Electronic Colloquium on Computational Complexity, Report No. 7 (2013)

Short lists with short programs in short time

Bruno Bauwens [*] Anton Makhlin [†‡] Nikolay Vereshchagin [†§] Marius Zimand [¶]

Abstract

Given a 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 + (the length of a shortest program for x). We show that for any universal machine, it is possible to compute in polynomial time on input x a list of polynomial size guaranteed to contain a $O(\log|x|)$-short program for x. We also show that there exist computable functions that map every x to a list of size $O(|x|^2)$ containing a $O(1)$-short program for x and this is essentially optimal because we prove that such a list must have size $\Omega(|x|^2)$. Finally we show that for some machines, computable lists containing a shortest program must have length $\Omega(2^{|x|})$.

Keywords: list-approximator, Kolmogorov complexity, on-line matching, expander graph

1 Introduction

The Kolmogorov complexity of a string x is the length of the shortest program computing it. Determining the Kolmogorov complexity of a string is a canonical example of a function that is not computable. Closely related is the problem of actually producing a shortest program for x. This problem is also not algorithmically solvable. When faced with a function that is not computable, it is natural to ask whether it can be effectively approximated in a meaningful way. This question has been investigated for Kolmogorov complexity in various ways. First of all, it is well-known that the Kolmogorov complexity can be effectively approximated from above. A different type of approximation is given by what is typically called *list computability* in algorithms and complexity theory and *traceability* in computability theory. For this type of approximation, one would like to compute a list of "suspects" for the result of the function with the guarantee that the actual result is in the list. Of course, the shorter the list is, the better is the approximation.

The list approximability of the Kolmogorov complexity, $C(x)$, has been studied by Beigel et al. [BBF$^+$06]. They observe that $C(x)$ can be approximated by a list of size $(n-a)$ for every constant a, where $n = |x|$. On the other hand, they show that, for every universal machine U, there is a constant c such that for infinitely many strings x (in fact for at least one x at each sufficiently large length n), any computable list containing $C_U(x)$ must have size larger than n/c.

In this paper we study list approximability for the problem of producing short programs. In order to describe our results, we need several formal definitions.

[*] LIRMM CNRS & University 2 of Montpellier. Supported by NAFIT ANR-08-EMER-008-01 project; http://www.bcomp.be

[†] Moscow State University. The work was in part supported by the RFBR grant 12-01-00864 and the ANR grant ProjetANR-08-EMER-008

[‡] E-mail: amakhlin@bk.ru

[§] E-mail: ver@mccme.ru, WWW home page: http://lpcs.math.msu.su/~ver.

[¶] Department of Computer and Information Sciences, Towson University, Baltimore, MD.; email: mzimand@towson.edu; http://triton.towson.edu/~mzimand. The author is supported in part by NSF grant CCF 1016158.

ISSN 1433-8092

A machine U is *optimal* if $C_U(x|y) \le C_V(x|y) + O(1)$ for all machines V (where the constant $O(1)$ may depend on V). An optimal machine U is *standard*[1], if for every machine V there is a total computable function t such that for all p,y: $U(t(p),y) = V(p,y)$ and $|t(p)| = |p| + O(1)$. For results that hold in polynomial time, we additionally assume these functions t run in time polynomial in $|p|$. Let $U(p)$ stand for $U(p,\text{the empty string})$ and $C_U(x)$ for $C_U(x|\text{the empty string})$. A *c-short program* for x with respect to U is a string p that satisfies $U(p) = x$ and $|p| \le C_U(x) + c$.

Given an optimal machine U, a *list-approximator* for c-short programs is a function f that on every input x outputs a finite list of strings such that at least one of the elements in the list is a c-short program for x on U. Let $|f(x)|$ denote the number of elements in the list $f(x)$. Obviously, for every optimal U, there is a (trivial) computable list-approximator f such that $|f(x)| \le 2^{|x|+O(1)}$.

The question we study is how small can $|f(x)|$ be for computable list-approximators f for c-short programs, where c is a constant or $O(\log|x|)$. At first glance it seems that in both cases $|f(x)|$ must be exponential in $|x|$. Surprisingly, this is not the case. We prove that there is a computable approximator with list of size $O(|x|^2)$ for c-short programs for some constant c depending on the choice of the standard machine U. And we show that this bound is tight. We show also that there is a *polynomial time computable* approximator with list of size $\text{poly}(|x|)$ for c-short programs for $c = O(|\log|x|)$.

We start with the positive results, i.e., the upper bounds. We show for every standard machine, there exists a list-approximator for $O(1)$-short programs, with lists of *quadratic* size.

Theorem 1.1. *For every standard machine U there exists a computable function f that for any x produces a list with $O(|x|^2)$ many elements containing a program p for x of length $|p| = C_U(x) + O(1)$.*

If we allow $O(\log|x|)$-short programs we can construct lists of polynomial size in *polynomial time*.

Theorem 1.2. *For every standard machine U, there exists a polynomial-time computable function f that for any x produces a list with poly$(|x|)$ many elements containing a program for x of length $C_U(x) + O(\log|x|)$.*[2]

Now we move to the lower bounds. We show that the quadratic lower bound in Theorem 1.1 is optimal: it is not possible to compute lists of subquadratic size that contain a $O(1)$-short program.

Theorem 1.3. *For all $c > 0$, for every optimal U, for every computable f that is a list-approximator for c-short programs,*

$$|f(x)| \ge \Omega(|x|^2/c^2),$$

for infinitely many x. (The constant hidden in Ω-notation depends on the function f and machine U.)

A weaker linear lower bound can be easily derived from a result of Bauwens [Bau12], improving a theorem of Gács [Gác74]. The result states that $C_U(C_U(x) \mid x)$ is greater than $\log|x| - O(1)$ for infinitely many x. Thus, for infinitely many x,

$$\log|x| - O(1) \le C_U(C_U(x) \mid x) \le \log|f(x)| + 2\log c + O(1),$$

and therefore $|f(x)| \ge \Omega(|x|/c^2)$.

The next theorem shows that, at least for some standard machine U, we can not compute lists of subexponential size containing a program of length exactly $C(x)$.

---

[1] This notion was introduced by Schnorr [Sch75], and he called such machines *optimal Gödel numberings* (of the family of all computable functions from strings to strings). We use a different term to distinguish between optimal functions in Kolmogorov's sense and Schnorr's sense

[2] Recently Jason Teutsch [T11] improved this result replacing $O(\log|x|)$ by $O(1)$.

Theorem 1.4. *There exists a standard*[3] *machine U such that for every computable f that is a list-approximator for 0-short programs,*

$$|f(x)| \geq 2^{|x|-O(1)},$$

for infinitely many x. (The constant hidden in O-notation depends on the function f.)[4]

The construction of U can be easily adapted to show that for every c there is a standard machine U_c satisfying the theorem for c-short programs. This exponential lower bound does not hold for all standard machines (see Remark 1 below).

