New Applications of the Incompressibility Method

(*Extended Abstract*)

Harry Buhrman[1], Tao Jiang[2], Ming Li[3], and Paul Vitányi[1]

[1] CWI, Kruislaan 413, 1098 SJ Amsterdam,
The Netherlands, {buhrman,paulv}@cwi.nl
Supported in part via NeuroCOLT II ESPRIT Working Group
[2] Dept of Computing and Software, McMaster University,
Hamilton, Ontario L8S 4K1, Canada, jiang@cas.mcmaster.ca
Supported in part by NSERC and CITO grants
[3] Dept of Computer Science, University of Waterloo,
Waterloo, Ontario N2L 3G1, Canada, mli@math.uwaterloo.ca
Supported in part by NSERC and CITO grants and Steacie Fellowship

Abstract. The incompressibility method is an elementary yet powerful proof technique based on Kolmogorov complexity [13]. We show that it is particularly suited to obtain average-case computational complexity lower bounds. Such lower bounds have been difficult to obtain in the past by other methods. In this paper we present four new results and also give four new proofs of known results to demonstrate the power and elegance of the new method.

1 Introduction

The incompressibility of individual random objects yields a simple but powerful proof technique: *the incompressibility method.* This method is a general purpose tool that can be used to prove lower bounds on computational problems, to obtain combinatorial properties of concrete objects, and to analyze the average complexity of an algorithm. Since the early 1980's, the incompressibility method has been successfully used to solve many well-known questions that had been open for a long time and to supply new simplified proofs for known results. Here we demonstrate how easy the incompressibility method can be used in the particular case of obtaining average-case computational complexity lower bounds. The purpose is to show that our average-case analyses are easy while such analyses using traditional methods are usually more difficult than worst-case analyses.

The Incompressibility Method: A general introduction to the theory and applications of Kolmogorov complexity can be found in [13]. A brief exposition of the basic concepts and notations required in the incompressibility method is also given in [12]. We need the following easy fact (sometimes only implicitly).

Lemma 1. *Let c be a positive integer. For every fixed y, every finite set A contains at least $(1-2^{-c})|A|+1$ elements x with $C(x|A,y) \geq \lfloor \log |A| \rfloor - c$.*

In a typical proof using the incompressibility method, one first chooses an incompressible object (that is, having—almost—maximal Kolmogorov complexity) from the class under discussion. The argument invariably says that if a desired property does not hold for this object, then the object can be compressed. This yields the required contradiction. Furthermore, since most objects are incompressible, the desired property usually holds on average.

Results: We give proofs for new results on: space filling curve fitting lower bounds, multidimensional random walks, communication complexity (average-case[1]) and the number of strings of which the complexity exceeds their length. We give new proofs for known results on: boolean matrix multiplication, majority finding, random walks, and communication complexity (worst case). Our new proofs are much simpler than the old ones.

Related Work: A survey of the use of the incompressibility method is [13] Chapter 6. The most spectacular successes of the method occur in the computational complexity analysis of algorithms. Applications in combinatorics are [14], in random graph theory [5], and a recent average-case analysis of Shellsort is given in [12].

2 Complexity often Exceeds Length

Applications of the incompressibility method sometimes require a large number of totally incompressible strings. Lemma 1 states that there is at least one string of every length n so that $C(x) \geq n$. We show here that this can be strengthened considerably: (i) A positive fraction of all strings of length n have complexity at least n; (ii) There is a fixed constant c such that for every k there is an n with $kc \leq n < (k+1)c$ such that a positive fraction of all strings of length n have complexity exceeding n (here c, k are positive integers). Item (i) can be formulated slightly more general:

Lemma 2. [2] *There is a constant $d > 0$ such that for every n there are at least $\lfloor 2^n/d \rfloor$ strings x of length n with $C(x|n) \geq n$ (respectively, $C(x) \geq n$).*

Proof. It is well-known that there is a constant $c \geq 0$ such that for every n every string x of length n has $C(x|n) \leq n + c$. Hence for every n and every x of length $l(x) \leq n - 2\log(n - l(x)) - c - 1$ we have $C(x|n) \leq C(x|l(x)) + 2\log(n - l(x)) < n$. Consequently, for some constant $c' > 0$ there are at most $2^n - 2^{n-c'}$ programs of length $< n$ available as shortest programs for the strings of length n (there are $2^n - 1$ potential programs and $2^{n-c'} - 1$ thereoff are already taken). Hence there are at least $2^{n-c'}$ strings x of length n with $C(x|n) \geq n$.

