Enumerating Complexity Revisited

ECCC

Alexander Shekhovtsov

Moscow Institute of Physics and Technology

alex.v.shekhovtsov@gmail.com

Georgii Zakharov

Moscow Institute of Physics and Technology

georgiizakharov0@gmail.com

Abstract

We reduce the best-known upper bound on the length of a program that enumerates a set in terms of the probability of it being enumerated by a random program. We prove a general result that any linear upper bound for finite sets implies the same linear bound for infinite sets.

So far, the best-known upper bound was given by Solovay. He showed that the minimum length of a program enumerating a subset S of natural numbers is bounded by minus three binary logarithms of the probability that a random program will enumerate S. Later, Vereshchagin showed that the constant can be improved from three to two for finite sets. In this work, using the method proposed by Solovay, we demonstrate that any bound for finite sets implies the same for infinite sets, modulo logarithmic factors. Using Vereshchagin's result, we improve the current best-known upper bound from three to two.

Organization

In the first section, we introduce definitions of deterministic and randomized complexity, present the results known prior to our work, and state Theorem 1, which represents the main result of this work. In the second section, we associate cats with the deterministic machine and ants with the randomized machine, and we introduce shadow positions for the ants, which will follow the ants with some delay. We describe how the shadow positions will mimic the real ones. In the third and fourth sections, we demonstrate that if all accumulations in the shadow positions are serviced by cats, then all the necessary infinite sets will be enumerated. The fifth section contains a discussion of the results.

1   Introduction

To formulate the main results of this work, we will introduce the concepts of deterministic and randomized computing machines.

1.1   Deterministic Enumerating Machine

Let's consider a deterministic machine, denoted as D, which takes a binary string as input and enumerates a certain subset of natural numbers. D is not required to halt and may enumerate an infinite set. We define the complexity of a set, denoted as $I_D(S)$, as the minimum input length for D to enumerate S. If there is no such input at all, we consider the complexity of the set to be infinite. As known, there exists a machine D_0 (such a machine is referred to as universal) in such a way that for any other machine D, there is a constant c such that:

$$I_D(S) \leq I_{D_0}(S) + c$$

We can set D equal to any such universal machine D_0 and define $I(S)$ as $I_D(S)$. Thus, $I(S)$ is defined up to a constant factor.

1.2   Probabilistic Enumerating Machine

Let's consider a probabilistic machine, denoted as M, which utilizes random bits (uniformly distributed and independent) and enumerates a certain set (natural numbers occasionally appear as output, with no explicit termination of operation). Such a machine defines a mapping from the Cantor space Ω (all infinite sequences of zeros and ones) to the family of subsets of the natural numbers (a sequence corresponds to the set that is enumerated when it is used as input as a sequence of random bits). The image of uniform measure then becomes a measure on the set of all subsets of the natural numbers, and this is the measure we refer to when discussing the probability (for machine M) of enumerating a certain set S. (This probability is non-zero only if the set S is enumerable, as implied by a variant of the de Leeuw–Moore–Shannon–Shapiro theorem for enumeration problems [2].) We define the complexity $H_M(S)$ as the negative binary logarithm of the probability that M enumerates S: $H_M(S) = -\log_2 Pr(M \ enumerates \ S)$. If this

probability is zero, we consider the complexity to be infinite.

Similar to the deterministic case, there exists a machine M_0 (referred to as optimal) in such a way that for any other M, there is a constant c such that:

$$H_M(S) \leq H_{M_0}(S) + c$$

We can set M equal to any such optimal machine M_0 and define $H(S)$ as $H_M(S)$. Therefore, $H(S)$ is defined up to a constant factor.

1.3 Connection Between Deterministic and Randomized Complexity

Since M can generate an input x for machine D using $|x| + O(\log|x|)$ of its random bits (for instance, one can first generate the length $|x|$ and then x itself), it is true that $H(S) \leq I(S) + O(\log I(S))$ for all $S \subset \mathbb{N}$.

