Short lists with short programs from programs of functions and strings

Nikolay Vereshchagin*

National Research University Higher School of Economics

Abstract

Let $\{\varphi_p\}$ be an optimal Gödel numbering of the family of computable functions (in Schnorr's sense), where p ranges over binary strings. Assume that a list of strings $L(p)$ is computable from p and for all p contains a φ-program for φ_p whose length is at most ε bits larger than the length of the shortest φ-programs for φ_p. We show that for infinitely many p the list $L(p)$ must have $2^{|p|-\varepsilon-O(1)}$ strings. Here ε is an arbitrary function of p.

1 Results

A *numbering of a family of computable functions of m variables* is a computable partial function $\varphi : (\{0,1\}^*)^{m+1} \to \{0,1\}^*$. We call p a *φ-index* or a *φ-program* for the function $\langle x_1 \dots, x_m\rangle \mapsto \varphi(p, x_1, \dots, x_m)$, which is denoted as φ_p. A numbering φ is *universal* if for all computable partial functions f from $(\{0,1\}^*)^m$ to $\{0,1\}^*$ there is p with $\varphi_p = f$.

By $C_\varphi(f)$ we denote the minimal length of a φ-program for f (*Kolmogorov complexity of f with respect to φ*). A numbering φ has *the Kolmogorov property*, if for every other numbering ψ there is a constant c such that $C_\varphi(f) \leqslant C_\psi(f) + c$ for all partial computable functions f.

A numbering φ is called a *Gödel numbering* if for every other numbering ψ there is a total computable function t (called a *translator* from ψ to φ) such that $\psi_p = \varphi_{t(p)}$ for all p. Let $|p|$ denote the length of p. A Gödel numbering φ is called an *optimal Gödel numbering* if for all numberings ψ there is a

*The work was done while visiting IMS (University of Singapore), the program "Algorithmic Randomness", 2–30 June 2014. The work was part supported by the Russian Academic Excellence Project '5-100' and by the RFBR grant 16-01-00362.

translator t from ψ to φ that has an additional property: $|t(p)| \leqslant |p| + O(1)$ (the translator is *linearly bounded*).[1] Every optimal Gödel numbering has the Kolmogorov property but not vice versa.

Example 1. Here is an example of an optimal Gödel numbering φ of the family of computable functions of m variables. Let Φ denote a universal numbering of the family of computable functions of $m+1$ variables. Let $p \mapsto \hat{p}$ denote a computable prefix encoding, for instance, $\hat{p} = 0^{|p|}1p$. Then $\varphi(\hat{p}q, x_1, \ldots, x_m) = \Phi(p, q, x_1, \ldots, x_m)$ is an optimal Gödel numbering of the family of computable functions of m variables. Indeed, the mapping $t(q) = \hat{p}q$ is a linearly bounded translator from the numbering Φ_p to φ.

The above definitions make sense also for $m = 0$. In this case φ_p is understood as $\varphi(p)$ if defined, and as a special symbol $\perp$ if not defined. Optimal Gödel numberings for $m = 0$ were called *standard machines* in [1] and we will use the same terminology. Kolmogorov complexity $C_U(x)$ of a string x with respect to a standard machine U is the usual Kolmogorov complexity (the minimal length of a U-program for x).

1.1 Short lists from programs of strings: the uniform approach

The paper [1] shows that for every standard machine U, given a string x we can find a short list of strings with a short program for x: the size (=cardinality) of the list is $O(|x|^2)$ and it contains a U-program for x of length at most $C_U(x) + O(1)$. We assume now, and later, that the list is given by a canonical index, rather than a c.e. index.

Is there a total algorithm that computes a short list with a short program for x from any U-program for x? "Total" means that on input q, the algorithm will output something even if it is not the case that $U(q) = x$, and more specifically, even if $U(q)$ does not halt. This question was asked recently by Alexander Shen [5]. Notice that there is no total algorithm that maps any program for x to x (otherwise the positive answer to the question would immediately follow from the cited result of [1]). We will show that the answer to this question is negative. More specifically, for every standard machine U, for every integer valued function ε of p (its domain is the set of all strings and the function is not necessarily computable) and for every total computable function $p \mapsto L(p)$ that maps each string p to a list of strings

[1] The term "optimal Gödel numbering" was introduced by Schnorr [4]. Teutsch and Zimand [2] call optimal Gödel numberings *Kolmogorov numberings*. However Kolmogorov has neither introduced nor studied them.

