Short lists with short programs in short time - a short proof*

Marius Zimand †

January 22, 2013

Abstract

Bauwens, Mahklin, Vereshchagin and Zimand [1] and Teutsch [5] have shown that given a string x it is possible to construct in polynomial time a list containg a short description of it. We simplify their technique and present a shorter proof of this result.

1 Introduction

Given that the Kolmogorov complexity is not computable, it is natural to ask if given a string x it is posible to construct a short list containing a minimal (+ small overhead) description of x. Bauwens, Mahklin, Vereshchagin and Zimand [1] and Teutsch [5] show that, surprisingly, the answer is YES. Even more, in fact the short list can be computed in polynomial time. More precisely, [1] showed that one can effectively compute lists of quadratic size guaranteed to contain a description of x whose size is additively $O(1)$ from a minimal one (it is also shown that it is impossible to have such lists shorter than quadratic), and that one can compute in polynomial-time lists guaranteed to contain a description that is additively $O(\log n)$ from minimal. Finally, [5] improved the latter result by reducing $O(\log n)$ to $O(1)$.

Theorem 1 ([5]). *For every standard machine U there is a constant c and a polynomial-time algorithm f such that for every x, $f(x)$ outputs a list of programs that contains a c-short program for x.*[1]

Let us explain the formal terms. Given a Turing machine U, a c-short program for x is a string p such that $U(p) = x$ and the length of p is bounded by c+ (length of a shortest program for x). A machine U is *optimal* if $C_U(x \mid y) \leq C_V(x \mid y) + O(1)$ for all machines V (where C is the Kolmogorov complexity and the constant $O(1)$ may depend on V). An optimal machine U is *standard* if for every machine V there is an efficient translator from any machine V to U, i.e., a polynomial-time computable function t such that for all p, y, $U(t(p), y) = V(p, y)$ and $|t(p)| = |p| + O(1)$.

Both [1] and [5] prove their results regarding polynomial-time computable lists as corollaries of somewhat more general theorems. We present in this note a direct proof of Theorem 1, which is simpler and shorter than the one in [5]. We emphasize that there is no technical innovation in the proof that we present below. We use the same general approach and the same ingredients as in [1] and [5], but, because we go straight to the target, we can take some shortcuts that render the proof simpler.[2]

Proof overview. Essentially we want to compress in polynomial time to (close to) minimal length, such that decompression is computable (not necessarily in poynomial time). This is of course impossible in absolute terms, but here we compress in a weaker sense, because we obtain not a single compressed string, but a list guaranteed to contain the (close to) optimally compressed string. It is natural

†Department of Computer and Information Sciences, Towson University, Baltimore, MD.; email: mzimand@towson.edu; http://triton.towson.edu/~mzimand. This work has been supported by NSF grant CCF 1016158.

[1] It can be shown that the list size is $O(n^{6+\delta})$ for any arbitrarily small constant δ.

[2] The proof given here also produces a smaller value for the constant in the theorem.

to think to use seeded extractors, because an extractor's output is close to being optimally compressed in the Shannon entropy sense. The problem is that we need an extractor with logarithmic seed (because we want a list of polynomial size) and no entropy loss (because we want to decompress). Unfortunately, such extractors have not yet been shown to exist. The key observation from [1], also used in [5], is that in fact a disperser is good enough, and then one can use the disperser from [4], which has the needed parameters. Now, why are dispersers sufficient? The answer, inspired by [3], stems from the idea from [1] to use for this kind of compression graphs that allow on-line matching. These are unbalanced bipartite graphs, which, in their simplest form, have LEFT $= \{0,1\}^{\text{n}}$, RIGHT $= \{0,1\}^{\text{k+small overhead}}$, and left degree = poly$(n)$, and which permit on-line matching up to size $K = 2^k$. This means that any set A of K left nodes, each one requesting to be matched to some adjacent right node, can be satisfied in the on-line manner(i.e., the requests arrive one by one and each request is satisfied before seeing the next one; in our proof we will allow a small number of requests to be discarded, but this should also happen before the next request arrives). The correspondence to our problem is roughly that strings in LEFT are the strings that we want to compress, and the strings in RIGHT are their compressed forms. We need on-line matching because we are going to enumerate left strings as they are produced by the universal machine and each time a string is enumerated we want to find it a match, i.e., to compress it. In order for a graph to allow matching, it needs to have good expansion properties. It turns out that it is enough if left subsets of a given size $K/O(1)$ expand to size K, and a disperser has this property. When we decompress, given the right node (the compressed string), we run the matching algorithm and see which left node has been matched to it. For this the decompressor needs to have n to be able to construct the graph, and this produces the $O(\log n)$ overhead. Thus this approach is good enough to obtain the result with $O(\log n)$-short programs from [1]. To reduce $O(\log n)$ to $O(1)$, we need the new ideas from [5]. The point is that this time we want LEFT to have strings not of a single length n, but of all lengths $n \geq k$ (because we can no longer afford to give n to the decompressor). In fact, it is not hard to see, that it is enough to restrict to lengths $k \leq n \leq 2^k$. This time we need expansion for all sets of size $\leq K$ (not just equal to a fixed $K/O(1)$, because we need each subset (of the match-requesting set A) of strings of a given length to expand. For this, the unbalanced lossless expander from [2] is good, except for one problem: The size of RIGHT in this expander is poly(K) and not the desired $K + O(1)$. This problem is fixed by compressing using again the disperser from [4] to a set of size $K \cdot \text{poly}(k)$, and, finally, using a simple trick, to size $K + O(1)$, which implies the $O(1)$ overhead we aim for.