In the proofs of upper bounds we use non-explicit and explicit bipartite graphs of small degree that have expander-like properties and hence have good matching abilities. The lower bounds are based on lower bounds for degree of such graphs. This connection is studied in Section 2. The upper bounds, i.e., Theorem 1.1 and Theorem 1.2, are proved in Section 3. The lower bounds, i.e., Theorem 1.3 and Theorem 1.4, are proved in Section 4. In Section 5, we observe that our construction can be used to improve Muchnik's Theorem [Muc02, MRS11], and a result concerning distinguishing complexity [BFL01].

2 List approximators for short programs and on-line matching

We will show that the problem of constructing approximators for short programs is equivalent to constructing families of bipartite graphs of certain type. Let a bipartite graph $(L, R, E \subset L \times R)$ be given, where the set L of left nodes and the set R of right nodes consist of binary strings. Assume that we receive "requests for matching" in the graph, each request having the form (a binary string $x \in L$, a natural number k). Such request means that we have to provide to the left node x a match of length at most k or slightly more. It might happen that we receive a request (x, k) and later a request (x, k') with the same x and another $k' < k$. In this case we are allowed to match x to the second right node, thus x becomes two matching right nodes. Assignments cannot be changed and different right nodes must have different matches (however any left node x may have different matches). We will sometimes call right nodes *hash-values*.

Definition 2.1. Let $c(n)$ be a function of n with natural values. A bipartite graph whose left and right nodes are binary strings *has matching with overhead* $c(n)$ if the following holds. For every set S of pairs $(x \in L, k)$ having at most 2^k pairs with the second component k (for all k) one can choose for every pair (x, k) in S a neighbor $p(x, k)$ of x so that $|p(x, k)| \leq k + c(|x|)$ and $p(x_1, k_1) \neq p(x_2, k_2)$ whenever $x_1 \neq x_2$. If this is done, we will say that $p(x, k)$ *matches* x.

A bipartite graph *has on-line matching with overhead* $c(n)$ if this can be done in the on-line fashion: requests for matching (x, k) appear one by one and we have to find $p(x, k)$ before the next request appears. All the made assignments cannot be changed.

A bipartite graph is *(polynomial time) computable* if given a left node x we can compute (in polynomial time) the list of all its neighbors.

Theorem 2.2. *Assume there is a computable graph with $L = \{0, 1\}^*$ where the left node x has degree $D(x)$ and which has on-line matching with overhead $c(n)$. Assume further that the matching strategy is computable. Then for every standard machine U there exists a computable function f that for any x produces a list with $D(x)$ many elements containing a program p for x of length $|p| = C_U(x) + c(|x|) +$*

[3]The construction implies the existence of such U with a stronger universality property: for every machine V there exists a string w_V such that $U(w_V p|z) = V(p|z)$ for all p, z.

[4]Independently, this result was obtained by Frank Stephan (personal communication).

$O(1)$. *If the graph is polynomial time computable then the function f is polynomial time computable, too.*

Proof. Run the optimal machine $U(q)$ in parallel for all strings q. Once $U(q)$ halts with the result x we pass the request $(x,|q|)$ to the matching algorithm in the graph and find a hash value p of length at most $|q|+c(|x|)$ for x.

By construction, every string x is matched to a string p of length at most $C_U(x)+c(|x|)$. Each right node is matched to at most one node in the graph. Hence there is a machine V such that $V(p)=x$ whenever p is matched to x. Thus for every string x there is a neighbor p of x with $|p|\le C_U(x)+c(|x|)$ and $V(p)=x$. As U is a standard machine, there is a (polynomial time) computable function t with $U(t(p))=V(p)$ and $|t(p)|\le|p|+O(1)$. Let $f(x)$ be the list consisting of $t(p)$ for all the neighbors p of x in the graph. By construction $|f(x)|=D(x)$ and we are done. □

Remark 1. If $c(n)=c$ is a constant function, then we can construct a standard machine U_1 which has a computable approximator for 0-short programs with lists of size $D(x)$. To this end let $U_1(0p)=V(p)$ and $U_1(1^{c+2}p)=U(p)$ and let $t(p)=0p$ in the construction of f in the proof of the above theorem. By Theorem 2.4 below there is a graph having on-line matching with constant overhead and degree $O(|x|^2)$. Thus there is a standard machine U_1 which has a computable approximator for 0-short programs with lists of size $O(|x|^2)$.

There is also a reduction in the other direction.

Theorem 2.3. *Assume that $c(n)$ is computable function and there are an optimal machine U and a computable function f that for any x produces a finite list containing a program p for x of length $|p|\le C_U(x)+c(|x|)$. Consider the bipartite graph G with $L=\{0,1\}^*$ where the neighbors of node x are all strings from $f(x)$. Then G has on-line matching with overhead $c(|x|)+O(1)$.*

Proof. For each n let G_n be the subgraph of G with $L=\{0,1\}^{\le n}$. W.l.o.g. we assume that all strings in $f(x)$ have length at most $|x|+O(1)$ and hence the graph G_n is finite. We claim that G_n has on-line matching with overhead $c(|x|)+O(1)$ for all n, (where the $O(1)$ constant does not depend on n).

We first show that this implies the theorem. Suppose that $M_1,M_2,\dots$ are on-line matching strategies for graphs $G_1,G_2,\dots$ It suffices to convert them to strategies $M'_1,M'_2,\dots$ for $G_1,G_2,\dots$ such that for all i, $j>i$ strategy M'_j is an extension of M'_i, i.e. on a series of requests only containing nodes from G_i, strategy M'_j behaves exactly as M'_i. Because each G_n finite, there are only finitely many different matching strategies for G_n. Hence, there is a strategy M'_1 that equals the restriction of M_n to G_1 for infinitely many n. Therefore there is also a strategy M'_2 that is an extension of M'_1 and equals the restriction of M_n to G_2 infinitely often, and so on.

It remains to show the claim. For the sake of contradiction assume that for every constant i there is n such that G_n has not on-line matching with overhead $c(|x|)+i$, (and f is a list-approximator for $c(|x|)$-short programs on U). Because G_n is finite, for all n and c one can find algorithmically (using an exhaustive search) whether G_n has on-line matching with overhead $c(|x|)+i$ or not. One can also find a winning strategy for that player who wins (“Matcher” or “Requester”). Therefore for every i we can algorithmically find the first n such that the graph G_n has not on-line matching with overhead $c(|x|)+i$ and the corresponding winning strategy for Requester for G_n.