$L(p)$ the following holds true. If for all p the list $L(p)$ contains a program for $x = U(p)$ of length at most $C_U(x) + \varepsilon$, then there are infinitely many pairs (a string x, its U-program p) such that the size of $L(p)$ is exponential in both $|x| - \varepsilon$ and $|p| - \varepsilon$.

Now we are going to state our result in a more precise form. Let $C_{U,L}(x)$ denote the minimal length of a U-program $p \in L$ for x. The smaller $C_{U,L}(x)$ is, the shorter the programs for x contained by L. Obviously, $C_{U,L}(x) \geqslant C_U(x)$ for all x and L.

Consider two examples of computable lists.

1) Let $L(p) = \{p\} \cup \{0,1\}^{<|p|}$. For all p we have $C_{U,L(p)}(U(p)) = C_U(U(p))$ so this list is perfect with respect to $C_{U,L}$. However if x is a random string and p its shortest U-program then the size of $L(p)$ is exponential in both $|p|$ and $|x|$.

2) Let $L(p) = \{p\}$. This list is perfect size-wise. However $C_{U,L}(U(p)) \gg C_U(U(p))$ provided $U(p)$ has little complexity, that is, $C_U(U(p)) \ll |p|$.

In both examples there are programs p such that either the size of the list is very big, or $C_{U,L(p)}(U(p))$ is much larger than $C_U(U(p))$. Our main result shows that such a trade off between the size of L and $C_{U,L}(U(p))$ holds for all computable lists for infinitely many programs p. More specifically, for every computable function $L(p)$ there are infinitely many pairs p, x with $U(p) = x$ such that

$$\log_2(\#L(p) + 2) + C_{U,L(p)}(x) \geqslant |p| + |x| + O(1) \geqslant |p| + C_U(x) + O(1). \quad (1)$$

The second inequality here is implied by the fact that the complexity is not larger than the length. The first inequality follows from the next theorem.

Theorem 1. *Let U be a standard machine and L a total computable function mapping strings to finite sets of strings. Then for some c and for all k the following holds. There is a string x and its U-program p of length between k and $k + c$ such that $\#L(p) \geqslant 2^{|x|} - 2$ and $C_{U,L(p)}(x) \geqslant |p| - c$.*

Notice that the last two inequalities imply inequality (1). This theorem implies a negative answer to Shen's question. Indeed, assume that for all p such that $U(p)$ is defined the list $L(p)$ has an $\varepsilon(p)$-shortest program for $U(p)$, that is,

$$C_{U,L(p)}(U(p)) \leqslant C_U(U(p)) + \varepsilon(p).$$

Subtracting this inequality from inequality (1) we obtain

$$\log_2(\#L(p) + 2) \geqslant |p| - \varepsilon(p) - O(1)$$

for all such p. Thus we obtain the following corollary from the theorem.

Corollary 2 (A negative answer to Shen's question)**.** *Let U, L be as in the theorem. Assume that for all p such that $U(p)$ is defined the list $L(p)$ has an $\varepsilon(p)$-shortest program for $U(p)$. Then for infinitely many p the size of $L(p)$ is at least*

$$2^{|p|-\varepsilon(p)-O(1)} - 2.$$

Let us stress that L is assumed to be a total function. If we allowed L to be defined only on those strings p for which $U(p)$ halts, then there would be a computable list $L(p)$ of quadratic size (in the length of $x = U(p)$) with a program for x of length $\leqslant C_U(x) + O(1)$, which follows from the result of [1] cited above.