2 The proof

We use bipartite graphs $G = (L, R, E \subseteq L \times R)$. We denote LEFT(G) = L, RIGHT(G) = R. For integers n, m, k, d we denote $N = 2^k, M = 2^m, K = 2^k, D = 2^d$. We denote $[n] = \{1, 2, \ldots, n\}$. A bipartite graph is explicit if there exists a polynomial-time algorithm that given $x \in$ LEFT(G) and i, outputs the i-th neighbor of x.

Definition 1. *A bipartite graph G is a (K, K')-expander if every subset of left nodes having size K, has at least K' right neighbors.*

Theorem 2 (Guruswami, Umans, Vadhan [2])**.** *For every constant α, every n, every $k \leq n$, and $\varepsilon > 0$, there exists am explicit $(K', (1-\varepsilon)DK')$ expander for every $K' \leq K$, in which every left node has degree $D = O((nk/\varepsilon)^{1+1/\alpha})$, $L = [N], R = [M]$, $M \leq D^2 \cdot K^{1+\alpha}$.*

Definition 2. *A bipartite graph $G = (L, R, E)$ is a (K, δ)-*disperser*, if every subset $B \subseteq L$ with $|B| \geq K$ has at least $(1-\delta)|R|$ distinct neighbors.*

Theorem 3 (Ta-Shma, Umans, Zuckerman [4])**.** *For every K, n and constant δ, there exists explicit (K, δ)-dispersers $G = (L = \{0,1\}^n, R = \{0,1\}^m, E \subseteq L \times R)$ in which every node in L has degree $D = n2^{O((\log\log n)^2)}$ and $|R| = \frac{\alpha KD}{n^3}$, for some constant α.*[3]

[3] [4] only indicates that $D = \text{poly}(n)$. The value $D = n2^{O((\log\log n)^2)}$ is obtained by reworking the proof in [4] using the extractor with constant loss from Theorem 4.21 in [2].

The key combinatorial object that we use is provided in the following lemma.

Lemma 4. *For every constant c and every sufficiently large k, there exists an explicit bipartite graph H_k with the following properties:*

1. $\mathrm{LEFT}(\mathrm{H_k}) = \{0,1\}^{\mathrm{k}} \cup \{0,1\}^{\mathrm{k+1}} \cup \ldots \cup \{0,1\}^{2^{\mathrm{k}}}$, $\mathrm{RIGHT}(\mathrm{H_k}) = \{0,1\}^{\mathrm{k+1}}$,
2. *Each left node x has degree* $\mathrm{poly}(|x|)$,
3. *H_k is a $(K/c^2, K)$-expander.*

We defer the proof of this lemma for later.

We show how the lemma implies Theorem 1. We start with the following lemma about on-line matching (recall that this means that one receives a sequence of requests to match left nodes with one of their adjacent right nodes and each request must be satisfied, or discarded, before seeing the next one).

Lemma 5. *If K on-line matching requests are made in a $(K/c^2, K)$-expander all but less than K/c^2 can be satisfied.*

Proof. Suppose there are K requests for matching left nodes and we attempt to satisfy them in the obvious greedy manner. Suppose that K/c^2 requests cannot be satisfied (because all their neighbors have been used to match previous requests). The K/c^2 left nodes that are not satisfied have K right neighbors and all of them have satisfied matching requests. This would imply that all the K requests have been satisfied, contradiction. □

Proof of Theorem 1.

