Sets of k-independent strings*

Ching-Lueh Chang,† Yuh-Danh Lyuu,‡ Yen-Wu Ti,§ Alexander Shen¶

Abstract

A bit string is random (in the sense of algorithmic information theory) if it is incompressible, i.e., its Kolmogorov complexity is close to its length. Two random strings are independent if knowing one of them does not simplify the description of other, i.e., the conditional complexity of each string (using the other as a condition) is close to its length. We may define independence of a k-tuple of strings in the same way.

In this paper we address the following question: what is that maximal cardinality of a set of n-bit strings if any k elements of this set are independent (up to a certain constant)? Lower and upper bounds that match each other (with logarithmic precision) are provided.

1 Randomness and independence

Algorithmic information theory identifies randomness with incompressibility. The *Kolmogorov complexity* of a random string x is defined as minimal length of a program that generates x, provided that the programming language is optimal in terms of program length. There are several versions of this notion (see, e.g., [1, 2] for the exact definition and basic results about Kolmogorov complexity), and we use the so-called *plain complexity* denoted by $\mathrm{C}(x)$. We also consider the *conditional complexity* of a string x when some other string y is given as a condition; it is defined as minimal length of a program that maps y to x, and is denoted by $\mathrm{C}(x|y)$. These notions are defined up to $O(1)$ additive term. We may also speak about complexities of natural numbers and tuples of strings using some computable encoding, and also use these objects as conditions. For simplicity we omit parentheses and write, e.g., $\mathrm{C}(x,y)$ instead of $\mathrm{C}((x,y))$ for complexity of the pair (x,y).

A bit string x is considered as random if it is incompressible, i.e., its complexity $\mathrm{C}(x)$ is close to its length $|x|$. Since we deal with finite strings, there is no sharp dividing line between random and non-random strings. Instead, we define *randomness deficiency* of a n-bit string x as

*The first three authors were supported in part by NSC grant 96-2213-E-002-024. A.S. was supported in part by NAFIT ANR-08-EMER-009-01 grant.

†Dept. Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan. Email: b89053@csie.ntu.edu.tw.

‡Corresponding author. Dept. Computer Science Information and Engineering, National Taiwan University, No.1, Sec. 4, Roosevelt Road, Taipei, Taiwan 106. Email: lyuu@csie.ntu.edu.tw. TEL: 886-2-33664888. FAX: 886-2-23628167.

§Dept. Computer Science Information and Engineering, National Taiwan University, Taipei, Taiwan. Email: d91010@csie.ntu.edu.tw.

¶Laboratoire Informatique Fondamentale, Marseille, France, CNRS & Univ. Aix–Marseille. On leave from IITP RAS, Moscow, Russia. Email: alexander.shen@lif.univ-mrs.fr.

$d(x) = n - \mathrm{C}(x)$. This quantity is non-negative (up to $O(1)$ additive term). One may also define randomness deficiency as $n - \mathrm{C}(x|n)$ (making the deficiency bigger) or $n - \mathrm{C}(x,n)$ (making it smaller): all three quantities coincide with logarithmic precision, as the following proposition (folklore) shows:

Proposition 1. *Let $d = n - \mathrm{C}(x,n)$. Then $n - \mathrm{C}(x|n) \le d + O(\log d)$.*

Proof. Assume that $\mathrm{C}(x|n) = n - u$ and p is a program of length $n - u$ that generates x given n. To generate (unconditionally) both x and n it is enough to specify p and u (knowing p, we know its length; adding u, we reconstruct n and then reconstruct x using p). The pair (p,u) can be encoded by $|p| + 2\log u + O(1) = n - u + 2\log u + O(1)$ bits (the factor 2, or at least some other factor greater than 1, is needed for separation). Therefore, $d \ge u - 2\log u + O(1)$ and this implies $u \le d + O(\log d)$. □

In this paper we always measure the randomness deficiency up to a logarithmic precision, so we can use any of three quantities listed above. Note that we speak about $O(\log d)$- and not $O(\log n)$-terms.

Let x and y be two n-bit strings. We want them not only to be random, but also to be independent (which is a different thing: if x is incompressible but y is equal to x, then x and y are not independent). This idea can be formalized in different ways. One possibility is to require not only that $\mathrm{C}(x)$ is close to n, but also that $\mathrm{C}(x|y)$ is close to n: even knowing y we cannot describe x in a simpler way than just writing all its bits (and vice versa: $\mathrm{C}(y|x)$ should be also close to n). Another possibility is to require that the complexity of (x,y) is close to $2n$ (here (x,y) can be replaced by the concatenation xy since x and y have the same length and can be reconstructed from xy). To understand the relation between these definition, one may use the following (folklore) version of Kolmogorov–Levin information symmetry theorem for deficiencies.