1.2 Short lists from programs of functions: the uniform approach

Theorem 1 easily translates to functions. Let $\varphi : (\{0,1\}^*)^{m+1} \to \{0,1\}^*$ be an optimal Gödel numbering of functions of $m \geqslant 1$ variables. Let Singl_x denote the function defined only on the m-tuple $\langle x, \ldots, x\rangle$ with value x. Let $C_{\varphi,L}(f)$ denote the minimal length of a φ-program $p \in L$ for f.

Theorem 3. *Let φ be an optimal Gödel function of $m > 0$ variables and L a total computable function mapping strings to finite sets of strings. Then, for some c and for all k the following holds. There is a string x and a φ-program p of length between k and $k + c$ for the function* Singl_x*, such that $\#L(p) \geqslant 2^{|x|} - 2$ and $C_{\varphi,L(p)}(\mathrm{Singl}_x) \geqslant |p| - c$.*

Remark 1. Theorem 3 holds for numberings of enumerable sets with the singleton set $\{x\}$ in place of the function Singl_x. The proof is exactly the same.

For the string p from Theorem 3 we have

$$\log(\#L(p) + 2) + C_{U,L(p)}(\varphi_p) \geqslant C_U(\varphi_p) + |p| - O(1). \tag{2}$$

Indeed, $\log(\#L(p)+2) \geqslant |x| \geqslant C_U(\varphi_p) - O(1)$ and $C_{U,L(p)}(\varphi_p) \geqslant |p| - O(1)$. Summing these inequalities we get (2).

Theorem 3 answers a question asked recently by Teutsch and Zimand. For a numbering φ of computable functions of one variable, Teutsch and Zimand [2] considered the set of minimal programs for φ, where p is called *minimal*, if for all $q < p$ we have $\varphi_q \neq \varphi_p$. Here $<$ denotes the lexicographical ordering of binary strings (more precisely, $p < q$ iff $|p| < |q|$ or $|p| = |q|$ and p is lexicographically less than q). The minimal φ-program for a function φ_q is denoted by $\min_\varphi(q)$. Teutsch and Zimand showed the following.

- If φ is a Gödel numbering and L a computable function L, that, on input p, returns a list $L(p)$ containing $\min_\varphi(p)$, then the size of that list is not bounded by a constant.

- Assume that φ is a numbering φ with the Kolmogorov property. If the computable function L, on input p, returns a list containing $\min_\varphi(p)$, then the size of the list must be $\Omega(|p|^2)$.

- There exists an optimal Gödel numbering φ such that if the computable function, on input p, returns a list containing $\min_\varphi(p)$, then the size of that list must be $\Omega(2^{|p|})$.

In summary, their results show that a computable list that contains the minimal φ-program cannot be too small.

Along the lines of the second result Teutsch and Zimand [2, Question 7] asked the following question: is there an optimal Gödel numbering φ with a computable list $L(p)$ that contains $\min_\varphi(p)$ and has size $O(|p|^2)$?

Theorem 3 implies a negative answer to this question. Indeed, if $\min_\varphi(p) \in L(p)$ for all p then $C_{\varphi,L(p)}(\varphi_p) = C_\varphi(\varphi_p)$ for all p. By (2) for the pair p, x existing by the theorem the size of $L(p)$ must be at least $2^{|p|-O(1)}$. In other words, the third result of Teutsch and Zimand holds for *all* optimal Gödel numberings φ.

1.3 Short lists from programs of strings and functions: non-uniform approaches

So far we are constructing for the given computable function L, strings p such that the list $L(p)$ has large parameters $\#L(p)$ and $C_{U,L(p)}(U(p))$. Let us consider the "short list with short programs" problem from the other end. Are there p's such that *every* short list L, computable from p by a total algorithm, has high parameters $\#L(p)$ and $C_{U,L}(U(p))$? In this form the question is trivial: we can hard-wire the shortest U-program q for $U(p)$ into a total algorithm which will return the list $\{q\}$, which has optimal parameters. The question becomes reasonable if we restrict the complexity, say by $O(\log |p|)$, of the total algorithm producing the list from p.

