same argument as in the proof of the second item of the Proposition. We obtain that

$$\left\lceil \epsilon 2^{\Delta+5} \right\rceil \geq \frac{n - \Delta - 4 - \log(1/\epsilon)}{\Delta + 4}.$$

Note that the right-hand side is decreasing in Δ and the left-hand side is increasing. Hence, any value of Δ that makes the above equation false is a lower bound for the overhead of a decompressor. A calculation shows that for all values of n and ϵ with $2^{-n/4} \leq \epsilon \leq 1/4$, the assignment

$$\Delta \;=\; \log \tfrac{n}{\epsilon} - \log\log \tfrac{n}{\epsilon} - 8.$$

violates the inequality. This finishes the proof. □

Limitations for polynomial time compression

Question. *Suppose there exists an $O(\log \frac{n}{\epsilon})$-optimal polynomial time computable compressor, does this give us an explicit conductor of logarithmic degree?*

We do not know the answer. But, we can prove something weaker: such a compressor implies the existence of a family of conductors of degree $O(2^\Delta)$ that can be computed by polynomial size circuits.

We sketch the proof. We need to prove a variant of Lemma 4.5 where the first invertible function is computable in polynomial time, and the second computable by a family of polynomial sized circuits.

Imagine, we want to evaluate the function F' constructed in the above proof in polynomial time. To do this, we face 2 obstacles. The first is that counting the number of n-bit input strings x' that lexicographically precede x, and satisfy $\#P_x > A$ for the given bound A, takes exponential time. To solve this, we consider strings x with $\#P_x > 2A$, and observe that at most $K/2$ strings satisfy this condition. We handle such strings by recursively applying the algorithm to a $(K/2, \varepsilon)$-invertible function.

A second obstacle is that one needs to decide whether P_x has size larger than A, and if the algorithm for F uses a very large amount of randomness, we can not brute-force search all random choices in polynomial time. However, if Δ is logarithmic, then A is polynomial. If $\#P_x$ is not too much above A, we can approximate $\#P_x$ in polynomial time with a probabilistic algorithm. For this, we try a few random seeds, and inspect the frequencies with which the output values are sampled. With minor modification, such an approximation is enough to execute the plan above.

Finally, with a standard technique, we transform probabilistic algorithms to circuits. We obtain circuits of size $\mathrm{poly}(\frac{n}{\epsilon}, 2^\Delta)$ which map n-bit strings to m-bit strings, and compute $(2^{m-\Delta}, \varepsilon)$-conductors.

5 ACKNOWLEDGEMENTS

The first author is grateful to Andrei Romashchenko and Sasha Shen for useful discussions and for their insightful suggestions.

REFERENCES

Bauwens, B., Makhlin, A., Vereshchagin, N., & Zimand, M. 2013. Short lists with short programs in short time. *In: Proceedings of 28th IEEE Conference on Computational Complexity, Stanford, California, USA*.

Bauwens, Bruno, & Zimand, Marius. 2014. Linear List-Approximation for Short Programs (or the Power of a Few Random Bits). *Pages 241–247 of: IEEE 29th Conference on Computational Complexity, CCC 2014, Vancouver, BC, Canada, June 11-13, 2014*. IEEE.

Brudno, A. A. 1982. Entropy and the complexity of the trajectories of a dynamic system. *Trudy Moskovskogo Matematicheskogo Obshchestva*, **44**, 124–149.

Capalbo, M. R., Reingold, O., Vadhan, S. P., & Wigderson, A. 2002. Randomness conductors and constant-degree lossless expanders. *Pages 659–668 of:* Reif, John H. (ed), *STOC*. ACM.

Cover, Thomas M. 1975. A proof of the data compression theorem of Slepian and Wolf for ergodic sources (Corresp.). *IEEE Transactions on Information Theory*, **21**(2), 226–228.

Cover, Thomas M., & Thomas, Joy A. 2006. *Elements of information theory (2. ed.)*. Wiley.

Dueck, G., & Wolters, L. 1985. The Slepian-Wolf theorem for individual sequences. *Problems of Control and Information Theory*, **14**, 437–450.

Guruswami, Venkatesan, Umans, Christopher, & Vadhan, Salil P. 2009. Unbalanced expanders and randomness extractors from Parvaresh–Vardy codes. *J. ACM*, **56**(4).

Han, Te Sun. 2003. *Information-spectrum methods in information theory*. Springer Verlag.

Han, Te Sun, & Verdú, Sergio. 1993. Approximation theory of output statistics. *IEEE Trans. Information Theory*, **39**(3), 752–772.

Hoeffding, Wassily. 1963. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Society*, **58**, 13–30.

Horibe, Yasuichi. 2003. A note on Kolmogorov complexity and entropy. *Applied mathematics letters*, **16**(7), 1129–1130.

Huffman, D.A. 1952. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, **40**(9), 1098–1101.

Jukna, S. 2011. *Extremal Combinatorics*. Springer Verlag. 2nd edition.

Kuzuoka, S. 2009. Slepian-Wolf coding of individual sequences based on ensembles of linear functions. *IEICE Trans. Fundamentals*, **E92-A**(10), 2393–2401.

Lempel, A., & Ziv, J. 1976. On the complexity of finite sequences. *IEEE Trans. Inf. Theory*, **IT-22**, 75–81.

Ming, Li, & Vitányi, Paul M.B. 2014. *Kolmogorov complexity and its applications*. Elsevier.

Miyake, S., & Kanaya, F. 1995. Coding theorems on correlated general sources. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Science*, **E78-A(9)**, 1063–1070.

