String Recognition on Anonymous Rings

Evangelos Kranakis[1,2] Danny Krizanc[1,2] Flaminia L. Luccio[1,2]

[1] Carleton University, School of Computer Science, Ottawa, ON, Canada. K1S 5B6

[2] Research supported in part by National Sciences Engineering Research Council of Canada grant.

Abstract. We consider the problem of recognizing whether a given binary string of length n is equal (up to rotation) to the input of an anonymous oriented ring of n processors. Previous algorithms for this problem have been "global" and do not take into account "local" patterns occurring in the string. Such patterns may be repetitive or discriminating, and can be used to provide efficient algorithms for recognizing strings. In this paper we give new upper and lower bounds on the bit complexity of string recognition. For the case of periodic strings, near optimal bounds are given which depend on the period of the string. For the case of a randomly chosen string, an optimal algorithm for the problem is given. In particular, we show that almost all strings can be recognized by communicating $\Theta(n \log n)$ bits. It is interesting to note that Kolmogorov complexity theory is used in the proof of our *upper* bound, rather than its traditional application to the proof of lower bounds.

1 Introduction

There have been several studies in the literature concerning the construction of communication efficient algorithms for computing functions on anonymous rings [2, 4, 5, 6, 1, 10], as well as on more general anonymous networks, like tori [3], hypercubes [7], Cayley networks [8], etc. In general, studies of the bit complexity of computing boolean functions of the inputs mainly resort to input collection before determining the output of the boolean function.

In particular, Attiya, Snir and Warmuth [2] showed that all the functions that are computable on an anonymous asynchronous ring of processors, can be computed using $O(n^2)$ messages (one bit message if the functions are boolean) using input collection. They also proved that every algorithm that computes the minimum of all inputs (the OR function in the boolean case), requires $\Omega(n^2)$ messages, i.e., the bounds in this case are tight. Furthermore, they show that almost all boolean functions on n variables require $\Omega(n^2)$ bits to compute. On the other hand, Moran and Warmuth [10] gave an $\Omega(n \log n)$ lower bound for computing any non-constant function and presented functions (derived from a recursive application of de Bruijn sequences) with a $O(n \log n)$ bit complexity.

In this paper we further investigate the bit complexity of computing boolean functions on an anonymous ring by concentrating on an interesting class of functions related to the problem of recognizing an input as being equivalent up to rotation to a fixed string. These functions are shown to exhibit behavior opposite to what was described above, i.e., almost all such functions can be computed using $O(n \log n)$ bits and functions requiring $\Omega(n^2)$ bits are the exception, not the rule. The bit complexity of string recognition is shown to be closely related to the underlying symmetry of the string.

1.1 Results of the paper

Let $x = x_1 x_2 \cdots x_n, y = y_1 y_2 \cdots y_n \in \{0,1\}^n$ be binary strings of length n and suppose that for each $i = 1, \ldots, n$ the i-th processor of an anonymous ring knows the i-th bit y_i of y as well as the entire string x. Give an efficient algorithm such that all n processors recognize whether or not the input string y is identical (up to rotation) to the string x is known as the **string recognition problem**.

We can reformulate our problem in terms of boolean functions. For any string $x \in \{0,1\}^n$ define the boolean function $f_x : \{0,1\}^n \to \{0,1\}$ as follows:

$$f_x(y) = \begin{cases} 1 \text{ if } y \text{ is a cyclic shift of } x \\ 0 \text{ otherwise.} \end{cases}$$

Clearly, being invariant under rotations, the function f_x is computable in the oriented anonymous ring [2]. We are interested in determining the bit complexity of computing $f_x(y)$.