In the unconditional case the proof is simpler and for $C(x|l(x)) \leq l(x) + c$ we find that there are at least 2^{n-c} strings x of length n with $C(x) \geq n$. This is

[1] As far as we could ascertain there is no written proof of this fact although the result may be known, see Section 7.

[2] A similar result for the *prefix version* $K(\cdot)$ of Kolmogorov complexity (also called *self-delimiting complexity*) is given in [6] using a more complicated argument. Note that neither result implies the other.

because we can dispense with the $2\log(n-l(x))$ term induced by the conditional 'n'. □

Can we prove that the complexity of many strings must *exceed* their lengths? The proof uses an incompressibility argument to show *negative* incompressibility.

Lemma 3. *There are constants $c, d > 0$ such that for every large enough n there are at least $\lfloor 2^n/d \rfloor$ strings x with $n - c \leq l(x) \leq n$ satisfying $C(x|n) > n$ (respectively, $C(x) > n$).*

Proof. **Conditional case:** For every n there are equally many strings of length $\leq n$ to be described and potential programs of length $\leq n$ to describe them. Since some programs do not halt for every large enough n there exists a string x of length at most n such that $n < C(x|n) \leq l(x) + c$.

Let there be $m \geq 1$ such strings. Given m and n we can enumerate all $2^{n+1}-m-1$ strings x of length $\leq n$ and complexity $C(x|n) \leq n$ by dovetailing the running of all programs of length $\leq n$. The lexicographic first string of length $\leq n$ not in the list, say x, is described by a program p giving m in $\log m$ bits plus an $O(1)$-bit program to do the decoding of x. Therefore, $\log m + O(1) \geq C(x|n) > n$ which proves the theorem for the conditional case.

Unconditional case: This follows similarly by padding the description of x up to length $n + c'$ for a constant c' and adding the description of c' to program p describing x. This way we can first retrieve c' from p and then retrieve n from the length of p. □

So there are lots of strings x that have complexity larger than their lengths. How much larger can this excess get? In the theory of Kolmogorov complexity the laws are invariant with respect to the choice of the particular reference universal Turing machine in the definition of the Kolmogorov complexity, the excess of maximal complexity over the length depends on this choice. Since one can choose a reference universal Turing machine such that $C(x) \leq l(x) + 1$ we cannot generally prove that the excess exceeds 1.

3 Average Time for Boolean Matrix Multiplication

Here is a simple illustration of average-case analysis using the incompressibility method. Consider the well-known problem of multiplying two $n \times n$ boolean matrices $A = (a_{i,j})$ and $B = (b_{i,j})$. Efficient algorithms for this problem have always been a very popular topic in the theoretical computer science literature due to the wide range of applications of boolean matrix multiplication. The best worst-case time complexity obtained so far is $O(n^{2.376})$ due to Coppersmith and Winograd [7]. In 1973, O'Neil and O'Neil devised a simple algorithm described below which runs in $O(n^3)$ time in the worst case but achieves an average time complexity of $O(n^2)$ [19].

Algorithm QuickMultiply(A, B)

1. Let $C = (c_{i,j})$ denote the result of multiplying A and B.
2. For $i := 1$ to n do
3. Let $j_1 < \cdots < j_m$ be the indices such that $a_{i,j_k} = 1$, $1 \leq k \leq m$.
4. For $j := 1$ to n do
5. Search the list $b_{j_1,j}, \ldots, b_{j_m,j}$ sequentially for a bit 1.
6. Set $c_{i,j} = 1$ if a bit 1 is found, or $c_{i,j} = 0$ otherwise.

An analysis of the average-case time complexity of QuickMultiply is given in [19] using probabilitistic arguments. Here we give a simple and elegant proof using incompressibility.