Muchnik, Andrei A. 2002. Conditional complexity and codes. *Theor. Comput. Sci.*, **271**(1-2), 97–109.

Musatov, D., Romashchenko, A. E., & Shen, A. 2011. Variations on Muchnik's Conditional Complexity Theorem. *Theory Comput. Syst.*, **49**(2), 227–245.

Nisan, N., & Zuckerman, D. 1996. Randomness is linear in space. *Journal of Computer and System Sciences*, **52**, 43–52.

Radhakrishnan, J., & Ta-Shma, A. 2000. Tight bounds for dispersers, extractors, and depth-two superconcentrators. *SIAM Journal on Discrete Mathematics*, **13**(1), 2–24.

Raz, R., Reingold, O., & Vadhan, S. 1999. Extracting all the randomness and reducing the error in Trevisan's extractor. *Pages 149–158 of: Proceedings of the 30th ACM Symposium on Theory of Computing*. ACM Press.

Raz, Ran, & Reingold, Omer. 1999. On Recycling the Randomness of States in Space Bounded Computation. *Pages 159–168 of:* Vitter, Jeffrey Scott, Larmore, Lawrence L., & Leighton, Frank Thomson (eds), *STOC*. ACM.

Raz, Ran, Reingold, Omer, & Vadhan, Salil P. 2002. Extracting all the Randomness and Reducing the Error in Trevisan's Extractors. *J. Comput. Syst. Sci.*, **65**(1), 97–128.

Romashchenko, A. 2005. Complexity interpretation for the fork network coding. *Information Processes*, **5**(1), 20–28. In Russian. Available in English as [Romashchenko 2016].

Romashchenko, Andrei. 2016. Coding in the fork network in the framework of Kolmogorov complexity. *CoRR*, **abs/1602.02648**.

Shannon, C. E. 1948. A mathematical theory of communication. *Bell Sys. Tech. Jour.*, **27**. Monograph B-1598.

Slepian, D., & Wolf, J.K. 1973. Noiseless coding of correlated information sources. *IEEE Transactions on Information Theory*, **19**(4), 471–480.

Ta-Shma, A., Umans, C., & Zuckerman, D. 2007. Lossless Condensers, Unbalanced Expanders, And Extractors. *Combinatorica*, **27**(2), 213–240.

Teutsch, Jason. 2014. Short lists for shortest descriptions in short time. *Computational Complexity*, **23**(4), 565–583.

Vadhan, Salil P. 2012. Pseudorandomness. *Foundations and Trends in Theoretical Computer Science*, **7**(1-3), 1–336.

Wyner, A.D., & Ziv, J. 1994. The sliding window Lempel-Ziv is asymptotically optimal. *Proc. IEEE*, **2**(6), 872–877.

Zimand, Marius. 2014. Short Lists with Short Programs in Short Time - A Short Proof. *Pages 403–408 of:* Beckmann, Arnold, Csuhaj-Varjú, Erzsébet, & Meer, Klaus (eds), *Language, Life, Limits - 10th Conference on Computability in Europe, CiE 2014, Budapest, Hungary, June 23-27, 2014. Proceedings*. Lecture Notes in Computer Science, vol. 8493. Springer.

Zimand, Marius. 2017. Kolmogorov complexity version of Slepian-Wolf coding. *Pages 22–32 of:* Hatami, Hamed, McKenzie, Pierre, & King, Valerie (eds), *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*. ACM.

Ziv, J. 1978. Coding theorems for individual sequences. *IEEE Trans. Inform. Theory*, **IT-24**, 405–412.
Ziv, J. 1984. Fixed-rate encoding of individual sequences with side information. *IEEE Trans. Inform. Theory*, **IT-30**, 348–352–412.
Zvonkin, A., & Levin, L. 1970. The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms. *Russian Mathematical Surveys*, **25**(6), 83–124.

A PROOF OF LEMMA 2.5

Lemma (Restated). *For all $x \in \mathcal{X}$, let $F_x \subseteq Y$. If $\#\mathcal{Y} < \min\{\frac{1}{4}K^2, \#\mathcal{X} - \frac{1}{2}K\}$, there exists an $x \in \mathcal{X}$ and a set $S \subseteq \mathcal{X}$ of size less than K such that*

$$F_x \subseteq \bigcup_{z \in S, z \neq x} F_z .$$

Proof. Note that by assumption on the size of $\mathcal{Y}$, there exists at least one $x \in \mathcal{X}$ such that

$$F_x \subseteq \bigcup_{z \in \mathcal{X}, z \neq x} F_z ,$$

because if this was not the case, we could build a 1-to-1 correspondence between elements of $\mathcal{X}$ and some elements of $\mathcal{Y}$, by mapping $z \in \mathcal{X}$ to an element in F_z that appears in no other set $F_{z'}$, and this contradicts $\#\mathcal{Y} < \#\mathcal{X} - K/2$. In fact, by the same reasoning, there exist $K/2$ such elements $x_1, \ldots, x_{K/2}$.

If there exists such an x_i for which F_{x_i} has size less than K, then we choose $x = x_i$ and obtain S by selecting for each $y \in F_x$ an element $z \in \mathcal{X}$ with $y \in F_z$. Otherwise, if each such set has size at least K, we construct $S \subseteq \mathcal{X}$ together with a set $Y \subseteq \mathcal{Y}$ iteratively as follows. Initially, S and Y are empty. At stage i, we check whether adding F_{x_i} to Y increases its cardinality by at least $K/2$. If this is indeed so, we add x_i to S and F_{x_i} to Y, and proceed to stage $i+1$. Otherwise, we choose $x = x_i$, and exit the iterative process. Finally, for each $y \in F_x \setminus Y$, we add an element z to S for which $y \in F_z$.