The input collection algorithm of Attiya, Snir and Warmuth [2] implies that for every x, $f_x(y)$ can be computed using $O(n^2)$ bits. Such an algorithm is "global", in the sense that the same algorithm is executed regardless of x or the input to the network. In the sequel, we show how taking into account repetitive or discriminating patterns within a string might provide more efficient algorithms for our problem. For the case of symmetric or repetitive strings we specify upper bounds (section 2.1) and lower bounds (section 3) which depend on the period k of a string x of length n, where $x = w^t v$, $t \geq 0$, v a prefix of w, and $k = |w|$. If $v = \emptyset$ then we give an $O(nk + \frac{n^2}{k})$ upper bound and $\Omega(n \log n + \frac{n^2}{k})$ lower bound. If $v \neq \emptyset$ then we give an $O(nk + n \log n)$ upper bound and $\Omega(n \log n)$ lower bound.

In section 2.2, we show that almost all strings can be recognized using $\Theta(n \log n)$ bits. Our upper bound is an interesting application of Kolmogorov complexity theory which has traditionally been used to show lower bounds. From an intuitive point of view, Kolmogorov complexity theory deals with the amount of information that a string contains [9, 12]. This quantity can be calculated in terms of the size of the shortest program that computes the string. For a random string x of length n, the length of its shortest program is very close to n, with high probability. A string with this property is called incompressible. We use this fact to derive an upper bound of $O(n \log n)$ bits for almost all strings, by showing that if no efficient algorithm exists to recognize a string then a short program exists to compute the string, contradicting its incompressibility.

1.2 Preliminaries

As our model of computation we consider the standard anonymous, asynchronous, ring of n processors [2]. The size n of the ring is known to the processors, and initially the processors are given a single bit as well as a string x of length n. By anonymity, the processors execute the same algorithm given the same data. At the end, it is desired that all the processors determine correctly whether or not the given string is a rotation of the input.

In the proofs below we further assume that the ring is oriented, i.e., the processors can distinguish in a "globally consistent" manner their left from their right neighbor. Nevertheless, all our results are valid for unoriented rings with only minor modifications in the definitions and the statements of the results.

Let n be the number of processors and let $0 \leq i < n$ denote the i-th processor. If b_p is the input to processor p then $\langle b_p : p < n \rangle$ denotes the network input. The right (respectively, left) neighbourhood of processor p of length k is the sequence of bits $\langle b_{p+i} : 0 \leq i < k \rangle$ (respectively, $\langle b_{p-i} : 0 \leq i < k \rangle$), where the operations $+, -$ are done modulo n.

Before we proceed with the description of our algorithms we give some necessary definitions.

Definition 1. Call u (respectively, v) a *prefix* (respectively, *suffix*) of the string x, if $x = uv$ for some string v (respectively, u); the prefix (respectively, suffix) is called *proper* if it is not equal to x. For any string x of length n let the *period* of x be the smallest integer $k \leq n$ such that for some strings w, v we have that: w has length k, v is a prefix of w, and $x = w^{\lfloor n/k \rfloor} v$. For any string u, let $|u|$ denote its length. Given a string $x = x_1 x_2 \ldots x_n$ of length n, let $x \uparrow s = x_1 x_2 \ldots x_s$ denote the first s bits of x, and $x^r = x_{r+1} x_{r+2} \ldots x_n x_1 x_2 \ldots x_r$ the r-th rotation of x. S_x^t is the set of all the different substrings of x, and of all its rotations, that are of length $t \leq n$, i.e., $S_x^t = \{x^i \uparrow t \mid i = 1, \ldots, n\}$.

Definition 2. Given a string x of length n, a program p of length $|p|$, a programming language S, let $K_S(x)$ denote the length of the minimum program that outputs x using S, i.e., $K_S(x) = \min\{|p| : S(p) = x\}$.

From now on we will use the notation $K(x)$ to refer to the Kolmogorov complexity of x with respect to an optimal method of specification S, i.e., a method whose complexity differs by at most a constant from other methods that compute x. If $K(x) \geq |x| - O(\log |x|)$ we say that the string x is *Kolmogorov random*. It is easy to derive the fact that almost every string of a given length is Kolmogorov random. To simplify our presentation we assume from now on that the string x is Kolmogorov random if $K(x) \geq |x| - \log |x|$. In particular, this implies that a string of length n is Kolmogorov random with probability at least $1 - 1/n$.