Let that strategy play against the following “blind” strategy of Matcher. Receiving a request (x,k) the Matcher runs $U(p)$ for all $p\in f(x)$, $|p|\le k+c(|x|)+i$, in parallel. If for some p, $U(p)$ halts with the result x, he matches the first such p to x and proceeds to the next request. Otherwise the request remains not fulfilled.

Consider the following machine V. On input (q,i) with $|q|=k$ it finds the first G_n such that the graph G_n has not on-line matching with overhead $c(|x|)+i$ and a winning strategy for Requester, and runs it against the blind strategy of Matcher. Then it returns x where (x,k) is the qth request with the second component k. As Requester wins, there is a request (x,k) that was not fulfilled. We have

$$C_U(x) \le C_V(x)+O(1) \le k+2\log i+O(1) \le k+i \tag{1}$$

(the last inequality holds for all large enough i). As the request (x,k) was not fulfilled, there is no p in $f(x)$ with $|p| \le k+i+c(|x|)$. Due to (1), $f(x)$ has no $c(|x|)$-short program for x, a contradiction. □

From the technical point of view, our main contributions are the following theorems.

Theorem 2.4 (Combinatorial version of Theorem 1.1)**.** *There is a computable graph with $L=\{0,1\}^*$ with left degree $D(x)=O(|x|^2)$ which has on-line matching with overhead $O(1)$.*

Theorem 2.5 (Combinatorial version of Theorem 1.2)**.** *There is a polynomial time computable graph with $L=\{0,1\}^*$ with left degree $D(x)=poly(|x|)$ which has on-line matching with overhead $O(\log|x|)$.*

Theorem 2.6 (Combinatorial version of Theorem 1.3)**.** *In every graph G with $L=\{0,1\}^n$ that has off-line matching with overhead c, the maximal degree of left nodes is $\Omega(n^2/(c+O(1))^2)$.*

Theorems 2.4, 2.5 and 2.6 imply Theorems 1.1, 1.2 and 1.3, respectively. Moreover, our on-line matching strategy used in the proofs of Theorems 2.4 and 2.5 are very simple: receiving a new request of the form (x,k) we just find the maximal $i \le k+c$ such that there is a free hash-value of length i and match x with the first neighbor of x, in some order, which is not used so far.

3 The upper bounds

In this section we prove Theorems 2.4 and 2.5. We will need the notion of a graph with on-line matching, introduced in [MRS11].

Definition 3.1. Say that a bipartite graph has *matching up to K with at most M rejections*, if for any set of left nodes of size at most K we can drop at most M its elements so that there is a matching in the graph for the set of remaining nodes. A graph has an *on-line matching up to K with at most M rejections* if we can do this in on-line fashion. For $M=0$ we say that the graph has *matching up to K*. A bipartite graph is called a *(K,K')-expander*, if every set of K left nodes has at least K' distinct neighbors.

Graphs that have matching up to K are closely related to (K,K)-expanders. Indeed, any graph having off-line matching up to K is obviously a (K',K')-expander for all $K' \le K$. Conversely, by Hall's theorem [Hal35] any graph which is a (K',K')-expander for all $K' \le K$ has off-line matching up to K.

In [MRS11] it was shown that a reduction from expanders to *on-line* matching is also possible. More specifically, every family of $(2^i,2^i)$-expanders, one for each $i<k$, sharing the same set L of left nodes can be converted into a graph with the same set L of left nodes that has on-line matching up to 2^k, at the expense of multiplying the degree by k and increasing hash-values by 1. (We will present the construction in the proof of Theorem 3.3.)

The connection between graphs with matching up to K and graphs with matching with overhead c is the following. If a graph G has (on-line) matching with overhead c then removing from G all left nodes of length different from n and all right nodes of length more than $k+c(n)$ we obtain a graph with (on-line) matching up to 2^k. On the other hand, assume that for some n for all $k < n$ we have a graph $G_{n,k}$ with $L = \{0,1\}^n$ and $R = \{0,1\}^{k+c(n)}$ which has (on-line) matching up to 2^k. Then the union G_n of $G_{n,k}$ over all $k < n$ has (on-line) matching with overhead c, provided all requests (x,k) satisfy $k < |x|$. At the expense of increasing the degree and c by 1, the graph G_n can be easily modified to have (on-line) matching with overhead c unconditionally: append 0 to all right nodes of G_n and for every $x \in \{0,1\}^n$ add a new right node $x1$ connected to x only.

In [MRS11] it is observed that every $(2^{k-1}, 2^{k-1})$-expander has on-line matching up to 2^k with at most 2^{k-1} rejections. We need a slight generalization if this fact.

Lemma 3.2. *Every $(2^{k-\ell}, 2^k - 2^{k-\ell})$-expander has on-line matching up to 2^k with at most $2^{k-\ell}$ rejections.*

Proof. Use the following greedy strategy for on-line matching: each time a left vertex is received, check if it has a neighbor that was not used yet. If yes, any such neighbor is selected as the match for that node. Otherwise, the node is rejected.

For the sake of contradiction, assume that the number of rejected nodes is more than $2^{k-\ell}$. Choose from them exactly $2^{k-\ell}$ rejected nodes. By expansion property, they have at least $2^k - 2^{k-\ell}$ neighbors and all those neighbors are used by the greedy strategy (otherwise the node having a non-used neighbor would not be rejected). Thus we have at least $2^k - 2^{k-\ell}$ matched left nodes and more than $2^{k-\ell}$ rejected nodes. Thus we have received more than 2^k requests. □

In particular, every $(2^k, 2^k)$-expander has on-line matching up to 2^{k+1} with at most 2^k rejections. For Theorem 2.4 we will use non-explicit such graphs, for Theorem 2.5 we will need explicit such graphs, which we obtain from the disperser of [TSUZ07].

Assume that for every n and $k < n$ we are given a $(2^k, 2^k)$-expander $G_{n,k}$ with $L = \{0,1\}^n$, $R = \{0,1\}^{k+c(n)}$ and the degree of all left nodes is at most $D(n)$. Assume further that given n, k and a left node x in $G_{n,k}$ we can algorithmically find the list of all neighbors of x in $G_{n,k}$.

Theorem 3.3. *Given a family of expanders as above we can construct a computable graph G with $L = \{0,1\}^*$ that has on-line matching with overhead $c(n) + O(\log n)$ and the degree of each left node is $O(D(n)n)$. The matching strategy for G is computable. Moreover, if given n, k and a left node x in $G_{n,k}$ we can find the list of all neighbors of x in time poly(n) then G is polynomial time computable.*

Proof. The main tool is borrowed from [MRS11]: all the graphs G_{nk} share the same set of left nodes while their sets of right nodes are disjoint. Let H_{nk} denote the union of G_{ni} over all $i < k$. Then H_{nk} has on-line matching up to 2^k (without rejections). Indeed, each input left node is first given to the matching algorithm for $G_{n,k-1}$ (that has on-line matching up to 2^k with at most 2^{k-1} rejections) and, if rejected is given to the matching algorithm for $G_{n,k-2}$ and so on.