In the first case, the inclusion of the lemma is satisfied by construction. In the second case, we first need to verify that there exists an index $i \leq K/2$ for which the cardinality of Y increases by at most $K/2$. Indeed, assume that for all such i the cardinality increases by more than this amount, then after $K/2$ stages, the cardinality of Y exceeds $(K/2) \cdot (K/2)$. On the other hand, $Y \subseteq \mathcal{Y}$ and by assumption, the cardinality of Y is at most $K^2/4$. A contradiction. This implies that S contains less than $K/2$ elements of the form x_i. In the last step, we add at most $\#(Y \setminus F_x)$ elements to S, and by construction this is at most $K/2$. In total S contains less than K elements. Also, by construction, the inclusion of the lemma is satisfied. □

B STANDARD DEFINITIONS OF CONDUCTORS AND CONDENSERS

We present the definitions of condensers and conductors from the literature, and explain their equivalence with our versions.

Let X be a random variable in $\mathcal{X}$ with probability measure P. The *minentropy* is

$$H_\infty(X) = \min\{\log(1/P(x)) : x \in \mathcal{X}\}.$$

We use the statistical distance to measure similarity of random variables. Let Y and Y' be random variables having the same range $\mathcal{Y}$. These variables are said to be ε-close if $|\Pr[Y \in T] - \Pr[Y' \in T]| \leq \varepsilon$ for all $T \subseteq \mathcal{Y}$.

Remark. In this definition, the maximal difference is obtained for the set T given by all values $y \in \mathcal{Y}$ for which the probability of Y exceeds the probability of Y'. Hence, if Y and Y' have statistical distance at most ε, then for any γ, the γ-excess of Y and Y' differ by at most ε.

A random variable with minentropy at least k is called a *k-source*. A random variable that is ε-close to a k-source, is called a *(k,ε)-source*.

Lemma B.1. *Let Y be a random variable in a domain of size at least K. Y is a $(\log K, \varepsilon)$-source if and only if it has $(1/K)$-excess at most ε.*

Proof. Assume that Y is a $(\log K, \varepsilon)$-source. By definition, there exists a $(\log K)$-source Y' with statistical distance at most ε from Y. Since the $(1/K)$-excess of Y' is zero, the remark above implies that Y has $(1/K)$-excess at most ε. Note that this direction does not use the assumption on the domain size.

Now assume that Y has $(1/K)$-excess at most ε. Let Y' be a $(\log K)$-source obtained by trimming the measure of Y to the value $1/K$, and redistributing all trimmed measure over the other values, while keeping them bounded by $1/K$. This is possible, because the domain of Y is at least K. The variables Y and Y' are ε-close, because for the optimal set T of the remark, the difference is precisely the $(1/K)$-excess of Y. Hence, Y is a $(\log K, \varepsilon)$-source. □

Lemma B.2. *A function $x \mapsto f(x, U_{\mathcal{D}})$ is a $(K \xrightarrow{\varepsilon} K')$-condenser if and only if for every $(\log K)$-source X, the variable $f(X, U_{\mathcal{D}})$ is a $(\log K', \varepsilon)$-source.*

Proof. The backwards implication follows directly from Lemma B.2: for every set S of size K, the function $f(U_S, U_{\mathcal{D}})$ is a (K', ε)-source, and hence, has $(1/K')$-excess at most ε.

For the forward direction, observe that each random variable that is a $(\log K)$-source, can be written as a mixture of uniform distributions on sets of size K.[6] Each such variable induces a $(\log K', \varepsilon)$-source. Therefore, the mixture induces a mixture of $(\log K', \varepsilon)$-sources, which is also a $(\log K', \varepsilon)$-source. □

Remark 5. A probabilistic function $\mathrm{F}\colon \mathcal{X} \to \mathcal{Y}$ is a (K,ε)-*extractor* if and only for every $(\log K)$-source, the variable $\mathrm{F}(X)$ is ε-close to $U_{\mathcal{Y}}$. There is only 1 distribution over $\mathcal{Y}$ that is a $(\log \#\mathcal{Y})$-source, which is the uniform distribution. Hence, F is a (K,ε)-extractor if and only if it is a $(K \xrightarrow{\varepsilon} \#\mathcal{Y})$-condenser.

Definition B.1 ([Capalbo *et al.* 2002]). *A function $f\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a $(k_{\max}, \varepsilon, a)$-*simple conductor *if for any nonnegative integer $k \le k_{\max}$ and any k-source X, the induced variable $f(X, U_d)$ is a $(k+a, \varepsilon)$-source.*

Lemma B.3. *If f is a $(k_{\max}, \varepsilon, 1)$-simple conductor, then $x \mapsto f(x, U_d)$ is a $(2^{k_{\max}}, \varepsilon)$-conductor according to Definition 2.2.*

Note that it is not enough to assume that f is a $(k_{\max}, \varepsilon, 0)$-simple conductor, because the definition only considers integers k, and for sets S whose size is not a power of 2, the output might loose 1 bit of minentropy. The proof follows directly from Lemma B.2.

To a probabilistic function $\mathrm{F}\colon \{0,1\}^n \to \{0,1\}^m$ whose evaluation requires r random bits, we associate the function

$$f\colon \{0,1\}^n \times \{0,1\}^r \to \{0,1\}^m$$

obtained by setting the random bits equal to the second argument.