2 Upper Bounds

2.1 Periodic strings

We now give an efficient algorithm for the string recognition problem. We can prove the following theorem.

Theorem 3. *Let $x = w^{\lfloor n/k \rfloor} v$ be a string of period k such that $|x| = n, |w| = k$ and v is a proper prefix of w. There is an algorithm for recognizing the string x on an n processor anonymous oriented ring with bit complexity $O(nk + \frac{n^2}{k})$ if $v = \emptyset$, and $O(nk + n \log n)$ if $v \neq \emptyset$.*

Proof The main proof of the theorem consists of an efficient input collection algorithm satisfying the requirements above. Let a string x be known to all the processors and let b_p be the input to processor p. In the output of the algorithm it is determined whether or not x is a rotation of the input. Each processor executes the following algorithm which is described in several phases.

Phase 1 The processor computes the period k of the given string x as well as strings w, v such that v is a proper substring of w and in addition $x = w^t v$, for some integer $t \geq 0$. This is a local step executed initially by all the processors and does not contribute in any way to the overall bit complexity of the final algorithm.

Phase 2 Let k be the period of the string x as computed in Phase 1. The processors now compute their left neighborhood of length k as well as their right neighborhood of length $2k + 1$ by executing the following algorithm.

/* computing the right neighborhood */ send b_p to the left;
for $i = 1$ to $2k$ **do** send bit received from the right to the left **od**
/* computing the left neighborhood */ send b_p to the right;
for $i = 1$ to $k - 1$ **do** send bit received from the left to the right **od**

The idea of this phase is to compute a *small* length neighborhood of the input string and use this to decide whether or not the input string is a rotation of x. It will become apparent in the case $v \neq \emptyset$ (discussed in the sequel) why we need to compute a neighborhood of length $2k + 1$ to the right and a neighborhood of length k to the left (for the time being we will only have use for the left neighborhood of length k). Clearly, the bit complexity of Phase 2 is $O(nk)$.

Phase 3 In this phase the processors will execute the main part of the algorithm that will determine whether or not the given string is a rotation of the input. The rest of the algorithm depends on whether or not $v = \emptyset$.

CASE $v = \emptyset$: Let u_p be the first k bits of the right neighborhood computed by processor p in Phase 2 above. Now p executes the following algorithm.

if u_p is not a rotation of w
  then broadcast "no" x is not a rotation of the input;
  else /* u_p is a rotation of w */
  if u_p is the lexicographically maximal rotation of w
    then send a "yes" around the ring;
    every processor that receives a "yes" forwards it around the ring;
    every processor that receives a "no" forwards it around the ring and halts;
  if u_p is the lexicographically maximal rotation of w and processor p receives
    $\lfloor \frac{n}{k} \rfloor$ "yes" messages **then** broadcast x is a rotation of the input;
    else /* it receives a "no" */ broadcast x is not a rotation of the input; **fi**
  fi

At the confirmation stage at most $\lfloor \frac{n}{k} \rfloor$ processors participate (i.e., the ones that have a lexicographically maximal rotation of w); each processor sends a "yes" or a "no" message around the ring (confirming that it has a lexicographically maximal rotation of w) and waits for $\lfloor \frac{n}{k} \rfloor$ "yes" messages or a "no" message. At most $\lfloor \frac{n}{k} \rfloor$ "yes" messages are sent and travel n steps. Finally, if any processor either has a neighborhood of length k which is not a rotation of w or receives a "no" message, it sends it to its neighbor and does not transmit again. Thus the overall bit complexity of this phase is $O(\frac{n^2}{k})$. (The "yes" or "no" messages require only $O(1)$ bits.)

At the end of this phase all processors will know whether or not all processors have seen a rotation of w. If indeed, all processors have seen a rotation of w then the string x is accepted else the string is rejected. The correctness of this last assertion is proved in the sequel. In the following argument we assume that the strings x, w, v are as in the statement of Theorem 3.

If there exists a processor $p \in \{i = 1, \ldots, n\}$ that does not see a rotation of the input, this obviously implies $x \neq w^t$, and p will broadcast a message to stop the other processors.

