Improving the Space-Bounded Version of Muchnik's Conditional Complexity Theorem via "Naive" Derandomization[⋆]

Daniil Musatov

Lomonosov Moscow State University, `musatych@gmail.com`

Abstract. Many theorems about Kolmogorov complexity rely on existence of combinatorial objects with specific properties. Usually the probabilistic method gives such objects with better parameters than explicit constructions do. But the probabilistic method does not give "effective" variants of such theorems, i.e. variants for resource-bounded Kolmogorov complexity. We show that a "naive derandomization" approach of replacing these objects by the output of Nisan-Wigderson pseudo-random generator may give polynomial-space variants of such theorems.
Specifically, we improve the preceding polynomial-space analogue of Muchnik's conditional complexity theorem. I.e., for all a and b there exists a program p of least possible length that transforms a to b and is simple conditional on b. Here all programs work in polynomial space and all complexities are measured with logarithmic accuracy instead of polylogarithmic one in the previous work.

1 Introduction

Many statements about Kolmogorov complexity may be proven by applying some combinatorial constructions like expanders or extractors. Usually these objects are characterized by some parameters, and one may say which parameters are "better". Very often the probabilistic method allows to obtain these objects with much better parameters than explicit constructions do. But exploiting the probabilistic method causes exponential-space brute-force search of an object satisfying the necessary property. And if this search is performed while describing some string to obtain an upper bound on its complexity then this bound cannot be repeated for polynomial-space complexity. On the other hand, replacing the probabilistic method by an explicit construction weakens the statement due to worse parameters.

We present a technique that combines advantages of both probabilistic and explicit-construction methods. The key idea is substitution for a random object with pseudo-random one obtained by Nisan-Wigderson generator and still possessing necessary property. We employ indistinguishability of Nisan-Wigderson generator's output from a random string by boolean circuits of constant depth

---

and polynomial size. If the necessary property can be tested by such circuit then it holds for a pseudo-random object as well as for a truly random object. Unfortunately, it is not clear how to build such a circuit for extractor property and similar ones, so we relax the property in a way that allows both proving the theorem and building polynomial constant-depth circuits.

By exploiting our method we improve previous result [10] generalizing Muchnik's conditional complexity theorem. The original theorem [7] states that for all a and b of length n there exists a program p of length $C(a|b)+O(\log n)$ that transforms b to a and has complexity $O(\log n)$ conditional on a. In [10] this result is restated for space-bounded complexity with gap rised from $O(\log n)$ to $O(\log^3 n)$. The main idea was to employ a property of extractors, proven in [2]: in extractor graph in every sufficiently big subset S of left part there are few vertices with all right-part neighbours having indegree from S twice greater than average. We refer to it as to "low-congesting property". An explicit extractor construction yields a space-bounded version of the theorem with polylogarithmic precision. In this paper we replace explicit extractor by pseudo-random graph, that does not necessary have the extractor property, but enjoys the low-congesting property for "relevant" subsets S. This replacement leads to decreasing the precision back to logarithmic term.

The rest of the paper is organized as follows. In Sect. 2 we give formal definitions of all involved objects and formulate necessary results. In Sect. 3 we formally state our space-bounded variant of Muchnik's theorem and specify all details of the proof.

2 Preliminaries

2.1 Kolmogorov complexity

Let $\mathcal{V}$ be a two-argument Turing machine. We refer to the first argument as to the "program" and to the second argument as to the "argument". (Plain) Kolmogorov complexity of a string x conditional on y with respect to $\mathcal{V}$ is the length of a minimal program p that transforms y to x, i.e.

$$C_\mathcal{V}(x \mid y) = \min\{p \colon \mathcal{V}(p, y) = x\}$$

There exists an optimal machine $\mathcal{U}$ that gives the least complexity up to an additive term. Specifically, $\forall \mathcal{V} \exists c \forall x, y \, C_\mathcal{U}(x|y) < C_\mathcal{V}(x|y) + c$. We employ such a machine $\mathcal{U}$, drop the subscript and formulate all theorems up to a constant additive term. The unconditional complexity $C(x)$ is the complexity with empty condition $C(x \mid \varepsilon)$, or the length of a shortest program *producing* x.

Now we define the notion of resource-bounded Kolmogorov complexity. Roughly speaking, it is the length of a minimal program that transform y to x efficiently. Formally, Kolmogorov complexity of a string x conditional on y in time t and space s with respect to $\mathcal{V}$ is the length of a shortest program p such that $\mathcal{V}(p, y) = x$, and $\mathcal{V}(p, y)$ works in t steps and uses s cells of memory. This complexity is denoted by $C_\mathcal{V}^{t,s}(x \mid y)$. Here the choice of $\mathcal{V}$ alters not only complexity, but also time and space bounds. Specifically, the following theorem holds:

Theorem 1. *There exist a machine $\mathcal{U}$ such that for any machine $\mathcal{V}$ there exists a constant c such that for all x, y, s and t it is true that $C_{\mathcal{U}}^{s,t}(x \mid y) \le C_{\mathcal{V}}^{cs, ct \log t}(x \mid y) + c$.*

In our paper we deal only with space bounds, so we drop the time-bound superscript in all notations.