Using this construction we can prove the theorem with slightly worse parameters as claimed. To this end identify right nodes of the graph H_{nk} with strings of length $k + c(n) + 1$ (the number of right nodes of H_{nk} does not exceed the sum of geometrical series $2^{k+c(n)} + 2^{k+c(n)-1} + \cdots < 2^{k+c(n)+1}$). The degree of H_{nk} is $D(n)k$.

Recall the connection between matching up to 2^k and matching with overhead (the third paragraph after Definition 3.1). We see that the family H_{nk} can be converted into a graph H_n with $L = \{0,1\}^n$ and degree $D(n)n(n-1)/2 + 1$ having on-line matching with overhead $c(n) + 2$. Finally, prepend each right nodes of H_n by a $O(\log n)$-bit prefix code of the number n and consider the union of all H_n. The resulting graph has on-line matching with overhead $c(n) + O(\log n)$, its set of left nodes is $\{0,1\}^*$ and the degree of every left node of length n is $O(D(n)n^2)$.

Now we will explain how to reduce the degree to $O(D(n)n)$. Consider four copies of $G_{n,k-1}$ with the same set L of left nodes and disjoint sets of right nodes (say append 00 to every right node to get the first copy, 01 to get the second copy and so on). Their union is a $(2^{k-1},2^{k+1})$-expander, and hence has matching up to 2^{k+1} with at most 2^{k-1} rejections. [5] Its left degree is $4D(n)$ and the length of right nodes is $k+c(n)+2$. Replace in the above construction of H_n the graph $H_{n,k}$ by this graph. Thus the left degree of H_n becomes $O(D(n)n)$ in place of $O(D(n)n^2)$. It remains to show that (the union of all graphs) H_n has still on-line matching with overhead $c(n)+O(\log n)$

Again the matching strategy is greedy. Once we receive a request (x,k) with $|x|=n$, we match x to $1x$ if $k\geq n$. Otherwise we pass x to the matching algorithm in $H_{n,k}$. If the algorithm rejects x, we pass x to the matching algorithm in $H_{n,k-1}$ and so on. We claim that we eventually find a match in one of the graphs $H_{n,i}$ for $i\leq k$. To prove the claim it suffices to show that the matching algorithm for $H_{n,k}$ receives at most 2^{k+1} input strings. This is proved by a downward induction on k (for any fixed n). For $k=n-1$ this is obvious: we try to match in $H_{n,n-1}$ up to 2^{n-1} strings. The induction step: by induction hypothesis the matching algorithm for $H_{n,k+1}$ receives at most 2^{k+2} input strings an thus rejects at most 2^k of them. The matching algorithm for $H_{n,k}$ thus receives at most 2^k rejected strings and at most 2^k new ones, coming from requests of the form (x,k). □

3.1 Proof of Theorem 2.4

A weaker form of Theorem 2.4 can be derived from Theorem 3.3 and the following lemma from [Muc02].

Lemma 3.4. *For all n and $k<n$, there exists a $(2^k,2^k)$-expander with $L=\{0,1\}^n$, $R=\{0,1\}^{k+2}$ and all left nodes have degree at most $n+1$.*[6]

Proof. We use the probabilistic method, and for each left node we choose its $n+1$ neighbors at random: all neighbors of each node are selected independently among all 2^{k+2} right nodes with uniform distribution, and the choices for different left nodes are independent too. We show that expansion property is satisfied with positive probability. Hence there exists at least one such graph. To estimate the probability that the property is not satisfied, consider a pair of sets L' and R' of left and right nodes, respectively, of sizes $2^k, 2^k-1$. The probability that the neighbors of all nodes in L' belong to R' is upper-bounded by $(1/4^c)^{(1/c)(n+c)2^k}=(1/4)^{(n+c)2^k}$. The total probability that expansion condition is not satisfied, is obtained by summing over all such L',R', i.e.

$$\left(\frac{1}{4}\right)^{(n+1)2^k}(2^n)^{2^k}(2^{k+2})^{2^k-1}\leq\left(\frac{2^n2^{k+2}}{4^{n+1}}\right)^{2^k}\leq\left(\frac{2^n2^{n+1}}{4^{n+1}}\right)^{2^k}=\left(\frac{1}{2}\right)^{2^k}<1.\quad\square$$

Remark 2. By the very same construction we can obtain a graph with $L=\{0,1\}^n$, $R=\{0,1\}^{k+2}$, $D=n+1$ that is a (t,t)-expander *for all* $t\leq 2^k$. Indeed, the probability that a random graph is not (t,t)-expander is at most $\left(\frac{1}{2}\right)^t$ (we may replace 2^k by t in the above formulas). By union bound, the probability that it happens for some $t\leq 2^k$ is at most the sum of geometric series $\sum_{t=1}^{2^k}\left(\frac{1}{2}\right)^t<1$. By Hall's theorem, this graph has off-line matching up to 2^k. An interesting open question is whether there is a graph with the same parameters, i.e. $L=\{0,1\}^n$, $R=\{0,1\}^{k+O(1)}$, $D=O(n)$, that has *on-line* matching up to 2^k.

[5] One can also consider the union of $G_{n,k-1}$ and $G_{n,k}$, which also has matching up to 2^{k+1} with at most 2^{k-1} rejections.

[6] This older lemma improves a technical result in [MRS11] by replacing $k+O(\log n)$ to $k+2$. Jason Teutsch suggested this improvement could turn list-approximators for $O(\log n)$-short programs to list-approximators for $O(1)$-short programs in a length-conditional setting.