If all the processors have seen a rotation of the input, then $x = w^t$ must hold. Let us assume on the contrary that there exist a processor p whose left neighborhood $w^{(t-1)} = w_t \ldots w_k w_1 \ldots w_{t-1}$, $t = 1, \ldots, k$ is a rotation of the input, but its neighbor $p+1$ sees another rotation $w^{(s-1)} = w_s \ldots w_k w_1 \ldots w_{s-1}$, $s = 1, \ldots, n$, which is not adjacent. Since p and $p+1$ are adjacent, their left neighborhood must have $k-2$ bits that are the same, i.e., $w_{(t+1) \bmod k} \ldots w_k w_1 \ldots w_{t-1} = w_s \ldots w_k w_1 \ldots w_{s-2}$. Moreover, if the two rotations cannot be adjacent, we have $w_t \neq w_{s-1}$, and since we are working with binary strings, we have $w_t = \overline{w_{s-1}}$. We now need the following:

Lemma 4. *Every rotation $w^{(t-1)} = w_t \ldots w_k w_1 \ldots w_{t-1}$, $t = 1, \ldots, k$, of the binary string $w^{(0)} = w_1 \ldots w_k$ of length k has the same number of ones.*

PROOF Proof omitted. ■

We have assumed that there can be two rotations with a different number of ones since $w_t \neq w_{s-1}$, but the above lemma shows that this is not possible, i.e., we have a contradiction. If each processor sees a rotation of the input, then the string can only be $x = w^t$.

CASE $v \neq \emptyset$: We assume that $x = w^t v$, where $v = v_1 \ldots v_l$ is a proper prefix of w of length l, i.e., $v = w_1 \ldots w_l$. Using the fact that w is the period of the string x, and assuming $x = x_1 \ldots x_n$, we can easily prove the following lemma.

Lemma 5. *There exists a substring $z = z_1 \ldots z_k$ of length k of x that has weight (i.e., number of bits equal to 1) different from the weight of w.*

PROOF Omitted. ■

Let v_p be the neighborhood of length $3k$ computed by processor p in Phase 2. It consists of the left neighborhood of length $2k+1$ and the right neighborhood of length k. In view of the representation of x in the form $w^t v$ it is an immediate consequence of Lemma 5 that there is a *unique* position in the string x whose

k-neighborhood is not a rotation of w, but such that the k-neighborhoods of all k positions to its left are indeed rotations of w and the k positions to its right have the *correct* values.

However, the converse of this is also true, in the sense that if the input string has a unique position satisfying the above conditions, as well as the k positions to its right have the correct k-neighborhoods, and in addition all other positions have a k neighborhood which is a rotation of w then in fact the input string itself must be a rotation of x. In the proof of the following lemma we assume that $x = w^{\lfloor \frac{n}{k} \rfloor}v$, $\lfloor \frac{n}{k} \rfloor \geq 2$. The case $x = wv$ can be solved using input collection with $O(n^2)$ bit complexity.

Lemma 6. *If after Phase 2 of the algorithm there exists at least one processor whose $3k$-bit string obtained in Phase 2 is different from $3k$ bits of the string w^3vw^3, then $x \neq w^{\lfloor \frac{n}{k} \rfloor}v$. If on the other hand all the processors see a correct $3k$-neighborhood, then $x = w^{\lfloor \frac{n}{k} \rfloor}v$, or $x = w^{s_1}vw^{s_2}v \dots w^{s_i}v$, for $s_j \geq 2$, $j = 1, \dots i$, $i \geq 2$.*

PROOF Proof omitted. ∎

It is now easy to convert this characterization into an $O(nk + n \log n)$ algorithm for determining whether or not the input is a rotation of x. The algorithm for processor p is as follows.

if a proper rotation is not seen **or** a halt is received
  then broadcast x is not a rotation of the input and halt
  else if in the "unique" position
        then send a message around the ring with a counter a **fi**;
    counter a is incremented by every processor;
    if in the "unique" position, and a counter $a < n$ is received,
    then broadcasts x is not a rotation of the input and halt
    else if $a = n$, **then** broadcast x is a rotation of the input; **fi**
    fi
fi