2.2 Extractors

An extractor is a function that extracts randomness from weak random sources. A k-weak random source of length n is a probabilistic distribution on $\{0,1\}^n$ with minentropy greater than k, that is, no particular string occurs with probability greater than 2^{-k}. An extractor with parameters n, m, d, k, ε is a function $\mathrm{Ext}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$, such that for any independent k-weak random source x of length n and uniform distribution u on $\{0,1\}^d$ the induced distribution $\mathrm{Ext}(x,u)$ on $\{0,1\}^m$ is ε-close to uniform, that is, for any set $Y \subset \{0,1\}^m$ its probability differs from its proportion by at most ε. This is interpreted as follows: an extractor receives n weakly random bits and d truly random bits independent from the first argument and outputs m almost random bits.

Like any two-argument function, an extractor may be viewed as a bipartite (multi-)graph: the first argument indexes a vertex in the left part, the second argument indexes an edge going from this vertex, and the value indexes a vertex in the right part which this edge directs to. That is, the graph has $N = 2^n$ vertices on the left, $M = 2^m$ vertices on the left, and all left-part vertices have degree $D = 2^d$. Throughout the paper, we say that a bipartite graph has parameters (n, m, d), if the same holds for it. For the sake of clarity we usually omit these parameters in extractor specification. The extractor property may be formulated in graph terms: for any left-part subset S of size greater than $K = 2^k$ and any right-part subset Y the fraction of edges directing from S to Y among all edges directing from S differs from the fraction of Y among all right-part vertices by at most ε. A proof of equivalence may be found in [13].

It is proven by the probabilistic method (see, for example, [13]) that for all n, k and ε there exists an extractor with parameters $d = \log(n - k) + 2\log(1/\varepsilon) + O(1)$ and $m = k + d - 2\log(1/\varepsilon) - O(1)$. Nevertheless, no explicit (that is, running in polynomial time) construction of such an extractor is known. Best current results ([14], [16]) for $m \approx k$ use $d = \mathrm{polylog}\, n$ truly random bits.

2.3 Nisan-Wigderson generators

The Nisan-Wigderson pseudo-random generator is a deterministic polynomal-time function that generates n pseudorandom bits from $\mathrm{polylog}(n)$ truly random bits. The output of such generator cannot be distinguished from truly random string by small circuits. Specifically, we exploit the following theorem from [12]:

Theorem 2. *For any constant d there exists a family of functions $G_n\colon \{0,1\}^k \to \{0,1\}^n$, where $k = O(\log^{2d+6} n)$, such that two properties hold:*

Computability: *G is computable in workspace* $\mathrm{poly}(k)$;

Indistinguishability: *For any family of circuits C_n of size* $\mathrm{poly}(n)$ *and depth d for any positive polynomial p for all large enough n it holds that:*

$$|\mathrm{Prob}_x\{C_n(G_n(x)) = 1\} - \mathrm{Prob}_y\{C_n(y) = 1\}| < \frac{1}{p(n)},$$

where x is distributed uniformly in $\{0,1\}^k$ and y — in $\{0,1\}^n$.

By rescaling parameters we get the following

Corollary 1. *There exists a family of functions $G_n\colon \{0,1\}^k \to \{0,1\}^N$, where $k = \mathrm{poly}(n)$ and $N = 2^{O(n)}$, such that two properties hold:*

- *G is computable in workspace* $\mathrm{poly}(n)$;
- *For any family of circuits C_n of size $2^{O(n)}$ and constant depth, for any constant c and for all large enough n it holds that:*

$$|\mathrm{Prob}_x\{C_n(G_n(x)) = 1\} - \mathrm{Prob}_y\{C_n(y) = 1\}| < 2^{-cn}.$$

The last corollary implies the following basic principle:

Lemma 1. *Let $\mathcal{C}_n$ be some set of combinatorial objects encoded by boolean strings of length $2^{O(n)}$. Let $\mathcal{P}$ be some property satisfied for fraction at least α of objects in $\mathcal{C}_n$ that can be tested by a family of circuits of size $2^{O(n)}$ and constant depth. Then for sufficiently large n the property $\mathcal{P}$ is satisfied for fraction at least $\alpha/2$ of values of G_n, where G_n is the function from the previous corollary.*