[6] A nice proof from [Vadhan 2012, Lemma 6.10] is as follows. Let $P_1, P_2, \ldots, P_e$ be probabilities on a finite set such that all $P_i \le 1/K$. Construct a circle with circumference 1, and lay out consecutive segments $I_1, I_2, \ldots, I_e$ on its circumference of lengths, respectively, $P_1, P_2, \ldots$. Consider a K-regular polygon inscribed in this circle. The K vertices of the polygon lie on different segments and therefore the polygon specifies K segments. For a subset S of K segments, let α_S be the probability that a random rotation of some fixed polygon specifies S. The result follows by showing that the distribution P is the same as the distribution of $\sum \alpha_S U_S$, where the sum is taken over all K-element sets of segments. Indeed, the probability that we obtain segment I_t by first selecting a random rotation of the fixed polygon, followed by a random selection of one the K segments specified by the rotated polygon is the same as the probability of obtaining I_t by selecting a random point on the circle and taking the segment containing the point, and the latter probability is equal to P_t.

Lemma B.4. *Let $k_{\max} \le m$. If $\mathrm{F}\colon \{0,1\}^n \to \{0,1\}^m$ is a $(2^{k_{\max}}, \varepsilon)$-conductor according to Definition 2.2, then the associated function f is a $(k_{\max}, \varepsilon, 0)$-simple conductor.*

Again this follows directly from the definitions and Lemma B.2.

C NON-EXPLICIT CONDUCTORS

Recall that $[n] = \{1, 2, \dots, n\}$. Proposition 2.7 follows from the following.

Proposition. *For all ε, n and $K_{\max}$, there exists a $(K_{\max}, \varepsilon)$-conductor $\mathrm{F}\colon [2^n] \to [4K_{\max}]$ whose evaluation requires at most $\log \frac{4n}{\varepsilon}$ bits of randomness.*

In the proof we use a theorem by Hoeffding. The following is obtained from equation 2.1 of Theorem 1 in [Hoeffding 1963].

Theorem (Hoeffding bound, simplified). *Let $X_1, \dots, X_t$ be independent variables taking values in the real interval $[0,1]$. Let μ be the expected value of $Z = X_1 + \cdots + X_t$, for all $\nu > \mu$:*

$$\Pr[Z \ge \nu] \;\le\; \left(\frac{\mu}{\nu}\right)^{\nu}.$$

Proof. Let $\mathcal{Y} = [4K_{\max}]$. If $\#\mathcal{Y} > 2^n$, the identity function F satisfies the conditions. If $K_{\max} \le 1/\varepsilon$, then we output a random element of $[2^{\lceil \log(1/\varepsilon)\rceil}] \subseteq \mathcal{Y}$. Assume $\varepsilon K_{\max} > 1$ and $\#\mathcal{Y} \le 2^n$.

We construct a function F such that

$$\forall X \subseteq [2^n] \text{ with } \#X \le K_{\max} \;\; \forall Y \subseteq \mathcal{Y} \text{ with } \#Y = \lceil \varepsilon \#X \rceil \;\; : \;\; \Pr[\mathrm{F}(U_X) \in Y] \;\le\; \varepsilon.$$

We first show that such an F is a conductor. Indeed, for any $K \le K_{\max}$ and any set $X \subseteq [2^n]$ of size K, consider the set Y of elements $y \in \mathcal{Y}$ for which $y = \mathrm{F}(U_X)$ has probability strictly more than $1/K$. We have $\#Y \le \varepsilon K$, because otherwise any subset $Y' \subseteq Y$ of size $\lceil \varepsilon K \rceil$ violates the condition above. The $(1/K)$-excess of $\mathrm{F}(U_X)$ is less than the probability that $\mathrm{F}(U_X) \in Y$, which, by the above condition, is at most ε.

Let $D = \lceil 3n/\varepsilon \rceil$. We obtain F from a function $f\colon [2^n] \times [D] \to \mathcal{Y}$ as $\mathrm{F}(x) = f(x, U_{[D]})$. The existence of the required f is shown by the probabilistic method: we select each value $f(x,i)$ randomly in $\mathcal{Y}$, and show that the property is satisfied with positive probability.

For a fixed (x,i) and a fixed set Y of size at most $\varepsilon K_{\max}$, such a random f satisfies $f(x,i) \in Y$ with probability at most $\varepsilon/2$, since $K_{\max}\varepsilon > 1$ and hence, $\#Y \le \lceil \varepsilon K_{\max} \rceil \le 2\varepsilon K_{\max} \le \varepsilon \#\mathcal{Y}/2$. For fixed sets X and Y satisfying the conditions, consider the event "$f(x,i) \in Y$ *for more than an ε-fraction of* $(x,i) \in X \times [D]$". By the simplified Hoeffding bound, this is at most $2^{-\varepsilon D \#X}$.

The number of sets X of size K is at most 2^{nK}. The number of sets Y of size $\lceil \varepsilon K \rceil$ is $(\#\mathcal{Y})^{\lceil \varepsilon K \rceil} \le 2^{n \lceil \varepsilon \rceil K}$. By the union bound, the probability that the condition is violated is at most:

$$\le \sum_{K \ge 1}^{K_{\max}} 2^{nK} \cdot 2^{nK} \cdot 2^{-\varepsilon K D}.$$

By choice of D, each term is at most 2^{-nK}, thus the sum is strictly smaller than 1. This implies that the required function f exists. □

D CONSTRUCTION OF EXPLICIT CONDUCTORS

We prove Proposition 2.8. For inductive purposes, it is convenient to switch to a stronger type of conductors, which correspond to "conductors of the extracting type" in Capalbo et al. [Capalbo *et al.* 2002].