The observations made above prove the correctness of the algorithm. Only one processor can be in the unique position. If there is more then one, this situation is detected by another processor in a unique position, by receiving $a < n$. Note that it is not possible that every processor has w^3 as a neighborhood ($|x| = n$ cannot be divided by $|w|$, since $|v| < |w|$). The bit complexity of the algorithm is straightforward. In the worst case in this step, a message of at most $\log n$ bits travels for n steps. The total bit complexity of the algorithm, in the case $v \neq 0$ is then $O(nk + n \log n)$. This completes the proof of Theorem 3. ∎

2.2 Kolmogorov random strings

We now present an algorithm that optimally recognizes a Kolmogorov random string (i.e., almost every string) using $\Theta(n \log n)$ bits. (The lower bound follows from [10].)

Theorem 7. *Let x be a Kolmogorov random string of length n. There is an algorithm for recognizing the string x on an n processor anonymous oriented ring with bit complexity $O(n \log n)$.*

PROOF Let x be a Kolmogorov random string of length n, given as an input to all the processors, and let b_p be the single bit of input at processor p. In the output of the algorithm it is determined whether or not the input bit configuration is a rotation of x. In the algorithm we use a constant a that will be more precisely calculated below. We will refer to Lemma 8 and 9 to justify the fact that, if the $c\lceil \log n \rceil$ neighborhood of a processor coincides with one of the strings contained in $S_x^t = \{x^i \uparrow t \mid i = 1, \ldots, n\}$ (see Definition 1), then, for n large enough, the input string must be x or one of its rotations.

Let $c = \max\{c', c''\}$, where c' and c'' are constants given in Lemma 8 and 9 below. A generic processor p executes the following phases.

Phase 1 Compute and store S_x^t, for $t = c \log n$. This is a local step and it does not contribute to the overall bit complexity of the final algorithm.

Phase 2 Compute the right neighborhood of length t as Phase 2 of algorithm above.

/*computing the right neighborhood*/ send b_p to the left;
for $i = 1$ to t **do** send bit received from the right to the left; **od**
Check whether the t-neighborhood coincides with one of the strings contained in S_x^t. The cost of this phase, in terms of bits transmitted, is $O(n \log n)$.

Phase 3 In this phase p will determine if the input string is a rotation of x. Since x is a Kolmogorov random string, if after $t-1$ steps all the processors have seen strings of S_x^t, then the input string must be x (see Lemma 9), otherwise, if at least one finds a different substring, it is not x. Let u_p be the t-neighborhood of processor p. Now p executes the following:

if $u_p \notin S_x^t$, **then** broadcast x is not a rotation of the input and halt;
  else if u_p is the maximal string **then** send a counter a around the ring;
    /* counter a is only incremented by processors who have seen an acceptable string */
    if a counter value $a = n$ is received
     then broadcast x is a rotation of the input and halt;
     else /* $a < n$, or another processor has seen an unacceptable string and has sent a halt */ broadcast x is not a rotation of the input and halt **fi**
    else /* u_p is not a maximal string */
    if an unacceptable string is seen
     then broadcast x is not a rotation of the input and halt
     else **if** a counter a is received **then** broadcast $a := a + 1$
      else /*a halt is received*/ store the information "x is (is not) a rotation of the input", broadcast it and halt **fi**
    fi
    fi
  fi

We can now precisely define a method to find the right value of c and we can justify Phase 2 and 3 of the algorithm.