To make this question precise consider the *total complexity* $CT_\Phi(a\,|\,b)$ defined as the minimal length of a Φ-program of a total function that maps b to a. Here Φ is an optimal Gödel numbering of computable functions of one variable.

Fix a natural number δ (the upper bound for total complexity). Then for each p consider the set

$$S_p^\delta = \{(i,j) \mid \exists L,\ CT(L|p) \leqslant \delta,\ \#L \leqslant 2^i,\ C_{U,L}(x) \leqslant j\},$$

where x stands for $U(p)$. The larger this set is, the more likely parameters are to have lists L with small $CT(L|p)$. If $\delta \geqslant \log|p| + O(1)$ then the list $\{0,1\}^i$ for $i = C_U(U(p))$ and the list $\{p\}$ witness that the set S_p^δ includes the entire grey set P on Fig. 1. The set S_p^δ may resemble the famous Kolmogorov

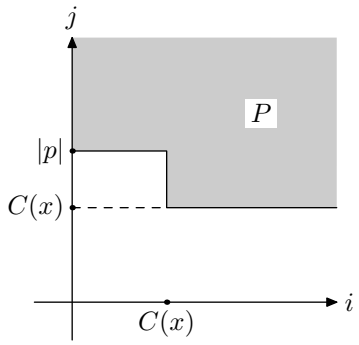

Figure 1: For all p, δ with $\delta \geqslant \log|p| + O(1)$ the set S_p^δ includes the grey set P. Here $x = U(p)$.

structure function, whose graph is defined as the boundary of the set

$$P_x = \{(i,j) \mid \exists L \ni x,\ \#L \leqslant 2^i,\ C(L) \leqslant j\}.$$

However, there is no connection between sets S_p^δ and P_x.

The set S_p^δ may be much larger then the grey set P. This happens when p contains a shortest program for $x = U(p)$. For example, this happens, when p consists of instructions "print x and stop", where x is an incompressible string. In this case the set S_p^δ coincides with the set of all points above the dashed line (we assume that δ is larger than some constant, namely, the length of the (total) program that maps every program of the form "print x' and stop" to x'). Are there infinitely many p's such that the set S_p^δ is close to the grey set P in the picture? In other words, are there infinitely many p's such that for every list L with $CT_\Phi(L|p) \leqslant \delta$, either $\log_2 \#L > C_U(x)$, or $C_{U,L}(x) \geqslant |p|$ (with certain accuracy)? A positive answer is provided by the following

Theorem 4. *Let U be a standard machine. For all n and all $k > n$ there is a string x with $C_U(x) = n + O(1)$ and its U-program p of length $k + O(1)$ such that for all $\delta < k - \log k - O(1)$ and all L with $CT_\Phi(L|p) \leqslant \delta$, either $\#L \geqslant 2^{n-\delta-\log k-O(1)}$, or $C_{U,L}(x) \geqslant |p| - O(1)$.*

Remark 2. Theorem 4 translates immediately to programs of functions. Indeed, let $\varphi : (\{0,1\}^*)^{m+1} \to \{0,1\}^*$ be an optimal Gödel numbering of functions that have m of variables. Let Singl_x denote the function defined only on the m-tuple $\langle x, \ldots, x\rangle$ with value x. Then φ-programs of Singl_x translate to U-programs of x and vice versa (for any standard machine U). Therefore $C_\varphi(\mathrm{Singl}_x)$ and $C_{\varphi,L}(\mathrm{Singl}_x)$ basically coincide with $C_U(x)$ and $C_{U,L}(x)$, respectively.

One can wonder what happens, if we replace in the definition of the set S_p^δ the total conditional complexity by the plain conditional complexity. We can make only some trivial observations in this case. More specifically, consider the set

$$T_p^\delta = \{(i,j) \mid \exists L,\ C(L|p) \leqslant \delta,\ \#L \leqslant 2^i,\ C_{U,L}(x) \leqslant j\},$$

where $x = U(p)$. If $\delta \geqslant \log|p| + O(1)$, then this set coincides with the set of all pairs above the dashed line on Fig. 1. Indeed, let q stand for the shortest program for $U(p)$ that appears the first in an enumeration of all programs for $U(p)$. Then the list $\{q\}$ has complexity at most $\log C(x) + O(1)$ conditional to both $x = U(p)$ and p.