From this lemma and Theorem 3.3 we obtain a computable graph with on-line matching with overhead $O(\log n)$, degree $O(|x|^2)$ and $L = \{0,1\}^*$ We now need to replace the $O(\log n)$ overhead by $O(1)$. Recall the $O(\log n)$ appeared from the prefix code of n added to hash values. To get rid of it we need a computable graph F_k in place of previously used $G_{n.k}$ with the same parameters but with $L = \{0,1\}^{>k}$, and not $L = \{0,1\}^n$. Such a graph is constructed in the following

Lemma 3.5. *For every k there is a computable bipartite graph F_k with $L = \{0,1\}^{\geq k}$, $R = \{0,1\}^{k+3}$ that is a $(2^k, 2^k)$-expander and the degree of every left node x is $O(|x|)$.*

Proof. We first build such a graph with left nodes being all strings of length between k and $K = 2^{k+3}$. This is again done by probabilistic method: we choose $|x|+3$ neighbors of every node x independently. Let L_i stand for all left nodes of length i. For any $i \in k,\dots,K$, the probability that all elements of a fixed $L' \subset L_i$ are mapped to a fixed set of size at most $2^k - 1$ at the right is at most $\left(\frac{1}{2^3}\right)^{(i+3)|L'|}$. The probability that some t_i elements in L_i are mapped into a fixed set of $2^k - 1$ elements at the right is bounded by

$$2^{it_i}\left(\frac{1}{2^3}\right)^{(i+3)t_i} = \left(\frac{1}{2^2}\right)^{it_i}\left(\frac{1}{2^3}\right)^{3t_i} \leq \left(\frac{1}{2^2}\right)^{(i+3)t_i+t_i} \leq \left(\frac{1}{2^2}\right)^{(k+3)t_i+t_i} = \left(\frac{1}{K}\right)^{2t_i}\left(\frac{1}{2}\right)^{2t_i}.$$

If $\sum_{i=k}^{K} t_i = t$ with $t = 2^k$, the probability that the union of neighbors of t_k elements in L_k, t_{k+1} elements in L_{k+1}, ..., and t_K elements in L_K are mapped to a fixed set of size at most $2^k - 1$ is bounded by $\prod_i \left(\frac{1}{2K}\right)^{2t_i} = \left(\frac{1}{2K}\right)^{2t}$. Multiplying by the number $K^{2^k-1} \leq K^t$ of different right sets of size $K-1$, and multiplying by the number K^t of different solutions to the equation $\sum_{i=k}^{K} t_i = K$, we find

$$\left(\frac{1}{2K}\right)^{2t} K^t\, K^t \leq \left(\frac{1}{4}\right)^t < 1.$$

Hence, the total probability to randomly generate a graph that is not an expander is strictly less than 1. Therefore, a graph satisfying the conditions must exist, and can be found by exhaustive search.

On the left side, we now need to add the strings of length larger than $K = 2^{k+3}$. These nodes are connected to all the nodes on the right side. Thus the degree of every such node x is $2^k \leq |x|/2^3 \leq O(|x|)$ and we are done. □

Remark 3. By the very same construction we can obtain a graph with $L = \{0,1\}^{\geq k}$, $R = \{0,1\}^{k+3}$, $D = O(n)$ that is a (t,t)-expander *for all* $t \leq 2^k$ and thus has off-line matching up to 2^k (use the union bound over all $t \in \{k,\dots,K\}$). An interesting open question is whether there is a graph with the same parameters that has *on-line* matching up to 2^k.

Now we can finish the proof of Theorem 2.4. Appending all 2-bit strings to all the right nodes of the graph F_{k-1} (and thus increasing the degree 4 times) we obtain a $(2^{k-1}, 2^{k+1})$-expander H_k. The union of H_k over all k is a computable graph, whose left degree is $O(|x|^2)$, and the set of left nodes is $\{0,1\}^*$. It has on-line matching with constant overhead. This is proved by the downward induction, as in Theorem 3.3. Indeed, by step s in the matching, there are only finitely many requests for matching. By a downward induction on k we can again prove that the number of matching requests in H_k is at most 2^{k+1}. Now the base of induction is the maximal k for which there has been at least one request for matching in H_k up to step s. We conclude that the request made at step s is satisfied and, since this holds for every s, we are done.

3.2 Proof of Theorem 2.5

By Theorem 3.3 we have to construct for every $k \le n$ an explicit $(2^k, 2^k)$-expander of left degree $\text{poly}(n)$, with 2^n left nodes and $\text{poly}(n)2^k$ right nodes. A graph is *explicit* if there is an algorithm that on input $x \in \{0,1\}^n = L$ lists in $\text{poly}(n)$ time all the neighbors of x.

The proof relies on the explicit disperser graphs of Ta-Shma, Umans, and Zuckerman from Theorem 3.7 below.

Definition 3.6. A bipartite graph $G = (L, R, E)$ is a (K, δ)-*disperser*, if every subset $B \subseteq L$ with $|B| \ge K$ has at least $(1-\delta)|R|$ distinct neighbors.

Theorem 3.7. *[Ta-Shma, Umans, Zuckerman [TSUZ07]] 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 $D = poly(n)$ and $|R| = \frac{\alpha KD}{n^3}$, for some constant α.*

Given n and k we apply this theorem to $K = 2^k$ and $\delta = 1/2$. We obtain a $(2^k, \frac{\alpha 2^k D}{2n^3})$-expander with degree $D = \text{poly}(n)$, $L = \{0,1\}^n$ and $|R| = \frac{\alpha KD}{n^3}$. Consider $t = \max\{1, \lceil \frac{2n^3}{\alpha D} \rceil\}$ disjoint copies of this graph and identify left nodes of the resulting graphs (keeping their sets of right nodes disjoint). We get an explicit $(2^k, 2^k)$-expander with 2^n left and $2^k\text{poly}(n)$ right nodes and degree $\text{poly}(n)t = \text{poly}(n)$.

4 The lower bounds

4.1 Proof of Theorem 2.6

Assume that G has off-line matching with overhead c. Let $G[\ell, k]$ denote the induced graph that is obtained from G by removing all right nodes of length more than k or less than ℓ. The graph $G[0, k+c]$ is obviously a $(2^k, 2^k)$-expander for every k. As there are less than 2^{k-1} strings of length less than $k-1$, it follows that the graph $G[k-1, k+c]$ is a $(2^k, 2^{k-1}+1)$-expander.

The next lemma inspired by [KST54] (see [RTS00, Theorem 1.5]) shows that any such expander must have large degree.

Lemma 4.1. *Assume that a bipartite graph with 2^ℓ left nodes and 2^{k+c} right nodes is a $(2^k, 2^{k-1}+1)$-expander. Then there is a left node in the graph with degree more than $D = \min\{2^{k-2}, (\ell-k)/(c+2)\}$.*

Proof. For the sake of contradiction assume that all left nodes have degree at most D (and w.l.o.g. we may assume that all degrees are exactly D). We need to find a set of right nodes B of size 2^{k-1} and 2^k left nodes all of whose neighbors lie in B. The set B is constructed via a probabilistic construction. Namely, choose B at random (all $\binom{2^{k+c}}{2^{k-1}}$ sets have equal probabilities). The probability that all neighbors of a fixed left node are in B is equal to

$$\frac{\binom{2^{k+c}-D}{2^{k-1}-D}}{\binom{2^{k+c}}{2^{k-1}}} = \frac{2^{k-1}(2^{k-1}-1)\cdots(2^{k-1}-D+1)}{2^{k+c}(2^{k+c}-1)\cdots(2^{k+c}-D+1)}.$$

