ON THE BIT COMPLEXITY OF DISTRIBUTED COMPUTATIONS 
IN A UNIDIRECTIONAL RING WITH A LEADER

by

Y. Mansour and S. Zaks

Technical Report #383 
November 1985
ABSTRACT

We study the bit complexity of pattern recognition in a distributed unidirectional ring with a leader. Each processor gets as input a letter from some alphabet, and these concatenated letters, starting at the leader, form the pattern of the ring. The leader initiates an algorithm that accepts or rejects this pattern. Thus each algorithm recognizes a language over a given alphabet. We prove the following (n is the size of the ring):

1. A language is recognized by an algorithm that uses $O(n)$ bits if and only if it is regular (also compared is the bit complexity vs. the number of rounds).

2. Every non-regular language requires at least $\Omega(n \log n)$ bits for its recognition (clearly, every language requires no more than $O(n^2)$ bits for its recognition).

3. For every function $g(n)$, $\Omega(n \log n) \leq g(n) \leq O(n^2)$, there is a language that requires $\Theta(g(n))$ bits for its recognition.

1. INTRODUCTION

Much research is recently concentrated on distributed computations, in attempts to understand their complexity. In these studies, a network of processors is given, and by exchanging messages they have to solve certain problems. The complexity measures studied are the message complexity and the bit complexity. A widely studied problem is identifying a unique processor, usually referred to as finding a leader. Alas, even the existence of a leader still leaves many open questions regarding the communication complexity of certain problems, and this study belongs to this category.

We study the bit complexity of distributed pattern recognition in a unidirectional ring with a leader. Each processor gets as input a letter, and the leader initiates an algorithm that accepts or rejects the pattern formed by the concatenation of these letters (starting at the leader). Thus, we can view an algorithm as computing a function or recognizing a language over a given alphabet. We study the communication complexity of this recognition problem.
Assuming a ring of size $n$, it is clear that every function can be computed in $n$ messages (the leader collects the information from all the processors and then computes the function locally), which is clearly also a lower bound. For that reason we consider bit complexity rather than message complexity. The leader can obtain all the information about all the processors in $O(n^2)$ bits, giving a trivial upper bound for the computation of every function, while $\Omega(n)$ bits is clearly a lower bound. In this study we shed some light on this $\Theta(n) - \Theta(n^2)$ range.

We show that a language is recognized by an algorithm that uses $O(n)$ bits if and only if it is regular ($n$ is the size of the ring); and that every non-regular language requires at least $\Omega(n \log n)$ bits for its recognition. In other words, for every function $g(n)$, $\Omega(n) < g(n) < O(n \log n)$, there is no language that requires $\Theta(g(n))$ bits for its recognition. The hierarchy of the non-regular languages, in terms of their bit complexity, is not the natural one; we show two examples: a linear language that requires $\Omega(n^2)$ bits, and a context-sensitive language that is not context-free and can be recognized in $O(n \log n)$ bits. We show that for every function $g(n)$, $\Omega(n \log n) < g(n) < O(n^2)$, there is a language that requires $\Theta(g(n))$ bit for its recognition. We end this study by discussing a trade-off between the bit complexity and the number of rounds needed for the recognition of regular languages, and show, for every $c$, a language that is recognized in one pass in $c n$ bits and in $\log c$ passes in $n \log c$ bits.

The problem of distributively finding a leader in a unidirectional ring with distinct identities appears in [DKR,P], and algorithms that use at most $O(n \log n)$ messages are shown; this bound is best possible ([PKR]). In [IR,ASW] it is proved that in the case when the identities are not distinct and the number of processors is unknown, there is no algorithm that can compute the ring size. Therefore, in order to be able to compute any function, we either must know the size of the ring or have a leader. The case where only the size of the ring is known is the main subject of [ASW] where - even though a leader cannot be found - functions concerning the pattern of the ring can still be computed, and results about the cost of this computation are discussed. Even the existence of a leader still leaves many open questions regarding the communication complexity...
of certain problems, and various studies in distributed graph algorithms make this assumption (see, for example, [F], [KRS] or [MS]). Another approach is discussed in [IR], where probability is used in order to break the symmetry (=find a leader).