2.4   Constant-depth circuits for approximate counting

It is well-known that constant-depth circuits cannot compute the majority function. All the more they cannot compute a general threshold function that equals 1 if and only if the fraction of 1's in its input exceeds some threshold α. Nevertheless, one can build such circuits that compute threshold functions approximately. Namely, the following theorem holds:

Theorem 3 ([1]). *Let $\alpha \in (0,1)$. Then for any (constant) ε there exists a constant-depth and polynomial-size circuit C such that $C(x) = 0$ if the fraction of 1's in x is less than $\alpha - \varepsilon$ and $C(x) = 1$ if the fraction of 1's in x is greater than $\alpha + \varepsilon$.*

Note that nothing is promised if the fraction of 1's is between $\alpha - \varepsilon$ and $\alpha + \varepsilon$. So, the fact that $C(s) = 0$ guarantees only that the fraction of 1's is at most $\alpha + \varepsilon$, and $C(s) = 1$ — that it is at least $\alpha - \varepsilon$.

3 Muchnik's theorem

3.1 Subject overview

An. Muchnik's theorem [7] on conditional Kolmogorov complexity states that:

Theorem 4. *Let a and b be two binary strings such that $\mathrm{C}(a) < n$ and $\mathrm{C}(a|b) < k$. Then there exists a string p such that*
- $\mathrm{C}(a|p, b) = O(\log n)$;
- $\mathrm{C}(p) \leq k + O(\log n)$;
- $\mathrm{C}(p|a) = O(\log n)$.

The constants hidden in $O(\log n)$ do not depend on n, k, a, b, p.

Informally, this theorem says that there exists a program p that transforms b to a, has the minimal possible complexity $\mathrm{C}(a|b)$ (up to a logarithmic term) and, moreover, can be easily obtained from a. (The last requirement is crucial, otherwise the statement trivially reformulates the definition of conditional Kolmogorov complexity.)

Several proofs of this theorem are known. All of them rely on the existence of some combinatorial objects. The original proof in [7] leans upon the existence of bipartite graphs with specific expander-like property. Two proofs by Musatov, Romashchenko and Shen [10] use extractors and graphs allowing large on-line matchings. Explicit constructions of extractors provide variants of Muchnik's theorem for resource-bounded Kolmogorov complexity. Specifically, the following theorem is proven in [10]:

Theorem 5. *Let a and b be binary strings of length n, and k and s be integers such that $\mathrm{C}^s(a|b) < k$. Then there exists a binary string p, such that*

- $\mathrm{C}^{O(s)+\mathrm{poly}(n)}(a|p, b) = O(\log^3 n)$;
- $\mathrm{C}^s(p) \leq k + O(\log n)$;
- $\mathrm{C}^{\mathrm{poly}(n)}(p|a) = O(\log^3 n)$,

where all constants in O- and poly*-notation depend only on the choice of the optimal description method.*

Application of our derandomization method decreases conditional complexities from $O(\log^3 n)$ to $O(\log n) + O(\log \log s)$ but increases the space limit in the last complexity from $\mathrm{poly}(n)$ to $O(s) + \mathrm{poly}(n)$.

3.2 Proof overview

Before we proceed with the detailed proof, let us present its high-level description. The main idea is the same in all known proofs: p is a fingerprint (or hash value) for a constructed in some specific way. This fingerprint is chosen via an underlying bipartite graph, the left part of which is treated as the set of length-n strings (i.e., all possible a's) and the right part of which is treated as the set of all possible fingerprints. To satisfy the last condition each left-part vertex should

have small outdegree, since in that case the fingerprint is described by its number among a's neighbours. To satisfy the first condition each fingerprint should have small indegree from the strings that have low complexity conditional on b (for arbitrary b).

If resources are unbounded then the existence of a graph satisfying all conditions may be proven by the probabilistic method and the graph itself may be found by brute-force search. In the resource-bounded case we suggest to replace a random graph by a pseudo-random one and prove that it still does the job. The proof proceeds in several steps. Firstly, in Sect. 3.3 we define the essential graph property needed to our proof. We call this property low-congestion. It follows from the extractor property, but not vice versa. Secondly, in Sect. 3.4 we specify the instrumental notion of space-bounded enumerability and prove some lemmas about it in connection to low-congesting graphs. Next, in Sect. 3.5 we employ these lemmas to prove that the low-congesting property is testable by small circuits. Hence, by applying the main principle (lemma 1) this property is satisfied for pseudo-random graphs produced by the NW generator as well as for random ones. Moreover, a seed producing a graph with this property may be found in polynomial space. Finally, in Sect. 3.6 we formulate our version of Muchnik's theorem and prove it using the graph obtained on the previous step. I.e., we describe the procedure of choosing a fingerprint in this graph and then calculate all complexities and space requirements and assure that they do not esceed the respective limitations.