Thus the set T_p^δ may be non-trivial only for $\delta < \log C(x) + O(1)$. We have no idea what it may look like.

For functions one can also consider an analogue of the set T_p^δ. Let

$$R_p^\delta = \{(i,j) \mid \exists L,\ C(L|p) \leqslant \delta,\ \#L \leqslant 2^i,\ C_{\varphi,L}(\varphi_p) \leqslant j\},$$

where $\varphi : (\{0,1\}^*)^{m+1} \to \{0,1\}^*$ denotes an optimal Gödel numbering of functions that have $m \geqslant 1$ variables. Nothing is known about the possible shapes of R_p^δ, even for $\delta > \log C(x)$. Notice that the knowledge of $C_\varphi(\varphi_p)$ does not help to find from p a shortest program for φ_p any more.

2 The proofs

We start with the proofs of Theorems 1 and 3. We first drop the requirement $|p| \geqslant k$. As a reward, the lower bound for the list size will be a little bit stronger: $2^{|x|} - 1$ in place of $2^{|x|} - 2$.

Proof of Theorem 1. Let us first show that the statement of Theorem 1 is invariant: if it holds for some standard machine U then it holds for any other standard machine U'. Indeed, assume that Theorem 1 holds for a standard machine U. To show Theorem 1 for another standard machine U' and a list $L'(p')$, choose a linearly bounded translator t from U' to U and a linearly bounded translator s from U to U'. Let c' be a constant with $|t(p')| \leqslant |p'| + c'$.

Apply Theorem 1 to the machine U and the list $L(p) = t(L'(s(p)))$. By Theorem 1 for all k there is a string x and its U-program p of length at most $k + c' + c$ such that $\#L(p) > 2^{|x|} - 1$ and the list $L(p)$ does not contain any U-program for x of length less than $k + c'$.

Let $p' = s(p)$. By construction,

$$|p'| \leqslant |p| + O(1) \leqslant k + c' + c + O(1).$$

We also have

$$\#L'(p') \geqslant \#t(L'(p')) \geqslant 2^{|x|} - 1.$$

Finally the list $t(L(p'))$ does not contain any U-program of length less than $k + c'$ for x. Hence the list $L(p')$ does not contain any U'-program of length less than k for x.

Thus it suffices to prove Theorem 1 for the standard machine U from Example 1, that is for $U(\hat{p}q) = \Phi(p, q)$, where Φ is a Gödel numbering of the family of computable functions of one variable.

We will let $p = \hat{r}q$ where q, r are certain strings, q is a string of length k and r does not depend on k. The statement of the theorem will follow from the following properties of r, q, where V denotes the function Φ_r (of one variable):

- q is a V-program of a string x such that
- $\#L(\hat{r}q) \geqslant 2^{|x|} - 1$ and
- the list $L(\hat{r}q)$ contains no U-program for x of length less than k.

Notice that the string $p = \hat{r}q$ has all the required properties.

It remains to find such r and q. The computable function V and its Φ-program r will be defined using the Kleene fixed point theorem [3]. By that theorem we may assume that computing V we have access to a Φ-program r for V. We construct an algorithm that enumerates the graph of V.

The algorithm enumerating the graph of V. We maintain for all k a string q_k of length k and a string x_k. At the start let q_k be any string

of length k and let x_k be the empty string. Enumerate all the pairs $\langle q_k, x_k\rangle$ into the graph of V thus letting $V(q_k) = x_k$.

Then we start an enumeration of the graph of U. Each time a new pair appears in that enumeration, we see if the current situation is good or not. We consider the current situation *good for* k if the pair $\langle q_k, x_k\rangle$ has been enumerated into the graph of V, $\#L(\hat{r}q_k) \geqslant 2^{|x_k|} - 1$ and the list $L(\hat{r}q_k)$ has no $\bar{U}$-program for x_k of length less than k, where $\bar{U}$ denotes the sub-function of U consisting of all the pairs enumerated so far.