A similar result for Turing machines computations is presented in [HA], [HE] and [T]. It is shown that a one-tape Turing-machine recognizes a language in time $O(n)$ if and only if the language is regular, and that a non-regular language requires at least $\Omega(n \log n)$ time. It will be argued in the summary section that the results for these two models do not seem to imply each other.

In Section 2 we introduce the model and the notations. In Section 3 we prove the results for regular languages. Section 4 deals with the lower bound for non-regular languages. Section 5 contains the examples mentioned above, about the bit complexity hierarchy, and trade-off between time and bit complexity. Summary and open problems are found in Section 6.

2. THE MODEL

The model under discussion is a distributed unidirectional ring of processors (see [DKR]), with one specific processor (the leader). The number of processors is not known to any of the processors. Each processor holds one letter from a given alphabet. The processors can communicate only through the edges of the ring, and each message is assumed to have a finite transmission time. All the processors, excluding the leader, execute the same algorithm.

We deal with algorithms that recognize the pattern of the ring. We assume that the leader initiates the algorithm. Therefore, the execution of each algorithm is unique, and can be described as a sequence of messages sent by the processors around the ring in a round-robin fashion, starting with the leader. We choose to terminate the algorithm when the leader accepts or rejects the pattern of the ring; in other words, the leader wants to know whether the word on the ring belongs to a given language (clearly, the leader can then inform all the other processors that the algorithm ter-
minated). Note that this implies that each processor sends and receives the same number of messages. We do not deal with languages that can be recognized without exchanging any messages.

We use the following notations and definitions:

\( n \) The size of the ring.

\( \Sigma \) The (finite) alphabet.

\( \sigma \) An element of \( \Sigma \) (a letter).

\( p_i \) The \( i \)-th processor. \( p_1 \) is the leader, \( p_i \), \( i > 1 \), is the processor that receives messages from \( p_{i-1} \), and \( p_1 \) receives messages from \( p_n \).

\( \sigma_i \) The initial value of \( p_i \).

\( w \) A word in \( \Sigma^* \).

We say that the ring is labeled with \( w \) if \( w = \sigma_1 \sigma_2 \ldots \sigma_n \).

\( L \) A language over \( \Sigma \).

\( A \) An algorithm.

Two algorithms are equivalent if they recognize the same language.

\( \text{pass} \) A sequence of \( n \) messages, the first of which sent by the leader.

\( \pi_A \) The total number of messages sent (or received) by each processor (e.g. the number of passes), during the execution of the algorithm \( A \).

\( M_A \) The set of messages used by the algorithm \( A \) (can be infinite).

\( \text{BIT}_A(n) \) The bit complexity of the algorithm \( A \). This means that for a ring labeled with \( w \), \( |w| = n \), \( A \) uses at most \( \text{BIT}_A(n) \) bits.

\( \text{BIT}_L(n) \) The bit complexity of a language \( L \). This means that

1. \( \text{BIT}_A(n) = \Omega(\text{BIT}_L(n)) \) for every algorithm \( A \) that recognizes \( L \), and
2. There exists an algorithm \( A \) that recognizes \( L \) such that \( \text{BIT}_A(n) = O(\text{BIT}_L(n)) \).

Clearly \( \Omega(n) \leq \text{BIT}_L(n) \leq O(n^2) \) for every language \( L \) that cannot be recognized without sending any messages (for example: \( \Sigma^* \), \( \emptyset \), \( \sigma \Sigma^* \)).
3. REGULAR LANGUAGES

In this section we show that the bit complexity $\text{BIT}_L(n)$ of a language $L$ satisfies $\text{BIT}_L(n) = \Theta(n)$ if and only if $L$ is a regular language. We first show that the condition is necessary.