We have the following lemmas:

Lemma 8. *Given a string $x = x_1x_2\ldots x_n$ of length n, and an integer $t \leq n$, there exist a constant $c' \in \mathcal{N}$, such that, if n is large enough, for each $t \geq c' \log n$, if $|S_x^t| < n$, then x is not Kolmogorov random.*

PROOF The proof follows from results reported in [11]. ■

Lemma 9. *There exist a constant $c'' \in \mathcal{N}$, such that, for n sufficiently large and for all $t \geq c'' \log n$, for any two strings $x = x_1x_2\ldots x_n$ and $y = y_1y_2\ldots y_n$ of length n, if x is Kolmogorov random, and $S_y^t \subseteq S_x^t$, then $x = y$ up to rotation.*

PROOF Proof omitted. ■

If the input string is x then from Lemma 8 there exists a unique leader since only one processor sees a maximal substring. Therefore the bit complexity in this case is $O(\sum_{i=0}^{\log n} 2^i(i+1)) + O(n) = O(n \log n)$. If the input string is not x, then there are at most $\frac{n}{c \log n}$ processors that see the maximal string and that send a counter. In this case the bit complexity is still $O(n \log n) + O(n) = O(n \log n)$ since each counter will stop after reaching the next of these processors, i.e., $c \log n$ bit messages will travel for at most n steps. The correctness of the algorithm is straightforward. If the input string is x, then there is a unique leader that acknowledges the fact that each processor has seen an acceptable neighborhood, and Lemma 9 states that this condition is sufficient. On the other hand, if there is more then one leader, or at least one processor finds an unacceptable neighborhood (and from Lemma 9, if x is not the input string then this must happen), then a halt is sent and eventually all the processors will end their computation.

The overall bit complexity of these phases is $O(n \log n)$. This completes the proof of Theorem 7. ■

3 Lower Bounds

In this section we prove lower bounds for the string recognition problem. In the case of Kolmogorov random string we need only the lower bound of $\Omega(n \log n)$ of [10] and the equivalent boolean function formulation of the string recognition problem stated in the introduction. This lower bound implies the optimality of the algorithm given above. For the case of $x = w^t v$ we have the following;

Theorem 10. *Let $x = w^{\lfloor \frac{n}{k} \rfloor} v$ be a string of period k such that $|x| = n$, $|w| = k$ and v is a proper prefix of w. The lower bound for recognizing any n-ary string on an n processors anonymous oriented ring is $\Omega(n \log n + \frac{n^2}{k})$ if $v = \emptyset$, $\Omega(n \log n)$ if $v \neq \emptyset$.*

PROOF Let x be a string as in the statement of the theorem. The $\Omega(n \log n)$ bound follows from [10]. Hence we only need to prove the $\Omega(\frac{n^2}{k})$ lower bound for the case $v = \emptyset$.

By Theorem 5.1 in [2] we know that a lower bound on the bit complexity of computing a function can be derived as follows. Given two input configurations y and z of length n, every algorithm that computes the function on input y requires at least $\sum_{i=0}^{\alpha} SI(y,i)$ messages in the worst case. Here α is a constant, such that there exist two processors p and q in the ring, for which p has the same α-neighborhood in y as q in z, but their output (at the end of the algorithm) is different. $SI(y,i)$ is the symmetry index function of the string y, i.e., the minimum number of elements in a equivalent i-neighborhood for processors in the configuration y.

Consider the string recognition problem with an input string $x = w^{\lfloor \frac{n}{k} \rfloor}$, $v = \emptyset$ and run the algorithm with input configurations $z = w^{\lfloor \frac{n}{k} \rfloor - 1} w_0 \ldots w_{k-1} \overline{w_k}$, and $y = x$. There exist two processors p in y and q in z, that have the same $\lfloor \frac{n}{4} \rfloor - 1$-neighborhood, but their output for the problem is different since $y = x$, and $z \neq x$. On input y there are at least $\frac{\lfloor \frac{n}{k} \rfloor}{2}$ processors who see the same $\lfloor \frac{n}{4} \rfloor - 1$-neighborhood, we have then:

$$\sum_{i=0}^{\alpha} SI(y,i) = \sum_{i=0}^{\lfloor \frac{n}{4} \rfloor - 1} \frac{\lfloor \frac{n}{k} \rfloor}{2} = \lfloor \frac{n}{4} \rfloor \frac{\lfloor \frac{n}{k} \rfloor}{2}, \tag{1}$$

i.e., the lower bound is $\Omega(\frac{n^2}{k})$. ■

4 Conclusions and open problems

In this paper we have studied the string recognition problem on anonymous, asynchronous, n-processor rings.