However, obtaining the converse inequality is much more challenging. Solovay proved a linear upper bound.

Theorem (Solovay, [4]). *There exists a constant c such that for any $S \subset \mathbb{N}$, it holds that $I(S) \leq 3 \cdot H(S) + 2\log H(S) + c$.*

In his work, Vereshchagin improved the constant from three to two for finite sets:

Theorem (Vereshchagin, [5]). *There exists a constant c such that for any finite $S \subset \mathbb{N}$, it holds that $I(S) \leq 2 \cdot H(S) + 2\log H(S) + c$.*

Vereshchagin also asked the question whether it is possible to improve the Solovay's bound by reducing the length of an auxiliary string in the Solovay's algorithm. We answer positively to this question. We will show how to extend any upper bound for finite sets to infinite sets. This result is formulated in the following theorem.

Theorem 1. *Let $\alpha \geq 1$ be such that for any finite $S \subset \mathbb{N}$,*

$$I(S) \leq \alpha \cdot H(S) + O(\log H(S))$$

Then the same bound holds for any infinite S (possibly with a different O-big).

Applying this theorem to Vereshchagin's theorem for finite sets, we improve the constant in Solovay's bound from three to two (ignoring logarithmic factors).

To prove this theorem, we will introduce convenient terms. We will represent the behavior of deterministic and randomized machines as the movement of cats and ants on a directed graph.

2 Graph, Cats, Ants, Shadow Positions

2.1 Graph of Subsets of Natural Numbers

Let G be a directed graph, with its vertices representing finite sets of natural numbers, and edges leading from a set to all of its proper supersets.

2.2 Cats

We enumerate the inputs for D in increasing order of their length (for example: ε, 0, 1, 00, 01, 10, 11, 000, ...). We associate a cat with each input for D. Each cat simulates the actions of D on the corresponding input. At any given time, a cat is positioned in the vertex corresponding to what D has enumerated up to that point. At time zero, all cats are in the $\varnothing$ vertex. As machine D outputs new numbers, the cats move along the edges of the graph to new positions. The cats are numbered in the same way as the inputs they correspond to. Hence, $I(S) \leq k \Leftrightarrow$ at least one of the first $2^{k+1} - 1$ cats enumerates the set S.

2.3 Ants

We model the behavior of machine M on different inputs using the metaphor of «ants». While machine M does not request random bits and only enumerates numbers, we envision the movement of a unit-sized ant on the graph G. An ant begins from an empty set and moves to corresponding sets as numbers appear in the output (the ones already enumerated).

When the machine reaches the point of requesting a random bit, the ant splits into two «children» — each of half the size. One follows the behavior of the machine with a random bit 0, and the other follows the behavior with a random bit 1, potentially moving differently. At some point, the ant may need the next random bit, at which point it will split again into two, and so on. At any moment in this modeling process, the ants are finite in number, and each corresponds to a binary word (representing the random bits the ant already knows). These cones form a partition of the Cantor space. Requesting a random bit corresponds to dividing one of the cones in half.

Using this metaphor, we distinguish between an «ant» itself (a node in the infinite tree) and its «position» in the graph. A node indicates which random bits have been used in machine M's computation, while a position represents the set of numbers that have already appeared in machine M's output. There are two types of changes that occur during the modeling process:

- The machine may request a random bit (in some computation path). Then, one ant (corresponding to the bits of x that are already requested) splits into two (corresponding to

nodes $x0$ and $x1$), inheriting the current position.

- The machine (in some computation path) outputs a new number m. The ant, corresponding to the bits already requested, updates its position (moving to a graph vertex obtained from the previous position by adding m).

In these terms, the mapping from the Cantor space to $\mathcal{P}(\mathbb{N})$ can be described as follows: for each point w in the Cantor space, there is a sequence of ants, each being the descendant of the previous one and continuing its path on the graph. The combination of these paths enumerates a set of natural numbers, which represents the image of the point w. (It is possible that only a finite number of bits from input w will be used. In this case, the last ant will not split further, and its path on the graph may be either finite or infinite.)