Definition D.1. *A probabilistic function* $\mathrm{F}\colon \mathcal{X} \to \mathcal{Y}$ *is an* ε*-conductor if it is an* $(\#\mathcal{Y}, \varepsilon)$*-conductor.*

We restate the proposition using this definition.

Proposition. *There exists an explicit family* $\mathrm{F}^{(n)}_{\varepsilon,k}$ *of* ε*-conductors* $\mathrm{F}^{(n)}_{\varepsilon,k}\colon \{0,1\}^n \to \{0,1\}^k$ *whose evaluation requires* $O(\log k \cdot \log \frac{n}{\varepsilon})$ *random bits, for all* ε*,* k *and* n*.*

Every ε-conductor is a $(\#\mathcal{Y} \xrightarrow{\varepsilon} \#\mathcal{Y})$-condenser, and hence a $(\mathcal{Y}, \varepsilon)$-extractor, by Remark 5 in appendix B. Raz, Reingold and Vadhan [Raz *et al.* 1999], and Guruswami, Umans, and Vadhan [Guruswami *et al.* 2009] construct explicit extractors from which we immediately obtain condensers. Both constructions actually give conductors. The latter uses the smallest amount of randomness for all choices of the parameters n, k, ε. Therefore, we explain why this construction also provides conductors.

To prove the main result in [Guruswami *et al.* 2009], several extractors from Theorem 2.9 (restated below) are concatenated. To show that such a concatenation provides larger extractors, Guruswami et al. use a known "composition lemma", see [Guruswami *et al.* 2009, Lemma 4.18]. We show that this concatenation also provides conductors, and for our purposes, an easier variant of the composition lemma is enough. The above proposition follows almost immediately from the next 2 results.

Theorem ([Guruswami *et al.* 2009], Theorem 1.5 or 4.17). *For all* n, κ *and* ε*, there is an explicit family of* $(2^{2\kappa} \xrightarrow{\varepsilon} 2^{\kappa})$*-condensers* $\mathrm{F}\colon \{0,1\}^n \to \{0,1\}^\kappa$ *whose evaluation requires* $O(\log \frac{n}{\varepsilon})$ *bits of randomness.*

Lemma D.1 (Composition lemma). *If* $\mathrm{S}\colon \mathcal{X} \to \mathcal{Y}_1$ *is a* $(K' \xrightarrow{\varepsilon} \#\mathcal{Y}_1)$*-condenser and* $\mathrm{T}\colon \mathcal{X} \to \mathcal{Y}_2$ *is an* ε_2*-conductor with* $\#\mathcal{Y}_2 \geq K'$*, then*

$$x \longmapsto (\mathrm{S}(x), \mathrm{T}(x))$$

is an $(\varepsilon_1{+}\varepsilon_2)$*-conductor.*

We first prove the proposition, and afterwards this lemma.

Proof of Proposition 2.8. For $k \geq n$ the function $\mathrm{F}(x) = x$ satisfies the conditions. For $k \leq 2$, the construction of an ε-conductor that uses $O(1)$ of randomness is also easy. Otherwise, we repeatedly apply Lemma D.1 to the condenser of Theorem 2.9, where we choose $K' = \#\mathcal{Y}_2$, as long as the new output size is at most k. In each application, the bit length of the output increases at least by a factor $3/2$. Hence, $O(\log k)$ applications are sufficient. After this, we do one more application, where $K' < \#\mathcal{Y}_2$ is chosen to obtain the correct output size. We obtain an $O(\varepsilon \log k)$-conductor that uses $O(\log k \cdot \log \frac{n}{\varepsilon})$ bits of randomness. The proposition follows after downscaling ε by a factor $O(\log k)$. □

To prove the lemma, we use a direct corollary of Lemma B.2.

Corollary. *If* $\mathrm{F}\colon \mathcal{X} \to \mathcal{Y}$ *is a* $(K' \xrightarrow{\varepsilon} M)$-condenser*, then for every set* $X \subseteq [\mathcal{X}]$ *of size at least* K'*, the variable* $\mathrm{F}(U_X)$ *has* $(1/M)$*-excess at most* ε*.*

Proof of Lemma D.1. Let $X \subseteq \mathcal{X}$ be a set of size at most $\#\mathcal{Y}_1 \cdot \#\mathcal{Y}_2$. We need to prove that the $(1/\#X)$-excess of $(\mathrm{S}(U_X), \mathrm{T}(U_X))$ is at most $\varepsilon_1 + \varepsilon_2$. If $\#X \leq K'$, then $\#X \leq \#\mathcal{Y}_2$, and already the second component has $(1/K)$-excess

at most ε_2. The first component can only decrease the excess, and hence the property is satisfied. Assume $\#X \geq K'$. By the condenser property and the corollary above, the variable $\mathrm{S}(U_X)$ has $(1/\#\mathcal{Y}_1)$-excess at most ε_1. For a list L, let U_L represent a randomly selected element from L. Our plan is to bound the excess of the pair using the following observation.

> *Let L be a list and b an integer. Assume L' is a maximal sublist of L that contains the same element at most b times. Thus, if an element y appears at most b times in L, the list L' contains all appearances, and otherwise precisely b appearances. Then, the $(b/\#L)$-excess of U_L is equal to the probability* $\Pr[U_L \notin L']$.