Proposition 2. *Let x and y be two strings of length n. Then*

$$d(x,y) = d(x) + d(y|x) + O(\log d),$$

where $d(x,y) = 2n - \mathrm{C}(x,y)$, $d(y|x) = n - \mathrm{C}(y|x)$, and $d(x) = n - \mathrm{C}(x)$ as defined earlier; the $O(\log d)$ term means that both sides of the equality differ at most by logarithm of one of them (multiplied by a constant).

(Note that if $a = b + O(\log b)$, then $a = \Theta(b)$ and $\log a = \log b + O(1)$, so $b = a + O(\log a)$.)

Proof. The classical Kolmogorov–Levin theorem says that

$$\mathrm{C}(x,y) = \mathrm{C}(x) + \mathrm{C}(y|x) + O(\log n),$$

so it gives the desired equality, but precision is not enough: $\log d$ could be much smaller than $\log n$. However, we can repeat the proof of the theorem and optimize it knowing that complexities of x and y are close to n.

Let $d(x) = u$ and $d(y|x) = v$, i.e., $\mathrm{C}(x) = n - u$ and $\mathrm{C}(y|x) = n - v$. Let p and q be the corresponding programs. To specify (x,y), it is enough to provide p, q, and u (or v), so $\mathrm{C}(x,y) \le |p| + |q| + 2\log u + O(1) = 2n - u - v + 2\log u + O(1)$ and $d(x,y) \ge u + v - O(\log u) = u + v - O(\log(u+v))$. (In the standard argument we specify $|p|$ instead of $n - |p|$, and this may require $O(\log n)$ bits.)

On the other hand, let $d(x,y) = w$, i.e., $\mathrm{C}(x,y) = 2n - w$. Consider the set S of all pairs (x,y) of n-bit strings such that $\mathrm{C}(x,y) \le 2n - w$. Consider its x-section $S_x = \{t \mid (x,t) \in S\}$. We claim that $\log_2 \#S_x = n - O(w)$. Indeed, let $\log_2 \#S_x = n - m$ for some m (rounded to an integer). To specify (x,y) when n is known it is enough to list x (n bits), w (requires $O(\log w)$ bits for a self-delimiting description), m (requires $O(\log m)$ bits) and the ordinal number of y in S_x (in the enumeration order, requires $n - m + O(1)$ bits). In total we get $2n - m + O(\log m) + O(\log w)$, so

$$2n - w - O(\log w) = \mathrm{C}(x,y|n) \le 2n - m + O(\log m) + O(\log w),$$

i.e., $m - O(\log m) \le w + O(\log w)$, so $m \le w + O(\log w) = O(w)$.

Describing y when x and n are known by w, m, and the ordinal number of y in S_x, we see that

$$\mathrm{C}(y|x,n) \le n - m + O(\log w)$$

(the bound $m = O(w)$ is used to replace $\log m$ by $\log w$). Also there is at most

$$2^{(2n-w)-(n-m)+O(1)} = 2^{n+m-w+O(1)}$$

string x' such that $S_{x'}$ has cardinality at least 2^{n-m-1}, and x is among them, so x can be specified (given n) by specifying m, w, and the ordinal number of x in the enumeration ($n + m - w + O(1)$ bits), i.e.,

$$\mathrm{C}(x|n) \le n + m - w + O(\log w).$$

In total we get

$$\mathrm{C}(x|n) + \mathrm{C}(y|x,n) \le n - m + n + m - w + O(\log w) = 2n - w + O(\log w),$$

so

$$d(x) + d(y|x) \ge d(x,y) - O(\log d(x,y))$$

(recall that condition n in the deficiency can be ignored since our equalities are true with logarithmic precision, proposition 1). □

Now recall our discussion about two ways to define independent pairs of random strings. If $d(x,y)$ is small, say, less than some D, then both $d(x)$ and $d(y|x)$ are small: they are at most $D + O(\log D)$. In the other direction we have factor 2: if $d(x)$ and $d(y|x)$ do not exceed D, then $d(x,y)$ does not exceed $2D + O(\log D)$. So the first requirement is stronger.

Similar result (with the same proof) holds for tuples: for example, for triples we have

$$d(x,y,z) \approx d(x) + d(y|x) + d(z|x,y)$$

(with the same precision: the difference between two sides is bounded by a logarithm of each side multiplied by a constant).