Theorem 1. *Suppose that the elements of A and B are drawn uniformly and independently. Algorithm QuickMultiply runs in $\Theta(n^2)$ time on the average.*

Proof. Let n be a sufficiently large integer. The average time of QuickMultiply is trivially bounded between $\Omega(n^2)$ and $O(n^3)$. By Lemma 1, out of the 2^{2n^2} pairs of $n \times n$ boolean matrices, at least $(n-1)2^{2n^2}/n$ of them are $\log n$*-incompressible* (with Kolmogorov complexities at least $2n^2 - \log n$ bits). Hence, it suffices to consider $\log n$-incompressible boolean matrices.

Take a $\log n$-incompressible binary string x of length $2n^2$, and form two $n \times n$ boolean matrices A and B by having the first half of x correspond to the row-major listing of the elements of A and the second half of x correspond to the row-major listing of the elements of B. We show that QuickMultiply spends $O(n^2)$ time on A and B.

Consider an arbitrary i, where $1 \leq i \leq n$. It suffices to show that the n sequential searches done in Steps 4 – 6 of QuickMultiply take a total of $O(n)$ time. By the statistical results on various blocks in incompressible strings given in Section 2.6 of [13], we know that at least $n/2 - O(\sqrt{n \log n})$ of these searches find a 1 in the first step, at least $n/4 - O(\sqrt{n \log n})$ searches find a 1 in two steps, at least $n/8 - O(\sqrt{n \log n})$ searches find a 1 in three steps, and so on. Moreover, we claim that none of these searches take more than $4 \log n$ steps. To see this, suppose that for some j, $1 \leq j \leq n$, $b_{j_1,j} = \cdots = b_{j_{4 \log n},j} = 0$. Then we can encode x by listing the following items in a self-delimiting manner: (1) A description of the above discussion; (2) The value of i; (3) The value of j; (4) All bits of x except the bits $b_{j_1,j}, \ldots, b_{j_{4 \log n},j}$. This encoding takes at most

$$O(1) + 2 \log n + 2n^2 - 4 \log n + O(\log \log n) < 2n^2 - \log n$$

bits for sufficiently large n, which contradicts the $\log n$-incompressibility of x.

Hence, the total number of steps required by the n searches is at most

$$\begin{aligned}
&\sum_{k=1}^{4 \log n} ((n/2^k - O(\sqrt{n \log n})) \cdot k) + (\log n) \cdot O(\sqrt{n \log n}) \cdot (4 \log n) \\
&< \sum_{k=1}^{\log n} kn/2^k + O(\log^2 n \sqrt{n \log n}) \\
&= O(n) + O(\log^2 n \sqrt{n \log n}) \;=\; O(n). \square
\end{aligned}$$

4 Space Filling Curves

Niedermeier, Reinhardt and Sanders recently studied the following problem [18]: In an $n \times n$ mesh, consider a computable *curve fitting scheme* that maps the numbers from $\{1, \ldots, n^2\}$ into the mesh, with each number occupying a unique point in the mesh. The goal is to minimize the Euclidean distance between numbers relative to their absolute difference. Many algorithms in parallel computing, computational geometry, and image processing depend on such "locality-preserving" curve fitting schemes for meshes. [18] shows that for any curve fitting scheme, there exists a pair i and j such that $d(i,j) \geq \sqrt{3.5|i-j|} - 1$, where d is Euclidean distance. However, for both theoretical and practical reasons, it would be more interesting to establish distance bounds that hold not only for one pair of i, j's but for "many" pairs. The question of such an "average-case analysis" was open. In fact, many experiments have been performed in the past by researchers in order to determine the average distance between two numbers in a curve fitting scheme [18]. We present the first "average-case" bound here. Our incompressibility argument is in fact simpler than the worst-case analysis in [18].

Theorem 2. *In any computable curve fitting scheme, for each number* $1 \leq i \leq n^2$, $d(i,j) \geq \sqrt{0.636|i-j|}$ *for* $\Omega(n^2)$ *different* j*'s. Furthermore, if* i *is mapped to a corner point, then* $d(i,j) \geq \sqrt{2.5|i-j|}$ *for* $\Omega(n^2)$ *different* j*'s.*

Proof. Let $N = n^2$. Consider a computable curve fitting scheme F and let i be a fixed number between 1 and N. Consider incompressible j's satisfying

$$C(j|i,P) \geq \log N, \tag{1}$$

where P is a fixed program to be defined below. It follows from Lemma 2 that there is a constant $c > 0$ such that there are N/c such j's satisfying (1).