3.3 Low-congesting graphs

In fact, the proof in [10] does not use the extractor property, but employs only its corollary. In this section we accurately define this corollary in a way that allows derandomization.

Fix some bipartite graph with parameters $(n,\ m,\ d)$ and an integer $k < n$. Let there be a system $\mathcal{S}$ of subsets of its left part with the following condition: each $S \in \mathcal{S}$ contains less than 2^k vertices, and the whole system $\mathcal{S}$ contains $2^{O(n)}$ sets (note that there are $\sum_{i=1}^{2^k} C_{2^n}^i > 2^{O(n)}$ different sets of required size, so the last limitation is not trivial). We refer to such systems as to "*relevant*" ones. Having a system $\mathcal{S}$ fixed, let us call the sets in it *relevant* also.

Lemma 2. *Let $\mathcal{S}_k = \{S \subset \{0,1\}^n \mid \exists b\, \exists s\, |b| = n$ and $S = \{x \mid \mathrm{C}^s(x|b) < k\}\}$. Then the system $\mathcal{S}_k$ is relevant.*

Proof. By a standard counting argument, each set $S \in \mathcal{S}_k$ contains less than 2^k elements. If b is fixed then the described sets are expanding while s is rising. Since the largest set is also smaller than 2^k, there are less than 2^k different sets for fixed b. Since there are 2^n different b, the total size of $\mathcal{S}_k$ is bounded by $2^n 2^k = 2^{O(n)}$.

Considering a modification of the upper system $\mathcal{S}_{k,\bar{s}} = \{S \subset \{0,1\}^n \mid \exists b\, \exists s < \bar{s}\, S = \{x \mid \mathrm{C}^s(x|b) < k\}\}$, one may note that it is relevant either.

Now fix some relevant system $\mathcal{S}$ and take an arbitrary set S in it. Call the α-*clot* for S the set of right-part vertices that have more than $\alpha DK/M$ neighbours in S (that is, at least α times more than on average). Call a vertex $x \in S$ α-*congested* (for S) if all its neighbours are in the α-clot for S. Say that G is (α, β)-*low-congesting* if there are less than βK α-congested vertices in any relevant S.

Following [2], we prove the next lemma:

Lemma 3. *Let G be an extractor graph for parameters n, m, d, k, ε. Then for any $\alpha > 1$ the graph G is $(\alpha, \frac{\alpha}{\alpha-1}\varepsilon)$-low-congesting.*

Proof. In fact in an extractor graph there are less than $\frac{\alpha}{\alpha-1}K$ α-congested vertices in any set S of size K, not only in relevant ones. We may treat S as an arbitrary set of size exactly K: since a congested vertex in a subset is also a congested vertex in the set, an upper bound for the number of congested vertices in the set holds also for a subset.

Let Y be the α-clot for S, and $|Y| = \delta M$. Then the fraction of vertices in Y equals δ and the fraction of edges directing from S to Y is greater than $\alpha\delta$ (by the definition of clot). A standard counting argument implies only $\delta < \frac{1}{\alpha}$, but by the extractor property these fractions differ by at most ε, so $(\alpha-1)\delta < \varepsilon$, i.e. $\delta < \frac{1}{\alpha-1}\varepsilon$. Next, let $T \subset S$ be the set of α-congested vertices in S, and $|T| = \beta K$. All edges from T direct to vertices in Y, so at least $D|T| = \beta DK$ edges direct from S to Y. In other words, the fraction of edges from S to Y is at least β. By the extractor property it must differ from the fraction of vertices in Y (that equals δ) by at most ε. So, $\beta < \delta + \varepsilon < \frac{1}{\alpha-1}\varepsilon + \varepsilon = \frac{\alpha}{\alpha-1}\varepsilon$. Putting all together, the number of α-congested vertices in any relevant S is less than $\frac{\alpha}{\alpha-1}\varepsilon K$, so the graph is $(\alpha, \frac{\alpha}{\alpha-1}\varepsilon)$-low-congesting, q.e.d.

3.4 Space-bounded enumerability

Say that a system $\mathcal{S}$ is *enumerable* in space q if there exists an algorithm with two inputs i and j working in space q that either outputs the jth element of the ith set from $\mathcal{S}$ or indicates that at least one of the inputs falls out of range. Note that for a polynomial space bound enumeration is equivalent to recognition: if one may enumerate a set then she may recognize whether a given element belongs to it by sequentially comparing it to all enumerated strings, and if one may recognize membership to a set then she may enumerate it by trying all possible strings and including only those accepted by the recognition algorithm. Only small auxiliary space is needed to perform these modifications.