Both products in the numerator and denominator have D factors and the ratio of corresponding factors is at least

$$\frac{2^{k-1}-D+1}{2^{k+c}-D+1} \ge 2^{-c-2}$$

(the last inequality is due to the assumption $D \le 2^{k-2}$). Thus the probability that all neighbors of a fixed left node are in B is at least $2^{-D(c+2)}$. Hence the average number of left nodes having this property is at least $2^{\ell-D(c+2)}$, which is greater than or equal to 2^k by the choice of D. Hence there is B that includes neighborhoods of at least 2^k left nodes, a contradiction. □

This lemma states that at least one left node has large degree. However, it implies more: if the number of left nodes is much larger than 2^ℓ, then almost all left nodes must have large degree. Indeed, assume that a bipartite graph with 2^{k+c} right nodes is a $(2^k, 2^{k-1}+1)$-expander. Choose 2^ℓ left nodes with smallest degree and apply the lemma to the resulting induced graph (which is also a $(2^k, 2^{k-1}+1)$-expander). By the lemma, in the original graph all except for less than 2^ℓ nodes have degree more than $D = \min\{2^{k-2}, (\ell-k)/(c+2)\}$.

Choose $n/4 < k \leq n/2$. As noticed, the graph $G[k-1, k+c]$ is a $(2^k, 2^{k-1}+1)$-expander and has less than 2^{k+c+1} right nodes. By Lemma 4.1 (applied to $\ell = 3n/4$, k, $c+1$), all except for at most $2^{3n/4}$ left nodes of $G[k-1, k+c]$ have degree at least $n/4(c+3)$.

Pick now ℓ different k's that are $c+2$ apart of each other, where ℓ is about $n/(4(c+2))$. For most left nodes for all picked k there are $n/4(c+3)$ edges from those nodes in $G[k-1, k+c]$. As all picked k's are $c+2$ apart of each other, the degree of all those nodes is $\Omega(n^2/(c+3)^2)$.

4.2 Proof of Theorem 1.4.

The size of list-approximators is closely related to total conditional Kolmogorov complexity, which was first introduced by A. Muchnik and was used in [Ver09, Bau10]. *Total conditional Kolmogorov complexity* with respect to U is defined as:

$$CT_U(u|v) = \min\{|q| : U(q,v) = u \wedge \forall z [U(q,z)\downarrow]\},$$

where $U(q,z)\downarrow$ means that $U(q,z)$ halts. If U is a standard machine then $CT_U(u|v) \leq CT_V(u|v) + c_V$ for every machine V. The connection to list-approximators is the following:

Lemma 4.2. *If f is computable function that maps every string to a finite list of strings then $CT_U(p|x) \leq \log|f(x)| + O(1)$ for any standard machine U and every p in $f(x)$. The constant in O-notation depends on f and U.*

Proof. Let $V(j,x)$ stand for the jth entry of the list $f(x)$, if $j \leq |f(x)|$, and for the empty string (say) otherwise. Obviously $CT_V(p|x) \leq \log|f(x)|$ for all p in $f(x)$. Hence $CT_U(p|x) \leq \log|f(x)| + O(1)$. □

Thus to prove the theorem it suffices to construct a standard machine U_0 such that for infinitely many x every 0-short p for x with respect to U_0 satisfies $CT_U(p|x) \geq |x| - O(1)$. To this end we fix any standard machine U and construct another machine V such that for some constant d and for every integer k there are strings p, x such that:
(a) p is the unique 0-short program for x with respect to V,
(b) $C_U(x) \geq k$,
(c) $|x| = |p| = k + d$,
(d) $CT_U(0p|x) \geq k$.

Once such V has been constructed, we let $U_0(0q|z) = V(q|z)$ and $U_0(1^{d+2}q|z) = U(q|z)$. The latter equality guarantees that U_0 is a standard machine. And both equalities together with items (a), (b) and (c) imply that $0p$ is the unique 0-short program for x with respect to U_0. Finally, item (d) guarantees that its total complexity conditional to x is at least $|x| - d - 1$.

The construction of V can be described in game terms. (The game-based technique in recursion theory was introduced by Lachlan [Lac70] and further developed by A.Muchnik and others [MMSV10, Ver08, MSV12].)

Description of the game. The game has integer parameters k,d and is played on a rectangular grid with 2^{k+d} rows and 2^{k+d} columns. The rows and columns are identified with strings of length $k+d$. Two players, Black and White, play in turn. In her turn White can either pass or put a pawn on the board. White can place at most one pawn in each row and at most one pawn in each column. Once a pawn is placed, it can not be moved nor removed. In his turn Black can either pass, or choose a column and *disable* all its cells, or choose at most one cell in every column and *disable* all of them. If a player does not pass, we say that she/he *makes a move*. Black is allowed to make less that 2^{k+1} moves. The game is played for an infinite time and White looses if at some point after her turn, all her pawns are in disabled cells.

We will show that, for $d=3$, for every k, White wins this game. More specifically, there is a winning strategy for White that is uniformly computable given k. Assume that this is done. Then consider the following "blind" strategy for Black: start enumeration of all strings x with $C_U(x)<k$ and all strings q of length less than k such that $U(q,x)\downarrow$ for all x of length $k+d$. That enumeration can be done uniformly in k. In his tth turn Black: disables all cells in the xth column, if on step t in this enumeration a new x of length $k+d$ with $C_U(x)<k$ appears; disables all cells (p,x) with $|x|=|p|=k+d$, $U(q,x)=0p$, if on step t a new string q of length less than k appears such that $U(q,x)\downarrow$ for all x of length $k+d$; and passes if none of these events occurs. Note that the total number of Black's moves is less that $2^k+2^k=2^{k+1}$, as required.

Now consider the following machine $V(p)$: let $k=|p|-d$ and let the White's computable winning strategy play against Black's blind strategy. Watch the play waiting until White places a pawn on a cell (p,x) in pth row. Then output x and halt. Note that such x is unique (if exists), as White places at most one pawn in each row. And that cell (p,x) satisfies all the requirements (a)–(d). Thus it suffices to design a computable winning strategy for White.

A winning White's strategy. The strategy is a greedy one. In the first round White places a pawn in any cell. Then she waits until that cell becomes disabled. Then she places the second pawn in any enabled cell that lies in another row and another column and again waits until that cell becomes disabled. At any time she chooses any enabled cell that lies in a row and a column that both are free of pawns. In order to show that White wins, we just need to prove that there is such cell. Indeed, Black makes less than 2^{k+1} moves, thus White makes at most 2^{k+1} moves. On each of Black's moves at most 2^{k+d} cells become disabled. On each of White's moves at most 2^{k+d+1} cells becomes non-free because either their column or row already has a pawn. Thus if the total number of cells is more than

$$2^{k+1}2^{k+d}+2^{k+1}2^{k+d+1}=6\cdot 2^{2k}2^{d},$$

we are done. The total number of cells is $2^{k+d}2^{k+d}=2^{2k}2^{2d}$. As 2^{2d} grows faster that $6\cdot 2^{d}$, for large enough d (actually for $d=3$) the total number of cells is larger than the number of disabled or non-free cells. The theorem is proved.