2.4 Shadow Positions

In the description of the construction, in addition to the positions of the ants on the graph, we use their *shadow positions*, which follow their actual positions with some delay. Specifically, when an ant's position changes, its shadow position remains unchanged, except for specific explicitly highlighted *shifts*. At the moment of such a shift, the shadow position of the ant moves to its real position. When an ant splits, its children inherit not only its position but also its shadow position. Then, the children may diverge, and after shifts, their shadow positions may also diverge.

From this description, it's evident that the shadow position of an ant is one of its previous (real) positions, making it a subset of those positions. Each ant can participate in several shifts (or none at all). For convenience, when we later discuss the number of shifts in which a given ant participated, we mean the total number of shifts for both the ant and its ancestors. (This number will be important to us, particularly whether it is finite or infinite.)

2.5 Shifts

The evolution and positions of ants are determined by the simulation of machine M and are not dependent on us. To describe the entire process, we need to explain when and to which ants shifts are applied. The shift will depend on a parameter ε and a function $l(\varepsilon)$ based on it. We will define the function $l(\varepsilon)$ and ε later.

The conditions for a shift arise when there is a finite set X for which there are more than ε ants whose positions are supersets of X, and their shadow positions are subsets of X. (The term «more» is understood in terms of the cumulative «weight» of the ants - that is, the measure of the corresponding set in the Cantor space.). More precisely, initially, the shadow positions of these ants are temporarily

set to be equal to X, and it is expected that one of the first $l(\varepsilon)$ cats will enter* X (the movement of cats is being simulated from the very beginning until a k cat enters the vertex X), after which the shadow positions are set to be equal to the real ones.

Note*: It is possible that none of the first $l(\varepsilon)$ cats will ever enter X. However, we will choose $l(\varepsilon)$ in such a way that such a cat will be found.

In case if there are several vertices X where conditions for a shift exist, any of them is chosen. Then (with new positions), it is checked whether there is another vertex where the shift condition is met, and a shift is also applied there, and so on, until there are no vertices where a shift is possible. Then the simulation of machine M and the corresponding processes of ant movement and splitting are resumed.

Note: It is possible in principle that in the same vertex, after a shift is performed, there are conditions for another shift (for example, the positions of a large number of ants and their shadow positions were in X, which made the shift possible, but nothing changed after it). However, we do not perform a second shift in the same vertex. (But we check all the others - whether there are any other vertices where a shift is possible. Sooner or later, all possible shifts will be exhausted since there is only a finite number of such vertices where a shift is possible at the current simulation moment.)

2.6 Main Lemma

Lemma. *Let S be a set with a probability of enumeration greater than ε (the threshold from the shift conditions). Let $S' \subset S$ be a finite subset of it. Then at some point, a shift will occur for the set X located between S' and S (i.e., $S' \subset X \subset S$).*

Before proving this lemma, let's show that it implies a desired bound for the complexity of enumerating infinite sets. Consider an increasing sequence of finite sets S' converging to S and apply this lemma to each of them, obtaining, for each, its set X (and one of the first $l(\varepsilon)$ cats servicing this set). Since there is a finite number of cats, one of them must have participated in servicing an infinite number of sets S', never leaving S and being in the supersets of infinitely many sets S'. Therefore, it will eventually enumerate S. Thus, we have:

$$I(S) \leq \log_2 l(\varepsilon)$$

Now we will show how to use this to prove the main theorem.

Proof of Theorem 1. Let k be an integer. We will show that if we set $\varepsilon = 2^{-k}$ and $l(\varepsilon) = 2^{\alpha k + O(\log k)}$, then during a shift, there will always be a cat among the first $l(\varepsilon)$ that comes to X. Using $I(S) \leq \log_2 l(\varepsilon)$, we will prove the theorem.