We define the following machine V ("the decompressor").

(1) On inputs of the form $00p$, V outputs p.
(2) On inputs of the form $01p$, V simulates $U(p)$ and if $U(p) = x$ and $|x| > 2^{|p|}$, outputs x.
(3) On inputs of the form $1p$, V works as follows:
V calculates its value on all inputs of the form $1p'$ with $|p'| = |p|$ as follows. Let $k = |p| - 1$. Enumerate the elements of the set $\{x \mid \exists q \text{ of length } k, U(q) = x\}$. When an element x is enumerated and $|x|$ is between k and 2^k, pass x to the online matching algorithm for H_k. If x is matched to p', then $V(p')$ outputs x. If x is rejected because all its right neighbors in H_k have already been used to match other elements during the computation of $V(1p')$ for strings p' of length $k-1$, continue the enumeration.

Observe that during computations of the form (3), at most K matching requests are made and therefore, by the property of H_k, there are fewer than K/c^2 rejections. It follows that if v is a rejected node then $C_U(v) \leq k - 2\log c + \log c + 2\log\log c + O(1) < k$, for c a large enough constant. Indeed a rejected string can be described by its index in the set of rejected strings written on exactly $k - 2\log c$ bits, and c (which is needed in order to reconstruct k and next enumerate the set of rejected strings). The additional $2\log\log c$ term is required for concatenating the index and c. It follows that if x is a string such that $C_U(x) = k$ and $k \in \{\log|x|, \ldots, |x|\}$, then there exists p of length $k+1$ such that $V(1p) = x$. Moreover, p is one of the right neighbors of x in H_k.

Now, for each x, let $list(x)$ be the list containing the following strings: $00x$, all strings of length $< \log|x|$ prefixed with 01, and all the neighbors of x in H_k prefixed with a 1, for $k = |x|, |x|-1, \ldots, \log(|x|)$. Note that for every x, $list(x)$ can be computed in polynomial time, and there exists $v \in list(x)$, $|v| \leq C_U(x) + O(1)$ such that $C_V(v) = x$. Finally, using the "translator" t from V programs to U programs, take $f(x) = \{t(v) \mid v \in list(x)\}$. Since t is computable in polynomial time, $U(t(v)) = V(v)$ and $|t(v)| = |v| + O(1)$, we are done. □

It remains to prove Lemma 4. We use two types of graphs given in the following two lemmas.

Lemma 6. *For every n, and $k \leq n$, there exists a bipartite graph $GUV_{n,k}$ with each left node having degree $D = \lambda(nk)^2$ (for some fixed constant λ),* $\mathrm{LEFT(GUV_{n,k})} = \{0,1\}^{\mathrm{n}}$, $\mathrm{RIGHT(GUV_{n,k})} = [\mathrm{M}]$ *with $M \leq D^2K^2$, which is a $(K', (1/2)DK')$-expander for every $K' \leq K$.*

Proof. This is the Guruswami, Umans, Vadhan expander with parameters $\alpha = 1, \varepsilon = 1/2$. □

Lemma 7. *For every k, there exists a bipartite graph F_k with each left node having degree $D = O(k^3)$,* $\mathrm{LEFT(F_k)} = \{0,1\}^{8\mathrm{k}}$, $\mathrm{RIGHT(F_k)} = \{0,1\}^{\mathrm{k}+1}$, *which is a (K,K)-expander.*

Proof. Consider the Ta-Shma, Umans, Zuckerman $(K, 1/2)$-disperser G, with $\mathrm{LEFT(G)} = \{0,1\}^{8\mathrm{k}}$, $\mathrm{RIGHT(G)} = \{0,1\}^{\mathrm{m}}$, left degree $D = O(k2^{O((\log\log k)^2)})$ and $|\mathrm{RIGHT(G)}| = \frac{\alpha \mathrm{KD}}{(8\mathrm{k})^3}$.

To increase the size of the right set to be at least $2K$, we make RIGHT consist of $2\lceil \frac{(8k)^3}{\alpha D} \rceil$ copies of RIGHT(G) connected to LEFT(G) in the same way as the original nodes. Thus each right node is labelled by a string of length $\geq k+1$ and the left degree is $O(k^3)$.

By merging the nodes whose labels have the same prefix of length $k+1$, we obtain the graph F_k, which as desired has $\mathrm{RIGHT(F_k)} = \{0,1\}^{\mathrm{k}+1}$ and is a $(K, 1/2)$-disperser (because the merge operation can only improve the dispersion property).