Note that changing the construction a little bit we can prove the same statement for c-short programs for every c. To this end we just need to let $U_0(1^{c+d+2}q)=U(q)$ instead of $U_0(1^{d+2}q)=U(q)$. The optimal machine U_0 constructed in this way depends on the choice of c, which is inevitable by Theorem 1.1.

5 Other applications of explicit graphs with on-line matching

The two first applications are related with the resource bounded Kolmogorov complexity. Recall a machine U is called standard if for any machine V there is a total computable function f such that $U(t(p),z) = V(p,z)$ and $|t(p)| \le |p| + O(1)$ for all p, z. In this section we assume that t is polynomial time computable and that running time of $U(t(p),z)$ is bounded by a polynomial of the computation time of $V(p,z)$. By $C_U^T(x|z)$ we denote the minimal length of p such that $U(p,z) = x$ in at most T steps.

Muchnik's Theorem [Muc02, MRS11]. Let a and b be strings such that $|a| = n$ and $C(a \mid b) = k$. Then there exists a string p such that (1) $|p| = k + O(\log n)$, (2) $C(p \mid a) = O(\log n)$, (3) $C(a \mid p, b) = O(\log n)$.

In our improved version, we replace (2) by (2') $C^{q(n)}(p \mid a) = O(\log n)$, where q is a polynomial.

Proof. Fix an explicit graph with $L = \{0,1\}^*$, polynomial degree, and that has computable on-line matching with logarithmic overhead. Given a string b run the optimal machine $U(q,b)$ in parallel for all q and once for some q, $U(q,b)$ halts with the result x pass the request $(x, |q|)$ to the matching algorithm in the graph. It will return a neighbor p of length at most $|q| + O(\log n)$ of x. At some moment a shortest program q for a conditional to b will halt and we get the sought p.

As the graph is explicit and has polynomial degree, we have $C^{\mathrm{poly}(n)}(p \mid a) = O(\log n)$ (requirement (2)). Requirement (1) holds by construction. Finally, $C(a \mid p, b) = O(\log n)$ as given p and b we may identify a by running the above algorithmic process (it is important that a is the unique string that was matched to p). □

Distinguishing complexity [BFL01].

Let V be a machine, x a string and T a natural number. The *distinguishing complexity* $CD_V^T(x)$ *with respect to* V is defined as the minimal length of p such that $U(p,x) = 1$ (p "accepts" x) in at most T steps, and $U(p,x') = 0$ for all $x' \ne x$ (p "rejects" all other strings). From our assumption for the standard machine U it follows that for every machine V there is a polynomial f and a constant c such that $CD_U^{f(T)}(x) \le CD_V^T(x) + c$. Indeed, let p is a shortest distinguishing program for x working in T steps with respect to V. Then $t(p)$ is a program for U that accepts x in $\mathrm{poly}(T)$ steps and rejects all other strings.

For a set A of binary strings let $A^{=n}$ stand for the set of all strings of length n in A.

Theorem 5.1 ([BFL01]). *For every function $\varepsilon(n)$ (mapping natural numbers to numbers of the form $1/$natural) computable in time poly(n) there is a polynomial f such that for every set A, for all $x \in A^{=n}$ except for a fraction $\varepsilon(n)$, $CD_U^{f(n),A}(x) \le \log|A^{=n}| + \mathrm{polylog}(n/\varepsilon(n))$.*

We mean here that the set A is given to the standard machine U as an oracle (so we assume that the standard machine is an oracle machine and all the requirements hold for every oracle.)

In our improved version, we obtain $CD^{f(n),A}(x) \le \log|A^{=n}| + O(\log(n/\varepsilon(n)))$.

Proof. For our improvement we need for every n, $k \le n$ and ε a bipartite graph $G_{n,k,\varepsilon}$ with $L = \{0,1\}^n$, $R = \{0,1\}^{k+O(\log n/\varepsilon)}$ and degree $\mathrm{poly}(n/\varepsilon)$ that has the following property:

> for every subset S of at most 2^k left nodes for every node x in S except for a fraction ε there is a right neighbor p of x such that p has no other neighbors in S.

Assume that we have such an explicit family of graphs $G_{n,k,\varepsilon}$. Explicit means that given n, k, ε, a left node x and i we can in polynomial time find the ith neighbor of x in $G_{n,k,\varepsilon}$. Then we can construct a machine V that given a tuple (p,i,n,k), a string x and A as oracle verifies that x is in $A^{=n}$ and that x is the i-th neighbor of p in $G_{|x|,k,\varepsilon(n)}$. If this is the case it accepts and rejects otherwise. By the property of the graph, applied to $S=A^{=n}$ and $k=\lceil\log|S|\rceil$ we see that

$$CD_V^{f(n),A}(x) \leq |(p,i,n,k)| \leq \log|A^{=n}| + O(\log n/\varepsilon(n))$$

for some polynomial $f(n)$ for all but a fraction $\varepsilon(n)$ for $x \in A^{=n}$. By the assumptions on U the same inequality holds for U.

The graph $G_{n,k,\varepsilon}$ is again obtained from the disperser of [TSUZ07]. Given n, k and ε we apply Theorem 3.7 to $K=2^k\varepsilon$ and $\delta=1/2$. We obtain a $(K,\frac{\alpha KD}{2n^3})$-expander with degree $D=\text{poly}(n)$, $L=\{0,1\}^n$ and $|R|=\frac{\alpha KD}{n^3}$. Consider $t=\max\{1,\lceil\frac{2n^3}{\alpha D}\rceil\}$ disjoint copies of this graph and identify left nodes of the resulting graphs (keeping their sets of right nodes disjoint). We get an explicit $(2^k\varepsilon,2^k\varepsilon)$-expander with 2^n left and $2^k\text{poly}(n)/\varepsilon$ right nodes and degree $D=\text{poly}(n)t/\varepsilon=\text{poly}(n)/\varepsilon$.

This graph, called $H_{n,k,\varepsilon}$, has the following "low-congestion property": *for every set of* 2^k *left nodes* S *for every node* x *in* S *except for a fraction* ε *there is a right neighbor* p *of* x *such that* p *has at most* D/ε *neighbors in* S.

Indeed, the total number of edges in the graph originating in S is at most $|S|D$. Thus less than $|S|D/(D/\varepsilon)=|S|\varepsilon$ right nodes are "fat" in the sense that they have more than D/ε neighbors landing in S. By the expander property of $H_{n,k,\varepsilon}$ there are less than $\varepsilon|S|$ left nodes in S that have only fat neighbors.