We assume that $\mathrm{S}(x)$ can be evaluated by a deterministic function $S\colon \mathcal{X} \times \mathcal{D}_1 \to \mathcal{Y}_1$ by setting $\mathrm{S}(x) = S(x, U_{\mathcal{D}_1})$. Moreover, we assume that $\mathcal{D}_1$ is finite. Let $D_1 = \#\mathcal{D}_1$. If this is not the case, we use an approximation for a finite $\mathcal{D}_1$ and take limits at the end of the proof. Similarly, assume T is evaluated by $T\colon \mathcal{X} \times \mathcal{D}_2 \to \mathcal{Y}_2$ with finite $D_2 = \#\mathcal{D}_2$.

Consider the list L containing all pairs $(S(x, i_1), T(x, i_2))$ for all $x \in X$, $i_1 \in \mathcal{D}_1$ and $i_2 \in \mathcal{D}_2$, which has length $D_1 D_2 \#X$. We need to show that U_L has $(1/\#X)$-excess at most $\varepsilon_1 + \varepsilon_2$. We construct a sublist L' that satisfies the conditions of the observation above.

- Let L_1 be the list of pairs $(S(x, i_1), x)$ for all $x \in X$ and $i_1 \in \mathcal{D}_1$.
- For each $y_1 \in \mathcal{Y}_1$, let L_{y_1} be the list of the first $D_1 \#\mathcal{Y}_2$ pairs of the form $(y_1, \cdot)$ in L_1.
- Split each L_{y_1} in D_1 sublists of size at most $\#\mathcal{Y}_2$ such that in each sublist, each right coordinate is unique. Denote these lists as $L_{y_1,e}$ for $e = 1, \ldots, D_1$.
- Let $P_{y_1,e}$ be the list of pairs $(y_1, T(x, i_2))$ for all $(y_1, x) \in L_{y_1,e}$ and $i_2 \in \mathcal{D}_2$.
- Let $P'_{y_1,e}$ be the sublist containing all D_2 first appearances of pairs of the form $(\cdot, y_2)$.
- Let L' be the concatenation of all lists $P'_{y_1,e}$ for $y_1 \in \mathcal{Y}_1$ and $e \leq D_1$.

By construction, each pair (y_1, y_2) appears at most $D_1 D_2$ times in L'. It remains to show that $\Pr[U_L \notin L'] \leq \varepsilon_1 + \varepsilon_2$. Note that if in the second and fifth step, we did not restrict the number of appearances, then we would obtain $L = L'$. We show that in these steps, we withhold at most an ε_1-fraction and ε_2-fraction of the elements in L, and this finishes the proof.

For the second step, note that since $\#\mathcal{Y}_2 \geq \#X/\#\mathcal{Y}_1$ and $\#L_1 = D_1 \#X$, we have $D_1 \#\mathcal{Y}_2/\#L_1 \geq 1/\#\mathcal{Y}_1$, and hence, the $(D_1 \#\mathcal{Y}_2/\#L_1)$-excess of U_{L_1} is at most ε_1. By the observation, at most an ε_1-fraction of elements are withhold.

For the fifth step, note that the right coordinates in each $L_{y_1,e}$ define a set X' of size at most $\#\mathcal{Y}_2$. By the conductor property, the variable $\mathrm{T}(U_{X'})$ has $(1/\#X')$-excess at most ε_2. By the observation, each list $P'_{y_1,e}$ contains all but an ε_2-fraction of the elements of $P_{y_1,e}$. Hence, this step withholds at most an ε_2-fraction of the elements. □

E PROOF OF PROPOSITION 3.1

Proposition (Restated). *If $\mathrm{F}_1\colon \mathcal{X}_1 \to \mathcal{Y}_1$ is (K_1, ε)-invertible and $\mathrm{F}_2\colon \mathcal{X}_2 \to \mathcal{Y}_2$ is (K_2, ε)-invertible, then $(x_1, x_2) \mapsto (\mathrm{F}_1(x_1), \mathrm{F}_2(x_2))$ is (3ε)-invertible in sets $S \subseteq \mathcal{X}_1 \times \mathcal{X}_2$ for which*

$$\begin{aligned} \#S &\leq K_1 K_2, \\ \#\{(z_1, z_2) \in S : z_2 = x_2\} &\leq K_1 \qquad \forall x_2 \in \mathcal{X}_2, \\ \#\{(z_1, z_2) \in S : z_1 = x_1\} &\leq K_2 \qquad \forall x_1 \in \mathcal{X}_1. \end{aligned}$$

Let the sets in the second and third condition be denoted as S_{x_2} and S_{x_1}. Let $g_1(S_1, \cdot)$ denote an inverse of F_1 in $S_1 \subseteq \mathcal{X}_1$ and similar for g_2. We need to determine an inverse g of the product function. More precisely, for every $(y_1, y_2) \in \mathcal{Y}_1 \times \mathcal{Y}_2$, we need to specify a value $g(S, y_1, y_2)$ that satisfies the conditions of (3ε)-invertibility.

A first idea is to define the set T containing all (x_1, x_2) for which $x_1 = g_1(S_{x_2}, y_1)$. Thus T contains at most one pair of the form $(\cdot, x_2)$ for all x_2, see the left of Figure 1. Let T_2 be the projection of T on the second coordinate. We apply $g_2(T_2, y_2)$, and if the outcome is x_2, we output the corresponding pair (x_1, x_2). If T_2 has size at most K_2, then the constructed mapping $g(S, \cdot)$ is indeed a (2ε)-inverse in S, but unfortunately, this assumption may not be true. For example, this happens if S is a large diagonal, see the middle of Figure 1. In the proof below, we use a second partition of S, and by evaluating g_1 a second time on its parts, we can prune T to the required size.