Lemma 4. *The system $\mathcal{S}_{k,\bar{s}} = \{S \subset \{0,1\}^n \mid \exists b \, \exists s < \bar{s} \; |b| = n \; and \; S = \{x \mid \mathrm{C}^s(x|b) < k\}\}$ is enumerable in space $O(\bar{s}) + \mathrm{poly}(n)$.*

Proof. Assume firstly that a set S is given (by specifying b and $s < \bar{s}$) and show how to enumerate it. Look through all programs shorter than k and launch them on b limiting the space to s and counting the number of steps. If this number exceeds c^s (for some constant c depending only on the computational model)

then the current program has looped. In this case or if the program exceeds the space limit we terminate it and go to the next one. Otherwise, if the program produces an output, then we check whether it has not been produced by any previous program. This check is performed as follows: repeat the same procedure for all previous programs and compare their results with that of the current one. If no result coincides then include the current result in enumeration, otherwise skip it and turn to the next program. Emulation of a program requires $O(s)$ space and no two emulations should run in parallel. All intermediate results are polynomial in n, so a total space limit $O(\bar{s}) + \text{poly}(n)$ holds.

Specifying a set S by b and s is not reasonable because a lot of values of s lead to the same sets. (And if $\bar{s} = 2^{\omega(n)}$ then the number of possible indexes (b, s) exceeds the limit $2^{O(n)}$ on the size of $\mathcal{S}$.) Instead, call a limit s pivotal if $\{x \mid \mathrm{C}^s(x|b) < k\} \neq \{x \mid \mathrm{C}^{s-1}(x|b) < k\}$ and consider only pivotal limits in the definition of $\mathcal{S}_{k,\bar{s}}$. Clearly, the latter modification of definition does not affect the system itself. The advantage is that there are less than 2^k pivotal limits, and the ith pivotal limit s_i may be found algorithmically in space $O(s_i) + \text{poly}(n)$. It is sufficient to construct an algorithm recognizing whether a given limit is pivotal, and the latter is done by a procedure similar to one described in the first paragraph: try all possible programs in space s, and in case they produce an output, check whether it is produced by any program in space $s - 1$. If at least one result is new then s is pivotal, otherwise it is not. Putting all together, a set S in $\mathcal{S}_{k,\bar{s}}$ is defined by a word b and the ordinal number i of a pivotal space limit s_i. Knowing these parameters, one may enumerate it in space $O(\bar{s}) + \text{poly}(n)$, as claimed.

The next lemma indicates that if a system $\mathcal{S}$ is enumerable in small space, then the same holds for the system of congested subsets of its members. We call a bipartite graph *computable* in space q if there exists an algorithm working in space q that receives an index of a left-part vertex and an index of an incident edge and outputs the right-part vertex this edge directs to.

Lemma 5. *Let $\mathcal{S}$ be a system of relevant sets enumerable in space s, G be a bipartite graph computable in space q and $\alpha > 1$ be a (rational) number. Then the system $\mathrm{Con}_\alpha \mathcal{S} = \{T \mid \exists S \in \mathcal{S}$ for which T is the set of α-congested vertices$\}$ is computable in space $O(\max\{s, q\}) + \text{poly}(n)$. Moreover, the iteration $(\mathrm{Con}_\alpha)^r \mathcal{S}$ is also computable in space $O(\max\{s, q\}) + \text{poly}(n)$ with constants in O- and poly- notations not depending on r, but possibly depending on α.*

Proof. Since for space complexity enumeration and recognition are equivalent, it is sufficient to recognize that a vertex x is α-congested. The recognizing algorithm works as follows: having a set S and a vertex $x \in S$ fixed, search through all neighbours of x (this requires space $O(q) + \text{poly}(n)$) and for each neighbour check whether it lies in the α-clot for S. If all neighbours do then x is congested, otherwise it is not. Having a neighbour y fixed, the check is performed in the following way: enumerate all members of S (using $O(s) + \text{poly}(n)$ space), for each member search through all its neighbours (using $O(q) + \text{poly}(n)$ space) and count the number of these neighbours coinciding with y. Finally, compare this

number with the threshold $\alpha DK/M$. Note that no two computations requiring space s or q run in parallel and all intermediate results need only polynomial space, so the total space requirement is $O(\max\{s,q\})+\mathrm{poly}(n)$, as claimed. Note also that O- notation is used only due to the possibility of computational model change, not because of the necessity of looping control. If a computational model is fixed then the required space is just $\max\{s,q\}+\mathrm{poly}(n)$.