Also, we can argue that $|i-j| \leq N/2$ for at least half of the j's. Since if this is not the case, we can change the universal TM in the definition of Kolmogorov complexity by just making the new universal TM print 0 (or 1) whenever the old universal TM prints 1 (or 0, respectively). Then for each j, let $\bar{j}$ be the 1's complement of j, we have either $|i-j| \leq N/2$ or $|i-\bar{j}| \leq N/2$. For all the j's satisfying Inequality (1), if more than half of them do not satisfy $|i-j| \leq N/2$, then we can use the new universal TM such that more than half of the $\bar{j}$'s satisfy $|i-\bar{j}| \leq N/2$. And under the new universal TM, the $\bar{j}$'s satisfy Inequality (1) if the j do so under the old universal TM.

Now given i, an index j can be reconstructed by a fixed program P from a description of at most $\log \pi d(i,j)^2$ bits. [3] [4] Thus, we have

$$\log \pi d(i,j)^2 \geq C(j|i,P) \geq \log N$$

[3] Since i and j are placed on the grid points of an $n \times n$ grid the value of $d(i,j)^2$ is an integer. Hence the precise description can be expressed in this claimed number of bits. While it would suffice to describe j by the index of an enumeration of points on the circumference of the disc which would cost $\log 2\pi d(i,j)$ bits this is not possible since $d(i,j)$ is radical.

[4] Here $\pi d(i,j)^2$ is the area of a disc centered on i with radius $d(i,j)$. Since the curve fitting scheme is computable we can enumerate all candidate's for j that are fitted in this circle. The index of j in this enumeration suffices to find j.

for $(N/2c)$-many j's. This implies $d(i,j) \geq \sqrt{N/\pi} \geq \sqrt{2|i-j|/\pi} \approx \sqrt{0.636|i-j|}$ for $(N/2c)$ j's. If F puts i at a corner point, then an index j can in fact be specified by $\log \frac{1}{4}\pi d(i,j)^2$ bits. Carrying out the above calculation, we obtain that

$$d(i,j) \geq \sqrt{4N/\pi} \geq \sqrt{8|i-j|/\pi} \approx \sqrt{2.5|i-j|}$$

for $(N/2c)$-many j's. □

This argument also applies to obtain similar results for other distances (including l_∞ and Manhattan) mentioned in [18].

5 Average Complexity of Finding the Majority

Let $x = x_1 \cdots x_n$ be a binary string. The *majority bit* (or simply, the *majority*) of x is the bit (0 or 1) that occurs more than $\lfloor n/2 \rfloor$ times in x. The majority problem is that, given a binary string x, determine a position i such that x_i is the majority bit of x using only bit comparisons. When x has no majority, we must report so.

The time complexity for finding the majority has been well studied in the literature (see, *e.g.* [1–3, 11, 22]). It is known that, in the worst case, $n - \nu(n)$ bit comparisons are necessary and sufficient [2, 22], where $\nu(n)$ is the number of occurrences of bit 1 in the binary representation of number n. Recently, Alonso, Reingold and Schott [3] studied the average complexity of finding the majority assuming the uniform probability distribution model. Using quite sophisticated arguments based on decision trees, they showed that on the average finding the majority requires at most $2n/3 - \sqrt{8n/9\pi} + O(\log n)$ comparisons and at least $2n/3 - \sqrt{8n/9\pi} + \Theta(1)$ comparisons. Here we present a new simple incompressibility proof establishing an upper bound on the average-case complexity of finding the majority which is precise in the first term. Our proof uses the following standard algorithm.

Algorithm Tournament($x = x_1 \cdots x_n$)

1. If $n \leq 3$ then find the majority directly.
2. Let $y = \epsilon$.
3. For $i := 1$ to $\lfloor n/2 \rfloor$ do
4. If $x_{2i-1} = x_{2i}$ then append the bit x_{2i} to y.
5. If n is odd and $\lfloor n/2 \rfloor$ is even then append the bit x_n to y.
6. Call Tournament(y).