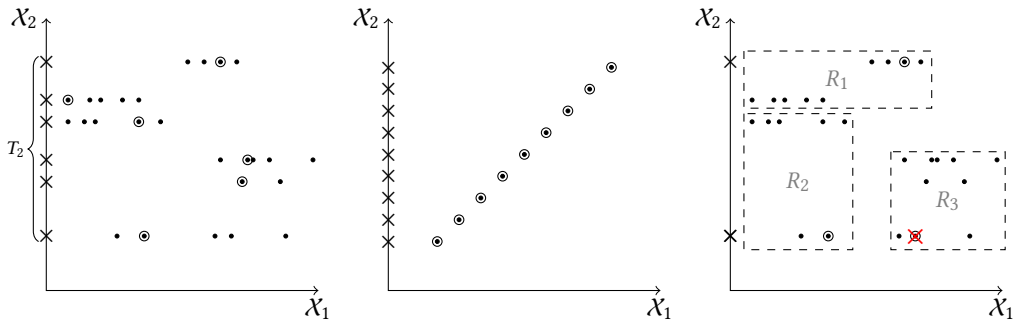

Fig. 1. Left: a set T denoted in circles. Middle: a large diagonal set. Right: Inputs are split in 3 parts, in which one is selected.

Proof. Let $R_1, \ldots, R_{K_2}$ be a partition of S where each set R_i has size at most K_1 and contains at most 1 pair of the form $(x_1, \cdot)$ for all $x_1 \in \mathcal{X}_1$, see the right of Figure 1.

Such a set can be obtained by assigning the elements of S as follows. For all $x_1 \in \mathcal{X}_1$, select the pairs of the form $(x_1, \cdot)$ in S, and add them to different sets R_i of smallest cardinality. Note that since $\#S_{x_1} \leq K_2$, this is always possible. In this way, the parts R_i are filled such that in each moment, their cardinalities differ by at most one.

Construction of $g(S, y_1, y_2)$. Let T be the set of pairs $\mathrm{x} = (x_1, x_2)$ in S for which

$$g_1(R'_{\mathrm{x}}, y_1) \;=\; g_1(S_{x_2}, y_1) \;=\; x_1,$$

where R'_{x} is the projection on the first coordinate of the part R_i that contains (x_1, x_2). Recall that T_2 is the projection of T on the second coordinate. Let the value of g be the pair (x_1, x_2) in T for which $g_2(T_2, y_2) = x_2$, provided the output of g_2 is defined. Otherwise, let the value be undefined.

In the example at the right of Figure 1, the pairs selected for R_2 and R_3 lie on the same horizontal line. The application of $g_1(S_{x_2}, y_1)$ selects one of them. We now prove that with probability at most 3ε:

$$g(S, \mathrm{F}_1(x_1), \mathrm{F}_2(x_2)) \;\neq\; (x_1, x_2).$$

Recall that S_{x_2} and R'_x have at most K_1 elements. T_2 has at most K_2 elements, since T contains at most 1 element from each part R_i, and there are K_2 such parts. If

$$\begin{aligned} g_1(R'_x, \mathrm{F}_1(x_1)) &= x_1 \\ g_1(S_{x_2}, \mathrm{F}_1(x_1)) &= x_1 \\ g_2(T_2, \mathrm{F}_2(x_2)) &= x_2, \end{aligned}$$

then the required output appears. Indeed, the first 2 conditions imply that $(x_1, x_2) \in T$, the last condition selects x_2, and the second condition implies that for this value of x_2, there is a unique pair in T of the form $(\cdot, x_2)$. Thus the output must be (x_1, x_2).

By invertibility and the union bound, the probability that one of these events does not happen is at most $\varepsilon + \varepsilon + \varepsilon$. The proposition is proven. □

F DEGREE LOWER BOUNDS FOR CONDENSERS: PROOF OF PROPOSITION 4.3

Proposition (Restated). *If $f : \mathcal{X} \times [D] \to \mathcal{Y}$ is a (K, ε)-condenser with $\#\mathcal{Y} \geq 2K \geq 4D/\varepsilon$ and $\varepsilon \leq 1/2$, then*

$$\lceil 2\varepsilon D \rceil \;\geq\; \frac{\log(\#\mathcal{X}/K)}{3 + \log(\#\mathcal{Y}/K)}.$$

Proof. We first increase ε to the minimal value such that $2\varepsilon D$ is a positive integer. This can be done without affecting the statement, because for $\varepsilon = 1/2$ the value is always integer.

If y is randomly selected according to a distribution on $\mathcal{Y}$ with $(1/K)$-excess at most ε, then for every subset $Y \subseteq \mathcal{Y}$ of size less than εK, we have $\Pr[y \in Y] < 2\varepsilon$. Therefore, by contraposition, it suffices to prove the following.