At the start $\bar{U} = \emptyset$ and thus the situation is good for all k. Each time a new pair appears in the enumeration of the graph of U, we look whether the situation has become bad for some k. Obviously that may happen only if a pair $\langle s, x_k\rangle$ with $|s| < k$ and $s \in L(\hat{r}q_k)$ is enumerated. In that case pick a new string q of length k ("new" means that q has not been used as q_k earlier). Let n be the integer with $2^{n+1} - 1 > \#L(\hat{r}q) \geqslant 2^n - 1$. For all strings x of length at most n consider the set $S(x) = \{s \mid \bar{U}(s) = x,\ |s| < k\}$ of $\bar{U}$-programs for x of length less than k. Pick any string x of length at most n such that $S(x)$ does not intersect the list $L(\hat{r}q)$. Such a string x exists, because $L(\hat{r}q)$ has less than $2^{n+1} - 1$ strings and the number of x's is $2^{n+1} - 1$. Then let $q_k = q$, $x_k = x$ and enumerate the pair $\langle q, x\rangle$ into the graph of V. The situation has become good for k. **End of Algorithm.**

By Kleene's theorem for some r this algorithm enumerates the graph of the function Φ_r. Let us show that for each k, starting from some moment the situation is good for k. Indeed, for any k the situation may become bad less than 2^k times for k, as that may happen only after a new pair of the form $\langle s, x_k\rangle$ with $|s| < k$ has appeared. On the other hand, the number of strings q of length k is 2^k and hence we indeed are able to repair the situation $2^k - 1$ times. □

Proof of Theorem 3. Let $\mathrm{Singl}_\perp$ stand for the nowhere defined function. There is a linearly bounded total computable translator t mapping any U-program for $x \in \{0,1\}^* \cup \{\perp\}$ (for a standard machine U) to a φ-program for the function Singl_x. There is also a linearly bounded total computable translator s mapping any φ-program for Singl_x back to a U-program for x. Given a list L we just apply Theorem 1 to the list $L'(p') = s(L(t(p')))$ and $k + c'$, where c' is a constant with $|s(p)| \leqslant |p| + c'$. □

It remains to prove Theorems 1 and 3 as they are stated, that is, with the requirement $|p| \geqslant k$ and with the lower bound $2^{|x|} - 2$ for the list size. Given any computable list $L(p)$ we add p into the list and apply Theorem 1 in the proven form to the resulting list $L'(p)$. The list $L'(p)$ does have a

U-program for x of length $|p|$ and has no U-program for x of length less than k. This implies that $|p| \geqslant k$. The program p fulfills Theorem 1 in the original form.

The proof of Theorem 3 is exactly the same.

Remark 3. As Jason Teutsch observed, one can prove Theorem 1 without using the fixed point theorem. To this end we modify the construction of V so that V becomes a standard machine. Specifically, we first let $V_{0q} = U_q$ for all strings $0q$ starting with zero, where U is any standard machine. Then we define V_{1q} so that for all k there is a string $1q$ of length $k+1$ such that

- $1q$ is a V-program of a string x such that
- $\#L(1q) \geqslant 2^{|x|} - 1$ and
- the list $L(1q)$ contains no V-program for x of length less than k.

This can be done by the same technique. The function V defined in this way satisfies the theorem. As we already observed, this implies that the theorem holds for all standard machines.

Proof of Theorem 4. The proof is very similar to that of Theorem 1. We construct a computable function V such that for all $k > n$ there are strings q, x with

$$|q| = k, \quad |x| = n, \tag{3}$$

$$V(q) = x, \tag{4}$$

$$C_U(x) \geqslant n - 1, \tag{5}$$

$$\begin{aligned} &\text{for all } \delta < k - \log k - 2 \text{ and all } L \text{ with } CT_\Phi(L|q) = \delta \\ &\text{and } \#L < 2^{n-\delta-\log k-1} \text{ we have } C_{U,L}(x) \geqslant k - 1. \end{aligned} \tag{6}$$

Here Φ is a Gödel numbering of the family of computable functions that have one variable.