Theorem 3. *On the average, algorithm Tournament requires at most* $2n/3 + O(\sqrt{n})$ *comparisons.*

Proof. Let n be a sufficiently large number. Again, since algorithm Tournament makes at most n comparisons on any string of length n, by Lemma 1, it suffices to consider running time of Tournament on δ-incompressible strings, where $\delta \leq \log n$. Consider an arbitrary $\delta \leq \log n$ and let $x = x_1 \cdots x_n$ be a fixed

δ-incompressible binary string. For any integer $m \leq n$, let $\sigma(m)$ denote the maximum number of comparisons required by algorithm Tournament on any δ-incompressible string of length m.

Among the $\lfloor n/2 \rfloor$ bit pairs $(x_1, x_2), \ldots, (x_{2\lfloor n/2 \rfloor - 1}, x_{2\lfloor n/2 \rfloor})$ that are compared in step 4 of Tournament, there are at least $n/4 - O(\sqrt{n\delta})$ pairs consisting of complementary bits, [13]. Clearly, the new string y obtained at the end of step 4 should satisfy $C(y) \geq l(y) - \delta - O(1)$. Hence, we have the following recurrence relation for $\sigma(m)$:

$$\sigma(m) \leq \lfloor m/2 \rfloor + \sigma(m/4 + O(\sqrt{m\delta}))$$

By straightforward expansion, we obtain that

$$\begin{aligned}
\sigma(n) &\leq \lfloor n/2 \rfloor + \sigma(n/4 + O(\sqrt{n\delta})) \\
&\leq n/2 + \sigma(n/4 + O(\sqrt{n\delta})) \\
&\leq n/2 + (n/8 + O(\sqrt{n\delta})/2) + \sigma(n/16 + O(\sqrt{n\delta})/4 + O(\sqrt{(n\delta)/4})) \\
&= n/2 + (n/8 + O(\sqrt{n\delta})/2) + \sigma(n/16 + (3/4) \cdot O(\sqrt{n\delta})) \\
&\leq \cdots \leq 2n/3 + O(\sqrt{n\delta})
\end{aligned}$$

Using Lemma 1, we can calculate the average complexity of algorithm Tournament as: $\sum_{\delta=1}^{\log n} \frac{1}{2^\delta}(2n/3 + O(\sqrt{n\delta})) + \frac{1}{n}n = 2n/3 + O(\sqrt{n})$ □

6 Multidimensional Random Walks

Consider a random walk in 1 dimension with fixed probability $p = \frac{1}{2}$ of taking a unit step left or right. It is well-known that the maximal distance from the start position in either direction in a random walk of n steps is in the order of $\sqrt{n}$ with high probability. For example, the *Law of the Iterated Logarithm*, [15], says that the limit superior of this distance equals $\sqrt{\frac{1}{2}n \log\log n}$ with probability 1 for n rises unboundedly. In a random walk in $k > 1$ dimensions where each step increases or decreases the distance from the origin by a unit in exactly one dimension in many applications we would like to know the probability of traveling distance d from the origin in any dimension in n steps. Nonetheless, probabilistic analyses of random walks as in [10, 21] apparently are not concerned with flexible tradeoffs between probability (here based on randomness deficiency) [5] and absolute upper or lower bounds on the largest distance traveled from the origin in every dimension as given in the theorem below. These new results may be very useful in applications in the theory of computation.

[5] With this approach to random walks we can by varying the complexity of the walk (which implies varying the probability of such a walk in the sense that low complexity has high probability and higher complexity less probability) regulate the possible variation in the distance covered in the walk (high complexity walks have precisely fixed distance while low complexity walks have more uncertainty).

Theorem 4. *Consider a random walk in k dimensions where each step is a unit step in any (but only one at a time) single dimension in positive or negative direction with uniform probability $1/2k$. Let $\delta(\cdot)$ be a monotonic nondecreasing function and let x be a random walk of length n such that $C(x|n) > n - \delta(n)$. If $n \gg k$ then the random walk x has all of the following properties (which therefore hold with probability at least $1 - 1/2^{\delta(n)}$ for a random walk of length n):*

(i) *For every dimension, the maximal distance the walk moves away from the starting position in either direction during the walk is $O\left(\sqrt{\frac{n}{k}(\delta(n) + \log \frac{n}{k})}\right)$;*