Thus, every left subset of size K has at least $(1/2) \cdot 2K$ right neighbors, i.e., F_k is a (K,K)-expander. □

We are now prepared to prove Lemma 4.

Proof of Lemma 4

Let us fix c and a sufficiently large k.

We first construct the graph G_k as the union $GUV_{k,k} \cup GUV_{k+1,k} \cup \ldots \cup GUV_{2^k,k}$.

Note that $\mathrm{LEFT(G_k)}$ consists of all strings having length between k and 2^k. For $\mathrm{RIGHT(G_k)}$, we shift the numerical labels of the right nodes in each set in the obvious way before taking the union, so that the sets that we union are pairwise disjoint. We have

$$|\mathrm{RIGHT(G_k)}| \leq \sum_{\mathrm{n=k}}^{2^{\mathrm{k}}} \lambda^2(\mathrm{nk})^4\mathrm{K}^2 = \lambda^2\mathrm{k}^4\mathrm{K}^2 \sum_{\mathrm{n=k}}^{2^{\mathrm{k}}} \mathrm{n}^4 \leq \lambda^2\mathrm{k}^4 \cdot \mathrm{K}^7 < \mathrm{K}^8,$$

for k sufficiently large. By padding each right node in G_k with $100\ldots0$, we label each right node by a string of length $8k$.

Note that, provided k is sufficiently large, G_k is a $(K/c^2, K)$-expander. Indeed take $B \subseteq \mathrm{LEFT(G_k)}$, $|B| = K/c^2$. B has strings of different lengths. If we partition B into subsets of strings corresponding to the different lengths, each subset with strings of length say n expands according to $GUV_{n,k}$ by a factor of $(1/2)\lambda(nk)^2 \geq c^2$ (if k is large enough). Since different subsets of the partition map into disjoint right subsets, the above assertion follows.

The degree of every left node x in G_k is bounded by $\mathrm{poly}(|x|)$ because the edges originating in x are those from the graph $GUV_{|x|,k}$. So G_k is almost what we need except that the right nodes have length $8k$ instead of $k+1$. We fix this issue by compressing strings of length $8k$ to length $k+1$ using the graph F_k from Lemma 7.

More precisely, we build the graph H_k by taking the product of the above graph G_k with the graph F_k. Thus $\mathrm{LEFT(H_k)} = \mathrm{LEFT(G_k)}$, $\mathrm{RIGHT(H_k)} = \mathrm{RIGHT(F_k)}$ and (x,y) is an edge in H_k if there exists $z \in \mathrm{RIGHT(G_k)} \subseteq \mathrm{LEFT(F_k)}$ such that (x,z) is an edge in G_k and (z,y) is an edge in F_k. As desired, $\mathrm{LEFT(H_k)}$ consists of all strings x having length between k and 2^k, $\mathrm{RIGHT(H_k)} = \{0,1\}^{\mathrm{k}+1}$, the degree of every left node x is bounded by $\mathrm{poly}(|x|)\mathrm{poly}(k) = \mathrm{poly}(|x|)$ and H_k is a $(K/c^2, K)$-expander, because each left subset of size K/c^2 expands to size at least K in G_k and then it keeps its size at least K when passing through F_k. □

Note. The above construction yields in Theorem 1 a list of size $O(n^8)$. If in Lemma 6 we take a small α (instead of $\alpha = 1$), we obtain list size $O(n^{6+\delta})$, for arbitrarily small constant δ.

3 Acknowledgements

We are grateful to Alexander Shen for his comments and for signalling an error in an earlier version. We thank Jason Teutsch for useful conversations that lead to a more precise estimation of the list size in Theorem 1.

References

[1] B. Bauwens, A. Makhlin, N. Vereshchagin, and M. Zimand. Short lists with short programs in short time. *ECCC*, TR13-007, 2013.

[2] Venkatesan Guruswami, Christopher Umans, and Salil P. Vadhan. Unbalanced expanders and randomness extractors from parvaresh–vardy codes. *J. ACM*, 56(4), 2009.

[3] D. Musatov, A. E. Romashchenko, and A. Shen. Variations on Muchnik's conditional complexity theorem. *Theory Comput. Syst.*, 49(2):227–245, 2011.

[4] A. Ta-Shma, C. Umans, and D. Zuckerman. Lossless condensers, unbalanced expanders, and extractors. *Combinatorica*, 27(2):213–240, 2007.

[5] J. Teutsch. Short lists for shorter programs in short time, 2012. CORR Technical Report arXiv:1212.6104.