The algorithm enumerating the graph of V. This time we maintain for all k a bunch of pairs $\{(q_{kn}, x_{kn}) \mid n = 0, 1, \ldots, k-1\}$. The length of q_{kn} is k and the length of x_{kn} is n. At the start let q_{kn} be the nth string of length k and let x_{kn} be the first string of length n (independent of k). Enumerate all the pairs $\langle q_{kn}, x_{kn}\rangle$ into the graph of V thus letting $V(q_{kn}) = x_{kn}$. By construction, if $x_{kn} \neq x_{k'n'}$ then $q_{kn} \neq q_{k'n'}$. Hence V is a function. Moreover, for all n and all $k > n$ there is a string x of length n (namely x_{kn}) that has a V-program of length k (namely q_{kn}).

Then we start an enumeration of the graph of U and an enumeration of the graph of Φ. We denote by $\bar{U}$ and $\bar{\Phi}$ the sub-functions of U and Φ consisting of all pairs (triples) enumerated so far. For each k we look if the situation is *good for* k. This means that for all $n < k$ the pair $\langle q_{kn}, x_{kn}\rangle$ has been enumerated into the graph of V, $C_{\bar{U}}(x_{kn}) \geqslant n-1$ and for all $\delta < k - \log k - 2$ and all L with $CT_{\bar{\Phi}}(L|q_{kn}) = \delta$ and $\#L < 2^{n-\delta-\log k-1}$ we have $C_{\bar{U},L}(x_{kn}) \geqslant k-1$. Here $CT_{\bar{\Phi}}(L|q)$ means the minimal length of p such that $\bar{\Phi}_p$ is defined *on all* q' *of length* k and $\bar{\Phi}_p(q) = L$.

At the start $\bar{U}$ and $\bar{\Phi}$ are empty and thus the situation is good for all k. Each time a new pair (triple) appears in the enumeration of the graphs of U or Φ, we look whether the situation has become bad for some k. This may may happen only if $C_{\bar{U}}(x_{kn})$ has become less than $n-1$ (for some $n < k$) or a new list L with $CT_{\bar{\Phi}}(L|q_{kn}) < k - \log k - 2$ appeared or for an old list L the value $C_{\bar{U},L}(x_{kn})$ has become less than $k-1$ (for some $n < k$). In all the cases we first change q_{kn} and then we change x_{kn}. The string q_{kn} is replaced by any new string q of length k ("new" means that q has not been used as q_{k*} earlier). The string x_{kn} is replaced by any string x of length n such that $C_{\bar{U}}(x) \geqslant n-1$ and the set $S(x) = \{s \mid \bar{U}(s) = x,\ |s| < k-1\}$ does not intersect the union (over all δ) of all lists L of cardinality less than $2^{n-\delta-\log k-1}$ with $CT_{\bar{\Phi}}(L|q) = \delta$. Note that for every p there is only one list L with $\bar{\Phi}_p(q) = L$ thus the total number of strings in all these lists is less than $\sum_{\delta<k} 2^{\delta} \cdot 2^{n-\delta-\log k-1} = 2^{n-1}$. On the other hand, the number of strings x of length n with $C_{\bar{U}}(x) \geqslant n-1$ is more than 2^{n-1}. Thus, there is such an x.

Then let $q_{kn} = q$, $x_{kn} = x$ and enumerate the pair $\langle q, x\rangle$ into the graph of V. The situation has become good for k. **End of Algorithm.**

We have to show that we are able to choose a new string of length k each time we need one. Any replacement of a string of the form q_{k*} is caused by

- discovering a new p of length less than $k - 2\log k - 2$ such that Φ_p is defined on all strings of length k (this may cause replacement of the whole bunch of q_{kn}'s, for all $n < k$), or
- discovering a new U-program r of length less than $k-1$, which may cause the replacement of q_{kn} only if $U(r) = x_{kn}$ thus for a single n, or
- discovering a new halting U-program of length less than $n-1 < k-1$ for x_{kn}, which again may cause the replacement of q_{kn} only for a single n.