(ii) *For every dimension, the maximum distance the walk is away from the starting position in either direction at the end of the walk is $O\left(\sqrt{\delta(n)\frac{n}{k}}\right)$; and*

(iii) *For every dimension, the minimum distance the walk is away from the starting position in either direction at the end of the walk is $\Omega\left(\sqrt{2^{-\delta(n)}\frac{n}{k}}\right)$.*

(iv) *For every dimension, the minimum distance the walk is away from the starting position in either direction at the end of an initial m-length segment x' with $x = x'z$ for some z, $C(x'|m) > m - \delta(m)$, and $m \gg k$, is $\Omega\left(\sqrt{2^{-\delta(m)}\frac{m}{k}}\right)$.*

Proof. To be given in the full version. □

7 Communication Complexity

Consider the following communication complexity problem (for definitions see the book by Kushilevitz and Nisan [17]). Initially, Alice has a string $x = x_1, \ldots, x_n$ and Bob has a string $y = y_1, \ldots, y_n$ with $x, y \in \{0,1\}^n$. Alice and Bob use an agreed-upon protocol to compute the inner product of x and y modulo 2

$$f(x,y) = \sum_{i=1}^{n} x_i \cdot y_i \bmod 2$$

with Alice ending up with the result. We are interested in the minimal possible number of bits used in communication between Alice and Bob in such a protocol. Here we prove a lower bound of n, which is tight since the trivial protocol where Bob sends all his n bits to Alice achieves this bound. In [17] the same lower bound is obtained by a different method. We also show an $n - O(1)$ lower bound for the average-case complexity. This lower bound isn't mentioned in [17] and is not implied by the lower bound in in exercise 3.30 in that reference. However, according to [20] exercise 3.30 was proven using a stronger version of our average-case result but the proof may not be written down nor remembered. It seems useful to have a written version as we present below.

Theorem 5. *Assume the discussion above. Every protocol computing the inner product function requires at least n bits of communication.*

Proof. Fix a communication protocol P that computes the inner product. Let A be an algorithm that we describe later. Let z be a string of length $2n$ such that $C(z|A,P,n) \geq 2n$. Let $z = x_1 \ldots x_n y_1 \ldots y_n$ Let Alice's input be $x = x_1 \ldots x_n$

and Bob's input be $y_1 \ldots y_n$. Assume without loss of generality that $f(x,y) = 0$ (the innerproduct of x and y is 0 modulo 2). [6] Run the communication protocol P between Alice and Bob ending in a state where Alice outputs that $f(x,y)$ is 0. Let C be the sequence of bits sent back and forth. Note that P can be viewed as a tree with C a path in this tree [17]. Hence C is self-delimiting. Consider the set S defined by

$$S := \{a : \exists b \text{ such that } P(a,b) = 0 \text{ and induces conversation C } , a, b \in \{0,1\}^n\}.$$

Given n, P and C, we can compute S. Let the cardinality of S be l. The strings in S form a matrix M over GF(2) with the ith row of M corresponding to the ith string in S (say in lexicographic ordering). Since for every $a \in S$ it holds that $f(a,y) = 0$ it follows that y is an element of the Null space of M ($y \in \text{Null}(M)$). Application of the Null space Theorem from linear algebra yields:

$$\text{rank}(M) + \dim(\text{Null(M)}) = n. \tag{2}$$

Since the cardinality of S is l and we are working over GF(2) it follows that the rank of M is at least $\log(l)$ and by (2) it follows that $\dim(\text{Null}(M)) \leq n - \log(l)$. The following is an effective description of z given n and the reconstructive algorithm A explained below:

1. C;
2. the index of $x \in S$ using $\log(l)$ bits; and
3. the index of $y \in \text{Null}(M)$ with $n - \log(l)$ bits.