**Theorem 1:** Let $L$ be a regular language. Then $\text{BIT}_L(n) = O(n)$.

**Proof:** Let $FA = (Q, \Sigma, \delta, q_0, F)$ be a finite automaton that recognizes the language $L$. We construct an algorithm $A$ that recognizes $L$ in one pass: initially, each processor in the ring will have a copy of $FA$, and the message that processor $p_i$ will send will contain the state of $FA$ after scanning $i$ letters, $i=1,2,\ldots,n$. This is done as follows: suppose the ring is labeled with $w = \sigma_1 \cdots \sigma_n$; then $p_1$ (the leader) sends to $p_2$ the message containing $q_1 = \delta(q_0, \sigma_1)$, and in general $p_i$ sends a message containing $q_i = \delta(q_{i-1}, \sigma_i)$, $i = 1, \ldots, n$. Clearly $q_n = \delta(q_0, w)$, so when $p_1$ receives this message it can decide whether $w \in L$.

Clearly, each of the $n$ messages require no more than $\lceil \log |Q| \rceil$ bits, hence $\text{BIT}_A(n) \leq \lceil \log |Q| \rceil n = O(n)$. Since each algorithm that recognizes $L$ requires at least $n$ bits, it follows that $\text{BIT}_L(n) = O(n)$.

[ ]

We now show that the condition is sufficient. We first show it for a one pass algorithm:

**Theorem 2:** Let $L$ be a language that is recognized by a one pass algorithm $A$, with $\text{BIT}_A(n) = O(n)$. Then $L$ is regular.

**Proof:** Let $A$ be a one pass algorithm that recognizes the language $L$. $\text{BIT}_A = O(n)$, and suppose $A$ is using the (possibly infinite) set $M_A = \{m_1, \ldots, m_k, \ldots\}$ of messages. We build a directed edge-labeled graph $\hat{G} = (\hat{V}, \hat{E})$ that represents the way the algorithm works, as
follows: $V = \{v_0, v_1, \ldots, v_i, \ldots\}$ where $v_0$ represents a special message that makes the leader initiate the algorithm, and every $v_i \in V$ represents the message $m_i \in M$; $e = (v_i, v_j) \in E$ is a directed edge from $v_i$ to $v_j$ with a label $l(e) = \sigma$ if the following holds: when a processor has the initial value of $\sigma$ and it receives the message $m_i$, it sends the message $m_j$ (here we use the assumption that all the processors execute the same algorithm).

From $\hat{G} = (\hat{V}, \hat{E})$, we construct the subgraph $G = (V, E)$ induced by the vertices reachable from $v_0$. If the graph $G$ is infinite, then by the König's Infinity Lemma (see [E]) we obtain a simple infinite path $v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_i \rightarrow \ldots$. Every prefix of size $n+1$ of the path represents a ring of size $n$ labeled with the word $w = \sigma_1 \ldots \sigma_n$, where $\sigma_i = l(e_i)$. The algorithm $A$ will send on this ring the $n$ distinct messages $m_1, m_2, \ldots, m_n$; at least $\Omega(n)$ of them will require $\Omega(log n)$ bits each, and therefore $BIT_A(n) = \Omega(n \log n)$. This contradicts the assumption that $BIT_A(n) = O(n)$. It follows that the graph $G$ is finite, and it clearly represents a state diagram of a finite automaton that recognizes $L$, hence $L$ is regular.

Actually, from the last part of the proof, the following stronger results hold:

**Corollary 1:** Let $A$ be an algorithm that recognizes a non-regular language in one pass. Then:

(a) $A$ uses an infinite number of messages, and

(b) $BIT_A(n) = \Omega(n \log n)$.

**Corollary 2:**

(a) Let $A$ be a one pass algorithm that uses an infinite number of distinct messages. Then $BIT_A(n) = \Omega(n \log n)$.