It remains to "split" right nodes of $H_{n,k,\varepsilon}$ so that D/ε becomes 1. This is done exactly as in [BFL01]. Using the Prime Number Theorem, it is not hard to show (Lemma 3 in [BFL01]) that for every set W of d strings of length n the following holds: *for every* $x\in W$ *there is a prime number* $q\leq 4dn^2$ *such that* $x\not\equiv x' \pmod q$ *for all* $x'\in W$ *different from* x (we identify here natural numbers and their binary expansions).

We apply this lemma to $d=D/\varepsilon$. To every right node p in $H_{n,k,\varepsilon}$ we add a prefix code of two natural numbers a,q, both at most $4dn^2$, and connect a left node x to (p,a,q) if x is connected to p in $H_{n,k,\varepsilon}$ and $x\equiv a \pmod q$. We obtain the graph $G_{n,k,\varepsilon}$ we were looking for. Indeed, for every S of 2^k left nodes for all $x\in S$ but a fraction of ε there is a neighbor p of x in $H_{n,k,\varepsilon}$ that has at most $d=D/\varepsilon=\text{poly}(n)/\varepsilon$ neighbors in S. Besides there is a prime $q\leq 4n^2d=\text{poly}(n)/\varepsilon$ such that $x\not\equiv x' \pmod q$ for all neighbors x' of p different from x. Thus the neighbor $(p,q,x \bmod q)$ of x in $G_{n,k,\varepsilon}$ has no other neighbors in S.

The degree of $G_{n,k,\varepsilon}$ is $D\times(4n^2D/\varepsilon)^2=\text{poly}(n)/\varepsilon^2$. The number of right nodes is

$$(\text{poly}(n)2^k/\varepsilon)(4n^2D/\varepsilon)^2 = 2^k\text{poly}(n)/\varepsilon^3.$$

Thus right nodes can be identified with strings of length $k+O(\log n/\varepsilon)$ and we are done. □

6 Acknowledgments

We are grateful to Andrei Romashchenko, Sasha Shen and Jason Teutsch for helpful discussions.

References

[Bau10] B. Bauwens. *Computability in statistical hypotheses testing, and characterizations of independence and directed influences in time series using Kolmogorov complexity*. PhD thesis, Ugent, May 2010.

[Bau12] B. Bauwens. Complexity of complexity and maximal plain versus prefix-free Kolmogorov complexity. In Artur Czumaj, Kurt Mehlhorn, Andrew M. Pitts, and Roger Wattenhofer, editors, *ICALP (1)*, volume 7391 of *Lecture Notes in Computer Science*, pages 100–108. Springer, 2012.

[BBF$^+$06] R. Beigel, H.M. Buhrman, P. Feijer, L. Fortnow, P. Grabowski, L. Longpre, A. Muchnik, F. Stephan, and L. Torenvliet. Enumerations of the Kolmogorov function. *Journal of Symbolic Logic*, 7(501):501 – 528, 2006.

[BFL01] Harry Buhrman, Lance Fortnow, and Sophie Laplante. Resource-bounded Kolmogorov complexity revisited. *SIAM J. Comput.*, 31(3):887–905, 2001.

[CRVW02] M. R. Capalbo, O. Reingold, S. P. Vadhan, and A. Wigderson. Randomness conductors and constant-degree lossless expanders. In John H. Reif, editor, *STOC*, pages 659–668. ACM, 2002.

[Gác74] P. Gács. On the symmetry of algorithmic information. *Soviet Math. Dokl.*, 15:1477–1480, 1974.

[Hal35] P. Hall. On representatives of subsets. *J. London Math. Soc.*, 1(10):26–30, 1935.

[KST54] T. Kovåri, V.T. Sòs, and P. Turàn. On a problem of K. Zarankiewicz. *Colloquium Math*, 3:50–57, 1954.

[Lac70] A. H. Lachlan. On some games which are relevant to the theory of recursively enumerable sets. *Annals of Mathematics*, 91(2):291–310, 1970.

[MMSV10] A. A. Muchnik, I. Mezhirov, A. Shen, and N. Vereshchagin. Game interpretation of Kolmogorov complexity. //arxiv.org/abs/1003.4712.

[MRS11] D. Musatov, A. E. Romashchenko, and A. Shen. Variations on Muchnik's conditional complexity theorem. *Theory Comput. Syst.*, 49(2):227–245, 2011.

[MSV12] A. Muchnik, A. Shen, and M. Vyugin. Game arguments in computability theory and algorithmic information theory. *ArXiv e-prints*, April 2012.

[Muc02] Andrei A. Muchnik. Conditional complexity and codes. *Theor. Comput. Sci.*, 271(1-2):97–109, 2002.

[RSW06] O. Reingold, R. Shaltiel, and A. Wigderson. Extracting randomness via repeated condensing. *SIAM J. Comput.*, 35(5):1185–1209, 2006.

[RTS00] J. Radhakrishnan and A. Ta-Shma. Tight bounds for dispersers, extractors, and depth-two superconcentrators. *SIAM Journal on Discrete Mathematics*, 13(1):2–24, February 2000.

[Sch75] C.P. Schnorr. Optimal enumerations and optimal Gödel numberings. *Mathematical Systems Theory*, 8(2):182–191, 1975.

[TS96] A. Ta-Shma. On extracting randomness from weak random sources. In *Proceedings of the 26th ACM Symposium on Theory of Computing*, pages 276–285, 1996.

[TSUZ07] A. Ta-Shma, C. Umans, and D. Zuckerman. Lossless condensers, unbalanced expanders, and extractors. *Combinatorica*, 27(2):213–240, 2007.

[TVZ05] L. Trevisan, S. P. Vadhan, and D. Zuckerman. Compression of samplable sources. *Computational Complexity*, 14(3):186–227, 2005.

[T11] J. Teutsch. Short lists for shorter programs in short time. http://arxiv.org/abs/1212.6104

[Ver08] N. Vereshchagin. Kolmogorov complexity and games. *Bulletin of the European Association for Theoretical Computer Science*, 94:51–83, Feb 2008.

[Ver09] N. Vereshchagin. Algorithmic minimal sufficient statistics: a new definition. Presented on the 4th conference on randomness, computability and logic, Luminy., nov 2009.

[Zim11] M. Zimand. Symmetry of information and bounds on nonuniform randomness extraction via Kolmogorov extractors. In *IEEE Conference on Computational Complexity*, pages 148–156. IEEE Computer Society, 2011.

[Zim12] M. Zimand. Nonuniform Kolmogorov extractors. *CoRR*, abs/1204.6696, 2012.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de