The three items above can be concatenated without delimiters. Namely, C itself is self-delimiting, while from C one can generate S and hence compute l. From the latter item one can compute the binary length of the index for $x \in S$, and the remaining suffix of the binary description is the index for $y \in \text{Null}(M)$. From the given description and P, n the algorithm A reconstructs x and y and outputs $z = xy$. Consequently, $C(z|A,P,n) \leq l(C) + \log l + (n - \log l)$. Since we have assumed $C(z|A,P,n) \geq 2n$ it follows that $l(C) \geq n$. □

Theorem 6. *The average communication complexity of computing the inner product of two n-bit strings is at least $n - O(1)$ bits.*

8 Acknowledgements

We thank Bill Gasarch for comments, Ian Munro for discussions on related subjects, Rolf Niedermeier for discussions on mesh indexing and their paper [18], and Bill Smyth for introducing us to the paper [3], and Osamu Watanabe for drawing our attention to the random walk problem.

[6] If this is not the case, we can simply use a different universal TM U' that does precisely what the current universal TM U does except it flips the first bit of every output. Thus z with first bit flipped would have Kolmogorov complexity $2n$ under U'.

References

1. L. Alexanderson, L.F. Klosinski and L.C. Larson, *The William Lowell Putnam Mathematical Competition, Problems and Solutions: 1965-1984*, Mathematical Association of America, Washington, DC, 1985.
2. L. Alonso, E. Reingold and R. Schott, Determining the majority, *Information Processing Letters* 47, 1993, pp. 253-255.
3. L. Alonso, E. Reingold and R. Schott, The average-case complexity of determining the majority, *SIAM Journal on Computing* 26-1, 1997, pp. 1-14.
4. N. Alon, J.H. Spencer and P. Erdős, *The Probabilistic Method*, Wiley, 1992,
5. H. Buhrman, M. Li, J. Tromp and P.M.B. Vitányi, Kolmogorov Random Graphs and the Incompressibility Method, *SIAM J. Comput*, To appear.
6. G.J. Chaitin, On the number of N-bit strings with maximum complexity, *Applied Mathematics and Computation*, 59(1993), 97–100.
7. D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Proc. of 19th ACM Symp. on Theory of Computing*, 1987, pp. 1-6.
8. C. Domingo, O. Watanabe, T. Yamazaki, A role of constraint in self-organization, *Proc. Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM98)*, LNCS, Springer-Verlag, Heidelberg, 1998, To appear.
9. P. Erdös and J.H. Spencer, *Probabilistic Methods in Combinatorics*, Academic Press, 1974.
10. W. Feller, *An Introduction to Probability Theory and Its Applications, Vols. 1 and 2*, Second Edition, Wiley, 1957.
11. D.H. Greene and D.E. Knuth, *Mathematics for the Analysis of Algorithms*, 3rd ed., Birkhäuser, Boston, MA, 1990.
12. T. Jiang, M. Li, P. Vitányi, Average-Case Complexity of Shellsort, in *these Proceedings.*
13. M. Li and P.M.B. Vitányi, *An Introduction to Kolmogorov Complexity and its Applications*, Springer-Verlag, New York, 2nd Edition, 1997.
14. M. Li and P.M.B. Vitányi, Kolmogorov complexity arguments in combinatorics, *J. Comb. Th., Series A*, 66:2(1994), 226-236. Printing Error, *Ibid.*, 69(1995), 183.
15. A.I. Khinchine, *Fundamenta Mathematicae*, 6(1924), 9–20.
16. A.N. Kolmogorov, Three approaches to the quantitative definition of information. *Problems Inform. Transmission*, 1(1):1-7, 1965.
17. E. Kushilevitz, N. Nisan, *Communication Complexity*, Cambridge Univ. Press, 1997
18. R. Niedermeier, K. Reinhardt, and P. Sanders, Towards optimal locality in Mesh-Indexings. *Proc. FCT'97*, LNCS Vol 1279, Springer-Verlag, 1997, pp. 364-375.
19. P. O'Neil and E. O'Neil. A fast expected time algorithm for boolean matrix multiplication and transitive closure. *Information and Control* 22, 1973, pp. 132-138.
20. N. Nisan, *Personal communication*, 30 Dec. 1998: "I'm not sure where this exercise was proved nor do I remember the details of the proof, but the idea was to take into account the sizes of rectangles when counting the discrepancy on the rectangle."
21. P. Révész, *Random Walk in Random and Non-Random Environments*, World Scientific, Singapore, 1990.
22. M.E. Saks and M. Werman, On computing majority by comparisons, *Combinatorica* 11, 1991, pp. 383-387.