The last observation is crucial for the "moreover" part of lemma. Indeed, by a simple counting argument the fraction of α-congested vertices in S is at most $1/\alpha$. That is why after at most $\log_\alpha 2^n = O(n)$ iterations the set of congested variables becomes empty. Each iteration adds $\mathrm{poly}(n)$ to the space requirement, so the overall demand is still $\max\{s,q\}+\mathrm{poly}(n)$ (with greater polynomial), as claimed.

3.5 Derandomization

In this section we show that, firstly, the low-congesting property may be (approximately) recognized by $2^{O(n)}$-sized constant-depth circuits, secondly, that there are low-congesting graphs in output of Nisan-Wigderson pseudo-random generator and, thirdly, one can recognize in polynomial space whether the NW-generator produces a low-congesting graph on a given seed. Being put together, last two lemmas provide a seed on which the NW-generator produces a low-congested graph.

Let us encode a bipartite graph with parameters (n, m, d) by a list of edges. The length of this list is $2^n 2^d m$: for each of 2^n left-part vertices we specify 2^d neighbours, each being m-bit long.

Lemma 6. *Let $\mathcal{G}$ be the set of all bipartite graphs with parameters (n, m, d) encoded as described above. Let k be an integer such that $1 < k < n$, and $\varepsilon > 0$. Then there is a circuit C of size $2^{O(n)}$ and constant depth, defined on $\mathcal{G}$, such that:*

- *If G is a (k, ε)-extractor then $C(G)=1$;*
- *If $C(G) = 1$ then G is a $(2.01, 2.01\varepsilon)$-low-congesting graph for $\mathcal{S}_k$ from lemma 2.*

Proof. We build a non-uniform circuit, so we may think that $\mathcal{S}_k$ is given. We construct a single circuit approximately counting the number of congested vertices in a given set, then replicate it for each relevant set and take conjunction. Since there are less than 2^{n+k} relevant sets, this operation keeps the size of circuit beeing $2^{O(n)}$. We proceed by constructing a circuit for a given set S.

The size of the input is $|S| \cdot 2^d m$. We think of it as of being divided into $|S|$ blocks of 2^d segments of length m. We index all blocks with elements $x \in S$ and index all segments of the block x by vertices y incident to x. It is easy to see that there is a constant-depth circuit that compares two segments (that is, has $2m$ inputs and outputs 1 if and only if the first half of inputs coincides with the second half). On the first stage we apply this circuit to every pair of segments, obtaining a long 0-1-sequence. On the second stage we employ a

counting circuit with $|S|D - 1$ arguments that is guaranteed to output 1 if more than $2.01DK/M$ of its arguments are 1's and to output 0 if the number of 1's is less than $2DK/M$. By theorem 3 there exists such circuit of polynomial (in the number of arguments) size and constant depth. For all segments y a copy of this circuit is applied to the results of comparing y to all other segments. If y lies in 2.01-clot then the respective copy outputs 1, and if it outputs 1, then y lies in 2-clot. On the third stage we take a conjuction of all second-stage results for segments lying in the same block x. If this conjunction equals 1 then all images of x lie in 2-clot, that is, x is 2-congested. Conversely, if x is 2.01-congested then the conjunction equals 1. Finally, we utilize another counting circuit with $|S|$ inputs that outputs 0 if *more* than $2.01\varepsilon K$ of its inputs are 1's and outputs 1 if *less* than $2\varepsilon K$ of its inputs are 1's. This circuit is applied to all outputs of the third stage. If the final result is 1 then less than $2.01\varepsilon K$ elements of S are 2.01-congested; and if less than $2\varepsilon K$ elements of S are 2-congested then the final result is 1.

The last claim holds for any relevant S. On the very last stage we take a conjunction of results for all S. If this conjunction is 1 then less than $2.01\varepsilon K$ elements in each S are 2.01-congested, and if G is a (k, ε)-extractor then by lemma 3 less than $2\varepsilon K$ elements in each S are 2-congested and hence this conjunction equals 1.

Since there are at most $2^{O(n)}$ gates on every stage and each stage has constant depth, the overall circuit has also $2^{O(n)}$ gates and constant depth, as claimed.

Lemma 7. *Let n, $m = k$, $d = O(\log n)$ and ε be such parameters that a random bipartite graph with parameters (n, d, m) is a (k, ε)-extractor with constant probability $p > 0$. Let q be the depth of the circuit from the previous lemma. Let $NW\colon \{0.1\}^l \to \{0, 1\}^N$, where $l = O(\log^{2q+6})$ and $N = 2^n 2^d m$, be the Nisan-Wigderson generator from corollary 1. Then $\mathrm{Prob}_u\{C(NW(u)) = 1\} > \frac{p}{2}$ for sufficiently large n, where C is the circuit from previous lemma.*

Proof. This is a straightforward application of lemma 1. By the previous lemma if a graph G is an extractor then $C(G) = 1$, so $\mathrm{Prob}_G\{C(G) = 1\} \geq p$. Since C is a constant-depth circuit, the property $C(G) = 1$ is tautologically testable by a constant-depth circuit. By lemma 1 $\mathrm{Prob}_u\{C(NW(u)) = 1\} > \frac{p}{2}$, q.e.d.