2 Sets of pairwise independent strings

A classical (and simple) observation: most n-bit strings have deficiency at most 2. Now let us ask similar question adding the independence requirement. Let S be a set of n-bit strings. Assume that $d(x,y) \le D$ for every (different) $x, y \in S$. What is the maximal possible cardinality of S? The same question can be asked with condition $d(x|y) \le D$. It turns out that in both cases we have maximal cardinality $2^{D+O(\log D)}$. Let us prove the corresponding upper and lower bounds.

Theorem 1. (1) *Let S be a set of n-bit binary strings such that $d(x|y) \le D$ for every two different $x, y \in S$. Then $\#S \le 2^{D+O(\log D)}$.*

(2) *There exists a set S of n-bit binary strings of size $2^{D-O(\log D)}$ such that $d(x,y) \le D$ for every two different $x, y \in S$.*

Note that to make the theorem stronger, we use different interpretation of independence in the upper and lower bounds.

Proof. The upper bound is simple: if $\#S > 2^m$, then S has two different elements x, y with the same m-bit prefix, so $d(x|y) \ge m - O(\log m)$. (Indeed, to specify y when x is given, it is enough to specify m (using $O(\log m)$ bits for self-delimiting description) and list the last $n-m$ bits of y.

To prove the lower bound, we first restrict ourselves to strings x with deficiency $d(x) = O(1)$; we have $\Omega(2^n)$ of them. Let x_1 be one of these strings. Delete all the strings y such that $d(y|x_1) \ge D$. There are at most $2^{n-D+O(1)}$ strings to be deleted. Let x_2 be one of the remaining strings; delete also all the strings y such that $d(y|x_2) \le D$. Continuing this process, we get a set $S = \{x_1, x_2, \ldots\}$ of at least $2^{D-O(1)}$ strings. For every i we have $d(x_i) = O(1)$ and for every i, j such that $i < j$ we have $d(x_j|x_i) \le D + O(1)$. Therefore (proposition 2) we have $d(x_i, x_j) \le D + O(\log D)$.

□

It is instructive to provide an alternative argument for the lower bound (which can be generalized for tuples, see section 3). Let N be some number (to be chosen later). Consider N independent random variables $X_1, \ldots, X_N$ uniformly distributed among n-bit strings. For fixed $i \ne j$ the probability of the event $d(X_i, X_j) > D$ is $2^{-D+O(1)}$. For a fixed i the probability of the event "$d(X_i, X_j) > D$ for some $j \ne i$" is bounded by $O(N2^{-D})$, and the expected number of i with this property (we call them "bad") is $NO(N2^{-D})$. Taking $N = 2^{D-c}$ for large enough constant c, we make this expected value less than $N/2$. Therefore, with positive probability only half of X_i are bad. Fix some outcome when this happens. Deleting all bad strings, we get a set S of size $2^{-D+O(1)}$ such that $d(x,y) \le D$ for every different $x, y \in S$.

The lower bound can be considered as a generalization of a lower bound for the code size: In coding theory we try to construct strings that have large Hamming distance. Our requirement is stronger: we want them to be independent (note that strings x and y that have Hamming distance significantly less than $n/2$ are dependent). One can construct a code by choosing codewords one by one (outside of the existing balls) or choose them all together randomly (and then deleting elements that are close to other elements). Both arguments, as we have seen, easily generalize to our case.

3 Sets of k-wise independent strings

Similar upper and lower bounds can be shown for tuples. Fix some number $t \ge 2$. (All the constants in the $O()$-notation below depend on this t.)

Theorem 2. (1) *Let S be a set of n-bit strings such that $d(x_1|x_2, \ldots, x_t) \le D$ for every two different $x_1, \ldots, x_t \in S$. Then $\#S \le 2^{D/(t-1)+O(\log D)}$.*

(2) *There exists a set S of n-bit strings of size $2^{D/(t-1)-O(\log D)}$ such that $d(x_1, \ldots, x_t) \le D$ for every different $x_1, \ldots, x_t \in S$.*

For $t = 2$ we get theorem 1.

Proof. Let us start with the second part (lower bound) that can be obtained as above. For some N consider N independent random variables $X_1, \ldots, X_N$ that are uniformly distributed among n-bit strings. For fixed $i_1, \ldots, i_t$ the probability of the event

$$d(X_{i_1}, \ldots, X_{i_t}) > D$$

does not exceed $2^{-D+O(1)}$. Therefore, for given i_1 the probability of the event "there exist $i_2, \ldots, i_t$ such that..." does not exceed $N^{t-1}2^{-D+O(1)}$. If the latter probability is less than $1/2$, then with positive probability only half of i_1 are bad. So letting $N = 2^{D/(t-1)-O(1)}$, we get $\Omega(N)$ good strings (as required).