Claim. *If the inequality of the proposition is false, then there exists a subset $Y \subseteq \mathcal{Y}$ of size less than εK, and K different $x \in \mathcal{X}$ for which*

$$\Pr\left[f\left(x, U_{[D]}\right) \in Y\right] \;\geq\; 2\varepsilon\,. \tag{*}$$

We construct Y using the probabilistic method. Let $s = \lceil \varepsilon K \rceil - 1$. Select Y randomly among all s-element subsets. For a fixed $x \in \mathcal{X}$, we lower bound the probability that $f(x, i) \in Y$ for all $i \in [2\varepsilon D]$. Note that the most unfavorable case is when $f(x, 1), \ldots, f(x, [2\varepsilon D])$ are all different. Therefore, the probability is at least

$$\geq \binom{\#\mathcal{Y} - 2\varepsilon D}{s - 2\varepsilon D} \Big/ \binom{\#\mathcal{Y}}{s} \;=\; \frac{s}{\#\mathcal{Y}} \cdot \frac{s-1}{\#\mathcal{Y}-1} \cdot \ldots \cdot \frac{s - 2\varepsilon D + 1}{\#\mathcal{Y} - 2\varepsilon D + 1}.$$

The last factor is the smallest and its numerator is at least $\varepsilon K - 2\varepsilon D \geq \varepsilon K/2$, because our assumptions imply $K \geq 4D$. Thus each factor exceeds $\varepsilon/(2A)$, where $A = \#\mathcal{Y}/K$. By a similar reasoning, the probability that $f(x, i) \notin Y$ for $i = 2\varepsilon D + 1, \ldots, D$ is at least $(1-\varepsilon)^{(1-2\varepsilon)D}$, where we used $(\#\mathcal{Y} - \#Y - D)/\#\mathcal{Y} \geq 1 - \varepsilon/2 - \varepsilon/2$, by the assumptions $\#\mathcal{Y} \geq 2K$ and $\#\mathcal{Y} \geq 2D/\varepsilon$. The probability that precisely $2\varepsilon D$ of the elements $f(x, 1), \ldots, f(x, [D])$ are selected in Y is at least

$$\geq \binom{D}{2\varepsilon D} \cdot \left(\frac{\varepsilon}{2A}\right)^{2\varepsilon D} \cdot (1-\varepsilon)^{(1-2\varepsilon)D} \;\geq\; \left(\tfrac{1}{2\varepsilon}\right)^{2\varepsilon D} \cdot \left(\tfrac{\varepsilon}{2A}\right)^{2\varepsilon D} \cdot (1-\varepsilon)^{\frac{1}{\varepsilon}\cdot\varepsilon D} \;\geq\; \left(\frac{2^{-3}}{A}\right)^{2\varepsilon D}.$$

For a randomly selected Y, the expected number of elements x satisfying (*) is at least $\#\mathcal{X}$ times this quantity, by additivity of expectations. This is at least K, because by assumption, the negation of the inequality of Proposition 4.3 holds, and is equivalent to

$$K < \#\mathcal{X}\left(\frac{2^{-3}}{A}\right)^{2\varepsilon D}.$$

Since for a random Y, the expected number of x satisfying (*) is at least K, at least 1 set Y exists that satisfies the conditions of the claim. Hence the proposition is proven. □

G RELATION WITH PREVIOUS RESULTS ON SHORT LISTS WITH SHORT PROGRAMS

As in Corollary 1.2 we fix a universal Turing machine $\mathcal{U}$. We say that a program is *c-short* for x if it outputs x and if its length is at most $c + C(x)$. In Bauwens et al. [Bauwens *et al.* 2013], it was shown that there exists a polynomial time computable function that on input x outputs a list of polynomially many programs containing an $O(\log |x|)$-short program for x. Both constructions used to prove Theorem 2.1 (the one in Corollary 2.13 and the one in Lemma 2.15) provide such lists of quasipolynomial size. Indeed, we can fix $\varepsilon = 1/2$, and consider the list indexed by the targets $m \leq |x|$ and randomness ρ, used in the evaluation of $\mathcal{C}_{\varepsilon,m}(x)$.

[Several small reformulations and corrections in the sketch.] We can reduce the list size to polynomial using dispersers from [Ta-Shma *et al.* 2007]. We briefly sketch the modifications. Let F_K be the explicit $(K, 1/2)$-disperser from [Ta-Shma *et al.* 2007, Theorem 1.4] with left set $\{0,1\}^n$, polynomial degree D, and a right set of size $\Omega(KD/n^3)$. Such dispersers define $(K, 1-\Omega(1/(n^3D)))$-condensers that can be evaluated with $O(\log n)$ bits of randomness. By adapting Lemma 2.10, we have that F_M is online $(M, \mathrm{poly}(n), \alpha)$-list invertible with $(1 - 1/\mathrm{poly}(n))K$ rejections, for some $\alpha < 1$. This implies that F_M is also $(M, \mathrm{poly}(n), \alpha)$-list invertibile with $M/2$ rejections, by a polynomial number of iterative applications of the inverses. Let $\mathrm{F}(x)$ be the value of $\mathrm{F}_K(x)$ for a randomly selected value $K \in \{M, M/2, \ldots, 1\}$. For some $\beta < 1$, we obtain that F is online $(M, \mathrm{poly}(n) \cdot \log M, \beta)$-list invertible with 0 rejections, by a similar argument as in Corollary 2.11. This implies the result.

Teutsch [Teutsch 2014] and Zimand [Zimand 2014] present lists containing $O(1)$-short programs. Like [Bauwens *et al.* 2013], these papers use online matching and explicit expanders. The condenser approach can not provide $O(1)$-short programs, because this would violate the lower bounds on the overhead given in Proposition 1.4. On the other hand, to obtain probabilistic optimal compression, condensers are unavoidable, as shown in Lemma 2.6. This suggests that there exists no "master theorem" from which we can directly obtain the results about short lists and optimal probabilistic compression.