Consider the following problem $\mathcal{R}$: find a string $u \in \{0, 1\}^l$ such that the graph $NW(u)$ is $(2.01, 2.01\varepsilon K)$-congested with respect to $\mathcal{S}_{k, s}$.

Lemma 8. *The problem $\mathcal{R}$ is solvable in space $O(s) + \mathrm{poly}(n)$.*

Proof. The existence of a solution follows from the previous lemma. Since we care only about the space limit, the search problem may be replaced by the corresponding recognition problem. The space bound for the latter one arises from corollary 1, lemma 4 and lemma 5. Indeed, by corollary 1 the graph $NW(u)$ is computable in polynomial space, and by lemma 4 the system $\mathcal{S}_{k, s}$ is enumerable in space $O(s) + \mathrm{poly}(n)$. Hence by lemma 5 the system $\mathrm{Con}_{2.01}\, \mathcal{S}_{k, s}$ is also enumerable in space $O(s) + \mathrm{poly}(n)$, therefore one may easily check whether each set in $\mathrm{Con}_{2.01}\, \mathcal{S}_{k, s}$ contains less than $2.01\varepsilon K$ elements, thus solving the recognition analogue of $\mathcal{R}$. Only polynomial extra space is added on the last step.

3.6 Proof of the theorem

Now we proceed by formulating and proving our version of Muchnik's theorem.

Theorem 6. *Let a and b be binary strings of length less than n, and s and k are numbers such that $\mathrm{C}^s(a|b) < k$. Then there exists a binary string p, such that*

- $\mathrm{C}^{O(s)+\mathrm{poly}(n)}(a|p,b) = O(\log\log s + \log n)$*;*
- $\mathrm{C}^s(p) \leq k + O(\log n)$*;*
- $\mathrm{C}^{O(s)+\mathrm{poly}(n)}(p|a) = O(\log\log s + \log n)$*,*

where all constants in O- and poly*-notations depend only on the choice of the optimal description method.*

Proof. Basically the proof proceeds as the respective proof of theorem 5 in [10]. Here we replace an explicitly constructed extractor by a pseudorandom graph described in Sect. 3.5.

Let a small constant ε and $d = O(\log n)$ be such that a random bipartite graph with parameters $(n, m = k, d)$ is a (k, ε)-extractor with probability greater than some positive constant μ. Let l be a parameter and NW be a function from lemma 7. By lemmas 7 and 6 the output of NW is a $(2.01, 2.01\varepsilon K)$-low-congesting graph for $\mathcal{S}_k$ with probability at least $\mu/2$. Applying the program from lemma 8, one may find in $O(s)+\mathrm{poly}(n)$ space a seed u for which $NW(u)$ is a $(2.01, 2.01\varepsilon K)$-low-congesting graph for $\mathcal{S}_{k,s}$. This u has low complexity: to perform this search one needs to know parameters n, k and $l = \mathrm{poly}(n)$, that is, $O(\log n)$ bits, and the space bound s, that requires $\log s$ bits. The last number may be reduced to $\log\log s$, because the space bound s may be replaced by the least degree of 2 exceeding s keeping the needed space to be $O(s)+\mathrm{poly}(n)$. For what follows, fix this seed u and the graph $G = NW(u)$.

By definition of the low-congesting property the number of 2.01-congested vertices in the set $S = \{x \mid \mathrm{C}^s(x|b) < k\}$ does not exceed $2.01\varepsilon K$. Clearly, a belongs to S. Firstly suppose that it is not 2.01-congested. Then it has a neighbour outside of 2.01-clot for S. This neighbour may be taken as p. Indeed, the length of p equals $m = k$, hence its complexity is also less than $k + O(1)$. To specify p knowing a, one needs to construct the graph $NW(u)$, for which only $O(\log n) + O(\log\log s)$ bits are necessary, and to know the number of p among a's neighbours, which is at most $d = O(\log n)$. Finding a given b and p proceeds as follows: construct G, enumerate S and choose the specified preimage of p in S. Then a is determined by the information needed to construct G ($O(\log n + \log\log s)$ bits), information needed to enumerate S (that is, k, $\log\log s$ and b) and the number of a among preimages of p (since p is not in 2.01-clot, there are not more than $2.01DK/M = 2.01D$ preimages, so $O(d) = O(\log n)$ bits are required). Summarizing, we get $O(\log n) + O(\log\log s)$ bits and $O(s)+\mathrm{poly}(n)$ space needed.