(b) Let $A$ be any algorithm. If there is a pass where all the processors, excluding the leader, have no knowledge other than their initial value $\sigma$, and if $A$ uses an infinite
number of distinct messages in this pass, then the bit complexity of that pass is
\( \Omega(n \log n) \) bits, hence \( BIT_A(n) = \Omega(n \log n) \).

We now show the general case.

**Theorem 3:** Let \( L \) be a language such that \( BIT_L(n) = O(n) \). Then \( L \) is regular.

**Proof:** Let \( A \) be an algorithm that recognizes the language \( L \) such that, for some \( c > 0 \)
\( BIT_A(n) < c \cdot n \) for every \( n \), so \( \pi_A < c \) (because the algorithm does not use null messages). We construct an equivalent one pass algorithm \( \tilde{A} \). \( p_1 \) will send a message containing all possible sequences of \( \pi_A \) messages; we later show that \( M_A \) is finite. The processor \( p_i \) when receiving the message simulates its behavior on every sequence of \( \pi_A \) messages it receives, and sends a message containing all the results. When the leader receives a message it can check which of the sequences of \( \pi_A \) messages is the one that it uses during the algorithm, and therefore \( \tilde{A} \) is clearly equivalent to \( A \).

We now show that \( M_A \) is finite. We construct an equivalent algorithm \( \tilde{A} \) that will not need any information about previous messages kept in the processors. \( \tilde{A} \) will perform the first pass like in \( A \), and in the \( i \)-th pass \( p_1 \) will send all its previous \( i-1 \) messages plus the new message of pass \( i \). Each processor will be able to simulate its action in previous passes and send to the next processor all the messages consisting of the \( i-1 \) messages it sent during the previous \( i-1 \) passes and the new message of the \( i \)-th pass.

Clearly, \( BIT_A(n) \leq \pi_A(c \cdot n) < c^2 n = O(n^i) \) bits. \( A \) and \( \tilde{A} \) are equivalent, and both use an infinite number of messages or a finite number of messages. If \( M_A \) is infinite, then there is a pass in \( \tilde{A} \) that can use an infinite number of messages, and by Corollary 2(b) \( BIT_{\tilde{A}}(n) = \Omega(n \log n) \) bits. This contradicts the fact that \( BIT_A(n) = O(n) \). \( M_A \) is therefore finite, which implies that \( M_A \) is finite. We conclude that \( BIT_A(n) \leq n \log^\pi |M| = O(n) \).

We have constructed a one pass algorithm \( \tilde{A} \) with a bit complexity of \( O(n) \). From Theorem 2 we conclude that the language recognized by \( \tilde{A} \) is regular. But \( A \) and \( \tilde{A} \) are equivalent, hence \( L \) is regular.
4. NON-REGULAR LANGUAGES

In this section we show:

**Corollary 3:** Every algorithm that uses a bounded number of passes and an infinite number of distinct messages uses at least $\Omega(n \log n)$ bits.

**Theorem 4:** Let $L$ be a non-regular language. Then $BIT_L(n) = \Omega(n \log n)$

**Proof:** Let $A$ be any algorithm that recognizes $L$. For each processor an *information state* contains its initial value and all the messages that it received in their order. We consider two cases:

Case 1: The number of passes is unbounded.

We define a sequence of words $(w_1, \ldots, w_i, \ldots)$ as follows: the word $w_i$ is a shortest word that requires at least $i$ passes before $A$ terminates, if there is more than one word we chose one that requires the largest number of passes; clearly $|w_i| \leq |w_{i+1}|$. $A$ requires $n_i$ passes for a ring labeled $w_i$, so $|w_i| < |w_{n_i+1}|$. Hence we can construct an infinite sub-sequence of words $(w_1, w_2, \ldots, w_{n_1}, \ldots)$, such that $|w_1| < |w_2| < \cdots < |w_j| < \cdots$. Assume that when $A$ terminates on $w_j = \sigma_1 \cdots \sigma_n$, two processors $p_k$ and $p_l$, $k < l$ are in the same information state. The execution of $A$ on $w_j = \sigma_1 \cdots \sigma_k \sigma_{k+1} \cdots \sigma_n$ will require the same number of passes (at least $i_j$ passes), since no processor will note the difference between these two rings. This contradicts the assumption that $w_j$ is a shortest word requiring at least $i_j$ passes. Therefore, when $A$ terminates, on every word in the sub-sequence each processor is in a distinct information state. In order to encode $n$ distinct information states we need at least $\Omega(n \log n)$ bits, out of which $O(n)$ are the initial values and the rest are bits sent by $A$, hence $BIT_A(n) = \Omega(n \log n)$. 
Case 2 : The number of passes is bounded.