The upper bound can be shown using a Muchnik-type combinatorial argument. In section 2 we used the following (evident) fact: two strings are dependent if they have the same prefix. However, it is not needed for prefixes to be the same: it is enough for one of the prefixes to be a simple function of the other one. For the same reason, $d(x|y,z)$ is big if a prefix of x (of a significant size) is a simple function of y and z or their prefixes. Following this idea, we prove a combinatorial statement.

Lemma 1. *For every m there exist a function $f\colon (x_2, \ldots, x_t) \mapsto f(x_2, \ldots, x_t)$ whose arguments $x_2, \ldots, x_t$ and values are m-bit binary strings, with the following property: every set $S \subset \mathbb{B}^m$ with $\#S > 2^{m/(t-1)+O(\log m)}$ contains pairwise different elements $x_1, x_2, \ldots, x_t$ such that $x_1 = f(x_2, \ldots, x_t)$.*

Before proving this lemma, let us show how it can be used to prove our theorem. Note that we may assume without loss of generality that the function f in the lemma has complexity $O(\log m)$. Indeed, if a function with this property exists, it can be found by an exhaustive search (given m). For a given set $S \in \mathbb{B}^n$ of size 2^s or bigger, let us consider m-bit prefixes of all the elements of S, where $m = D + c\log D$ for large enough constant c (to be chosen later). If two prefixes of different elements of S coincide, we have found $x, y \in S$ such that $d(x|y) > D$ (and we may use $t-2$ arbitrary strings as a condition). If not, the set of prefixes contains 2^s elements. If $s > m/(t-1) + O(\log m)$, the lemma guarantees that $y_1 = f(y_2, \ldots, y_t)$ for some prefixes $y_1, \ldots, y_t$ (all different) of strings $x_1, \ldots, x_t \in S$ (also different). This implies $d(x_1|x_2, \ldots, x_t) \geq m - O(\log m)$, and $m - O(\log m) > D$ if the constant c (see above) is large enough. This should not happen for our S, so $s \leq m/(t-1) + O(\log m) = D/(t-1) + O(\log D)$. This finishes the proof of the theorem modulo our lemma. It remains to prove the lemma.

Proof of the lemma. We use the probabilistic argument and show that a random function f has the required property with positive probability. The number of different sets S of size 2^s is at most $(2^m)^{2^s}$. For each set S of this size we consider the event "$f(x_2, \ldots, x_t) \notin S$" for every tuple $(x_2, \ldots, x_t)$ of different elements of S. For each tuple the probability is $1 - 2^s/2^m$, and for different tuples the events are independent. The number of tuples is $(2^s)^{t-1}/O(1)$ (we divide by $(t-1)!$ and should take into account that elements in tuple are different; note that constants hidden in O-notation depend on t). So we get the following upper bound for the probability:

$$(2^m)^{2^s} \left(1 - \frac{2^s}{2^m}\right)^{2^{s(t-1)-O(1)}}$$

If $s = m/(t-1) + c\log m$, this equals

$$(2^m)^{2^s}\left(1-\frac{1}{2^{m-s}}\right)^{2^{m+c(t-1)\log m-O(1)}} = (2^m)^{2^s}\left(\left(1-\frac{1}{2^{m-s}}\right)^{2^{m-s}}\right)^{2^{s+c(t-1)\log m-O(1)}}$$

Recall that $(1-1/u)^u \approx e$, we get the upper bound

$$(2^m)^{2^s}\left(\frac{1}{e}\right)^{2^{s+c(t-1)\log m-O(1)}}$$

(to be exact, we should use some $e' < e$ instead of e). We have to prove that the bound is less than 1, so we may delete factor 2^s in the exponents and show that

$$2^m\left(\frac{1}{e}\right)^{2^{c(t-1)\log m-O(1)}} < 1,$$

which is evident if c is large enough. This finishes the proof of the lemma.

□

Acknowledgements

Authors are grateful to Paul Vitanyi, the Editor of this Journal.

References

[1] Li M., Vitányi P., *An Introduction to Kolmogorov Complexity and Its Applications*, Second Edition, Springer, 1997. (638 pp.)

[2] Alexander Shen, *Algorithmic Information Theory and Kolmogorov Complexity*, Uppsala Universitet, Technical Report 2000-034. Available at: `http://www.it.uu.se/research/publications/reports/2000-034`.