We have shown a $\Theta(n \log n)$ bit complexity bound for Kolmogorov random strings, which in turns implies this same bit complexity for almost all strings. For strings of the form $x = w^t v$, $t \geq 0$, $|w| = k$, we have shown an $O(nk + n \log n)$ bit upper bound when $v \neq \emptyset$ and an $O(nk + \frac{n^2}{k})$ bit upper bound when $v = \emptyset$. When $v = \emptyset$ we have also shown an $\Omega(n \log n + \frac{n^2}{k})$ bit lower bound which shows that the upper bound is tight, for $k = O(\sqrt{n})$. When $v \neq \emptyset$ the upper bound $O(nk + n \log n)$ is tight for $k = O(\log n)$. An interesting open problem would be tighten these bounds in all cases.

As a final remark, we observe that the proof given for the string recognition problem when x is a Kolmogorov random string can be modified and extended to other networks like tori, meshes, etc.

Acknowledgments

The authors would like to thank Paul Vitányi for pointing out the usefulness of reference [11] in proving lemma 9. Thanks also to Nicola Santoro for the idea on how to tighten the bounds in the case of periodic strings with $v = \emptyset$.

References

1. H. Attiya and Y. Mansour, "Language Complexity on the Synchronous Anonymous Ring", *Theoretical Computer Science*, **53** 169-185, (1987).
2. H. Attiya, M. Snir and M. Warmuth, "Computing on an Anonymous Ring", *Journal of the ACM*, **35 (4)**, 845-875, (1988).
3. P.W. Beame and H.L. Bodlaender, "Distributed Computing on Transitive Networks: The Torus", *6th Annual Symposium on Theoretical Aspects of Computer Science, STACS*, Springer Verlag LNCS,**349**, 294-303, (1989).
4. H.L. Bodlaender, S. Moran, M.K. Warmuth, "The Distributed Bit Complexity of the Ring: from the Anonymous to the Non-Anonymous Case", *Information and Computation*, **108 (1)**, 34-50, (1994).
5. P. Ferragina, A. Monti and A. Roncato, "Trade-off between Computation Power and Common Knowledge in Anonymous Rings", *Pre-Proceedings of Colloquium on Structural Information and Communication Complexity*, Ottawa, Canada, May 16-18 (1994).
6. O. Goldreich, L. Shrara, "On the Complexity of Global Computation in the Presence of Link Failures: the Case of Ring Configuration", *Distributed Computing*, **5, (3)**, 121-131, (1991).
7. E. Kranakis and D. Krizanc, "Distributed Computing on Anonymous Hypercube Networks", *Proceedings of the 3rd IEEE Symposium on Parallel and Distributed Processing*, Dallas, Dec. 2-5, 722-729, (1991).
8. E. Kranakis and D. Krizanc, "Computing Boolean Functions on Cayley Networks", *Proceedings of the 4th IEEE Symposium on Parallel and Distributed Processing*, Arlington, Texas, Dec. 1-4, 222-229, (1992).
9. M. Li, P.M.B. Vitanyi, "Kolmogorov Complexity and its Applications", Handbook of Theoretical Computer Science, Algorithms and Complexity. The MIT Press, Cambridge, Massachusetts, (1990).
10. S. Moran and M. K. Warmuth, "Gap Theorems for Distributed Computation", *Proceedings of the 5th ACM Symposium on Principles of Distributed Computing*, 131-140, (1986).
11. J. Seiferas, "A Simplified Lower Bound for Context-free Language Recognition," Information and Control, **69**, 255-260, (1986).
12. A.K. Zvonkin, L.A. Levin, "The Complexity of Finite Objects and the Development of the Concepts of Information and Randomness by means of the Theory of Algorithms", *Russian Math. Surveys*, **25**, 83-124, (1970).