Now we turn to the case when a *is* 2.01-congested. By lemma 5 the set $\mathrm{Con}_{2.01} S$ of 2.01-congested vertices is enumerable in $O(s)+\mathrm{poly}(n)$ space. Take

parameters $n_1 = n$, $m_1 = k_1 = \log K_1 = \log(2.01\varepsilon K)$, $d_1 = O(\log n)$ and $\varepsilon_1 = \varepsilon$ such that an extractor with these parameters exists with probability greater than μ. Lemmas 6, 7 and 8 are applicable to the new situation, so we may find a new u_1 such that the graph $G_1 = NW(u_1)$ is $(2.01, 2.01\varepsilon K_1)$-low-congesting for $\mathrm{Con}_{2.01}\, \mathcal{S}_{k,\,s}$ with probability at least $\mu/2$. By assumption, a belongs to the set $\mathrm{Con}_{2.01}\, S$. If it is not 2.01-congested in it then we choose p analogously to the initial situation. Otherwise we reduce the parameters again and take a low-congesting graph for $\mathrm{Con}^2_{2.01}\, \mathcal{S}_{k,\,s}$, and so on. By the "moreover" part of lemma 5, this reduction may be performed iteratively for arbitrary number of times keeping the space limit to be $O(s) + \mathrm{poly}(n)$.

The total number of iterations is less than $\log_{1/2.01\varepsilon} k = O(\log n)$. Finally p is defined as a neighbour of a not lying in 2.01-clot in graph G_i. To find p knowing a one needs to know i and the same information as on the upper level. To find a knowing p and b also only specifying i is needed besides what has been specified on the upper level. So, all complexities and space limits are as claimed and the theorem is proven.

Acknowledgments

References

1. M. Ajtai. Approximate counting with uniform constant-depth circuits. In Advances in computational complexity theory, pages 1–20. American Mathematical Society, 1993.
2. H. Buhrman, L. Fortnow, S. Laplante, Resource bounded Kolmogorov complexity revisited, *SIAM Journal on Computing*, **31**(3):887–905, 2002.
3. H. Buhrman, T. Lee, D. van Melkebeek. Language compression and pseudorandom generators. In *Proc. of the 15th IEEE Conference on Computational Complexity*, IEEE, 2004.
4. J. Hitchcock, A. Pavan, N. Vinodchandran. Kolmogorov complexity in randomness extraction. Electronic Colloquium on Computational Complexity (ECCC) (09–071) (2009)
5. M. Li, P. Vitanyi, An Introduction to Kolmogorov Complexity and Its Applications, 2 ed., 1997. Springer-Verlag.
6. An.A. Muchnik, On basic structures of the descriptive theory of algorithms. *Soviet Math. Dokl.*, **32**, 671–674 (1985)
7. An. Muchnik, Conditional complexity and codes, *Theoretical Computer Science*, v. 271, no. 1–2, p. 97–109 (2002).
8. D. Musatov, Extractors and an effective variant of Muchnik's theorem. Diplom (Master thesis). *Faculty of Mechanics and Mathematics, MSU*, 2006. `http://arxiv.org/abs/0811.3958` (in Russian)

9. D. Musatov, Improving the space-bounded version of Muchnik's conditional complexity theorem via "naive" derandomization, 2010, `http://arxiv.org/abs/1009.5108` (Online version of this paper)

10. D. Musatov, A. Romashchenko, A. Shen. Variations on Muchnik's conditional complexity theorem. In: A. Frid, A. Morozov, A. Rybalchenko, K. Wagner (eds.) CSR. Lecture Notes in Computer Science, vol. 5675, pp. 250-262 (2009)

11. D. Musatov, A. Romashchenko, A. Shen. Variations on Muchnik's conditional complexity theorem, to be published in TOCS

12. N. Nisan and A. Wigderson. Hardness vs. Randomness. *Journal of Computer and System Sciences.* 49, 1994, pp. 149–167.

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

14. O. Reingold, R. Shaltiel, A. Wigderson, Extracting randomness via repeated condensing, *SIAM Journal on Computing* **35**(5):1185–1209, 2006.

15. D. Slepian and J. K. Wolf. Noiseless coding of correlated information sources. *IEEE Transactions on information Theory*, vol. 19, 1973, pp. 471–480.

16. L. Trevisan. Construction of extractors using pseudo-random generators. In *Proc. 45th Annual Symposium on Foundations of Computer Science*, pp. 264–275.

17. A. Zvonkin, L. Levin. 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 (1970)