Consider a randomized machine M' that models the movement of the shadow positions of ants. To do this, M' needs to know ε, so M' uses the first $2\lceil \log_2 k \rceil + 2$ of its bits to determine k and then compute $\varepsilon = 2^{-k}$. Thus, if at some point, a set X

has at least 2^{-k} shadow ants concentrated in it, then M' will have a probability of being in this set of at least $\varepsilon' = 2^{-k-2\log_2 k-3}$. However,

$$\alpha \log_2 1/\varepsilon' + O(\log_2 \log_2 1/\varepsilon') = \alpha k + O(\log_2 k)$$

Therefore, we can choose $l(\varepsilon)$ such that any set enumerated by M' with high probability will be enumerated by one of the first $l(\varepsilon)$ cats. In other words, during a shift, there will always be a cat that comes to X.

Now, we need to prove the main lemma.

3 Proof of the Main Lemma

3.1 Number of Shifts

At every moment t of the process, the Cantor space is divided into cones corresponding to ants. For each ant, we can count the total number of shifts in which it (itself or its ancestors) participated. We can define the set W_k^t as the set in the Cantor space corresponding to ants with k or more shifts by the time t. This is a basic (open and closed) set; the larger k is, the smaller W_k^t (for a given t). On the other hand, W_k^t grows with increasing t. We can take the union over all t and obtain an open set W_k (it is even effectively open since the process is algorithmic - we assume that ε is rational). We obtain a decreasing sequence of open sets:

$$W_1 \supset W_2 \supset W_3 \supset \ldots \supset W_k \supset \ldots$$

in the Cantor space. Membership in the sequence w to the set W_k means that in the chain of ants corresponding to shifts in the direction of the bits of w, there were at least k shifts (in total, for all participants). This chain can be finite or infinite (the same ant, without splitting, can participate in several shifts, or even an infinite number of them).

Now, we can consider the set $W_\infty = \bigcap_k W_k$. It consists of those sequences where the corresponding chain of ants is involved in an infinite number of shifts.

3.2 The Case of Non-empty Intersection

In addition to W_∞, consider the set U in the Cantor space, consisting of those sequences for which the probabilistic machine M enumerates the set S. Do the sets U and W_∞ intersect? We will show that if they do intersect, then the statement of the lemma is true, and then we will contradict the assumption that the intersection is empty.

Let's assume that the intersection is not empty, and w is a sequence belonging to both U and W_∞. Then, with the bits from w, the machine M enumerates S, and the corresponding w ants are involved in an infinite number of shifts. Since the machine M enumerates S, all the positions of these ants will be subsets of S. Furthermore, for our finite

set S', there will be a moment when the position of the ant becomes a superset of S'. This is also true for the shadow position in one of the following moments, because with each shift (and there are infinitely many of them), the shadow position catches up with the real position. After this, another shift will occur, and the set X for this shift will contain S' (because it contains the current shadow position) and will be contained in S (because it is contained in the real position of the same ant).

3.3 The Case of Empty Intersection

We must lead to a contradiction the assumption that the intersection of the sets U (sequences where the machine enumerates S) and $W_\infty = \bigcap_k W_k$ is empty. (Note that we can forget about S' at this point.) Let's assume that it is. Then, the sets $U \cap W_k$ decrease and have an empty intersection. Therefore, their measures tend to zero, and we can find such an N that the measure of the intersection $U \cap W_N$ is very small (less than some δ, which we will choose later and which will be less than the excess of the measure of U above the threshold ε). Choose and fix such N.

The set W_N is open and is a union of an increasing sequence of sets W_N^t. For a sufficiently large T, the difference between the measures of W_N and W_N^T can be made arbitrarily small. Recall that for a given t, the set W_N^t consists of cones (in the Cantor space) over ants (vertices of a binary tree) that have already undergone N shifts by time t. The rest of the ants with their cones form the complement to the set W_N^t. Division of ants does not change these sets, but as time increases, there will be shifts with these ants or their descendants, and then some part of the complement to W_N^t will move to W_N^t. We want this process to be almost complete by the time of T. Moreover, we want similar processes for all sets $W_1, \ldots, W_N$ to be almost complete as well. We will require that the difference between the measures of W_n and W_n^T is less than δ/N for all $n = 1, 2, \ldots, N$. (Since N is fixed before choosing T, this is possible.) This means that the total measure of the ants that will undergo their 1st, 2nd, ..., $(N-1)$th shift after time T (either themselves or their descendants) is not greater than $N \cdot (\delta/N) = \delta$.