Thus the total number of strings q_{k*} we need is less than

$$\sum_{\delta<k-\log k-2} k2^{\delta}+k+2^{k-1}<2^{k}.$$

To prove the theorem fix $n<k$ and let (q,x) be a pair satisfying the Equations (3), (4), (5) and (6). Let p be the U-program for x obtained from q by translation from V to U. The first two equations translate to the following ones:

$$|p|\leqslant k+O(1),\quad |x|=n, \tag{7}$$

$$U(p)=x. \tag{8}$$

The inequality (5) remains unchanged, as it does not involve p and V. We claim that the last condition (6) translates to the following one:

$$\begin{aligned}&\text{for all } \delta<k-\log k-O(1) \text{ and all } L \text{ with } CT_{\Phi}(L|p)=\delta\\ &\text{and } \#L<2^{n-\delta-\log k-O(1)} \text{ we have } C_{U,L}(x)\geqslant k-1.\end{aligned} \tag{9}$$

Indeed, the only place where Condition (6) has p or V is the equation $CT_{\Phi}(L|p)=\delta$, and for any list L we have $CT_{\Phi}(L|q)\leqslant CT_{\Phi}(L|p)+O(1)$. Indeed, let s be a linearly bounded translator from V to U. Then $\Phi(r,s(q))$ is a computable function hence there is a total computable function t with $\Phi_{t(r)}(q)=\Phi_r(s(q))$. If Φ_r is total then so is $\Phi_{t(r)}$. Hence $CT_{\Phi}(L|q)\leqslant CT_{\Phi}(L|s(q))+O(1)=CT_{\Phi}(L|p)+O(1)$. Now assume that

$$\delta\leqslant k-\log k-c,\qquad CT_{\Phi}(L|p)=\delta,\qquad \#L<2^{n-\delta-\log k-c}.$$

The second assumption implies that $CT_{\Phi}(L|q)=\delta'$ for some $\delta'\leqslant\delta+O(1)$. If c is a large enough constant, we can deduce from the first and the third assumptions that $\delta'<k-\log k-2$ and $\#L<2^{n-\delta'-\log k-1}$. Thus by (6) we have $C_{U,L}(x)\geqslant k-1$.

Let us show that Equations (5), (7), (8) and (9) imply Theorem 4. The complexity of x is at most $|x|+O(1)=n+O(1)$ and at least $n-1$. The length of p is at most $k+O(1)$. To see that it is at least $k-O(1)$, note that the inequality $C_{U,L}(x)\geqslant k-1$ for $L=\{p\}$ implies that $|p|\geqslant k-O(1)$. The Condition (9) implies the last claim from Theorem 4, since $k=|p|+O(1)$, as we have seen. □

Acknowledgments. The author is sincerely grateful to Alexander Shen for asking the question and hearing the preliminary version of the proof of the result. The author is grateful to Jason Teutsch for the idea of how to

omit the use of the fixed point theorem. The author thanks anonymous referees for helpful remarks. The author is also grateful to the hospitality of the IMS of the University of Singapore.

References

[1] Bruno Bauwens, Anton Makhlin, Nikolay Vereshchagin, Marius Zimand. Short lists with short programs in short time. In Proceedings 28th IEEE Conference on Computational Complexity (CCC), Stanford, CA, pages 98–108, June 2013. ECCC report TR13-007.

[2] Jason Teutsch and Marius Zimand. On approximate decidability of minimal programs. 2014. Available from http://arxiv.org/abs/1409.0496 and http://people.cs.uchicago.edu/˜teutsch/papers/teutschpubs.html.

[3] Hartley Rogers, Jr., The Theory of Recursive Functions and Effective Computability, MIT Press, 1987.

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

[5] Alexander Shen. A talk on some open problems in Kolmogorov complexity. The talk was delivered on a meeting during the IMS program "Algorithmic Randomness" (IMS, University of Singapore, 2–30 June 2014) around June 20, 2014.