So, consider all ants that have undergone fewer than N shifts by time T. What proportion of their continuations is made up of sequences from U (which lead to the enumeration of S)? By construction, U barely intersected with W_N (less than δ), and the intersection with W_N^T will be even smaller. Therefore, the measure we are interested in will be no less than $P(U) - \delta$. Let's look at the positions of these ants. Some of them have exited S (they are not subsets of S), and such ants do not contribute to S. Therefore, if we discard them, we will be left with a finite set of ants for which:

- their current positions are subsets of S (and this is also true for their shadow positions).

- the measure of their continuations that are in U is not less than $P(U) - \delta$.

- the measure of them and their descendants who will undergo a shift is not more than δ.

Let X be a finite subset of S, which is a union of all shadow positions of ants from this set. All points in U enumerate S, so for the corresponding ants, their position will eventually become a superset of X. Applying measure continuity again, we conclude that the measure of the ants whose position will become a superset of X at some point will be no less than $P(U) - 2\delta$. For some of these ants, their shadow position may have changed, but the measure of such ants will not be greater than δ. So if $P(U) - 3\delta > \varepsilon$, conditions for a shift with the set X will be created at some point. Therefore, the measure of the ants that have undergone a shift will be greater than ε, and we can assume without loss of generality that $\delta < \varepsilon$. This will result in a contradiction.

Thus, the case of an empty intersection is impossible.

4 Discussion

One of the most important questions in theoretical computer science is how much more efficient randomized computations are compared to deterministic ones. As Theorem 1 demonstrates, randomized computations are not superior to deterministic ones when considering infinite sets instead of finite ones. Moreover, the gap between complexities can be reduced to a factor of 2 (modulo logarithmic factors). To prove the theorem, we utilized a trick first used by Solovay: instead of cats (the deterministic machine) trying to catch the actual positions of ants (the randomized machine), the cats catch the shadow positions of the ants, which are slightly modified real positions. In Solovay's method, managing shadow positions required transmitting additional $H(S)$ bits of information. We were able to reduce the number of additional bits to $O(\log H(S))$ and, thereby, improve the upper bound.

Can we reduce the upper bound even more or find a matching lower bound? In his work [5], Vereshchagin used Martin's game to obtain an upper bound for finite sets. However, Ageev [1] proved that the quadratic bound is both a lower and upper bound in Martin's game. Thus, to improve the upper bound further, a novel technique must be developed. It is also unknown if the lower bound in Martin's game does extend to a lower bound of deterministic complexity.

5 Acknowledgements

References

[1] M. Ageev. Martin's game: a lower bound for the number of sets. *Theoretical Computer Science*, 289(1):871–876, 2002.

[2] K. de Leeuw, E. F. Moore, C. E. Shannon, and N. Shapiro. Computability by probabilistic machines. In C. E. Shannon and J. McCarthey, editors, *Automata Studies*, pages 183–212. Princeton University Press, Princeton, NJ, 1956.

[3] D. A. Martin. Borel indeterminacy. *Annals of Mathematics*, 102:363–371, 1978.

[4] R. M. Solovay. On random r.e. sets. In R. Chaqui A.I. Arruda, N.C.A. da Costa, editor, *Non-Classical Logics, Model Theory and Computability*, pages 283–307. North-Holland, Amsterdam, 1977.

[5] N. K. Vereshchagin. Kolmogorov complexity of enumerating finite sets. *Inform. Process. Lett.*, 103(1):34–39, 2007.