If the number of distinct messages is finite then \( A \) uses only \( O(n) \) bits and by Theorem 3 is a regular language; otherwise, by Corollary 3 we conclude that \( BIT_A(n) = \Omega(n \log n) \).

5. MISCELLANEOUS

We conclude with few notes about the bit complexity hierarchy, and the trade-off between the number of passes vs. the bit complexity for regular languages.

(1) There is an linear language \( L \), such that \( BIT_L(n) = n^2 \).

Let \( L = \{ x \mid x = w^c w, w \in \{ a, b \}^* \} \). It is easy to show that at least \( \Omega(n^2) \) bits must be sent by every algorithm that recognizes \( L \).

(2) There is a context-sensitive language that is not context-free and that requires only \( O(n \log n) \) bits.

Let \( L = \{ 0^n 1^n 2^n \mid n = 1, 2, \ldots \} \). \( L \) can be recognized in \( O(n \log n) \) bits (by sending three counters).

(3) For every function \( g(n) \), \( \Omega(n \log n) < g(n) < O(n^2) \) there exists a language \( L \) such that \( BIT_L(n) = \Theta(g(n)) \).

Let \( \Sigma \) be any alphabet, and \( L = \{ w \mid x, y \in \Sigma^* \text{, } i > 0 \text{, such that } w = x^i y \text{, } |x| > |y| \text{ and } |g(|w|)| = |x| \} \). For every \( i > 0 \), \( \sigma^i \in L \), so for every \( n \) there is at least one word \( w \in L \), such that \( n = |w| \).

Every algorithm that recognizes \( L \) requires at least \( \Omega(|x| \cdot |w|) \) bits. Even if we assume that \( n \) is known to every processor, and even more, every processor knows which bit of \( x \) it holds, still at least \( n - |x| - |y| \) of the processors will send \( |x| \) bits for \( w \in L \) (otherwise, there will be two distinct words of size \( |x| \) for which one of these processors will send the same sequence of bits, and clearly one of these subwords yields a word not in \( L \) that will be accepted, a contradiction). Therefore we have the lower bound of \( \Omega(g(n)) \) bits. The following algorithm recognizes \( L \) in \( O(g(n)) \) bits: The leader determines \( n \) (using \( O(n \log n) \) bits), and then
determines $|x| = \left[ \frac{g(n)}{n} \right]$, and compares every segment of length $|x|$ to the next segment (using $O(|x| \cdot n) = O(g(n))$ bits). Therefore $BIT_L(n) = \Theta(g(n))$

(4) This note concerns the number of bits vs. the number of passes in recognition of regular languages.

We show an example of a language that requires $cn$ in two passes and $2^n$ in one pass, and in the general case, for every $c > 0$, there is a language that is recognized in one pass with a bit complexity of $cn$ and can be recognized in $\log c$ passes with no more than $n \log c$ bits. Let $\Sigma = \{a_0, a_1, \ldots, a_{2^k - 2}\}$. Let $L = \{w \mid w \in \Sigma^* \}$, and $a_{|w| \mod 2^k - 1}$ appears an even number of times in $w$. If we want to recognize $L$ in two passes we can do it with bit complexity of $(2k + 1)n$. In the first round we check $|w| \mod 2^k - 1$ (using $k$ bits), and in the second pass we send the result (using $k$ bits) and an extra bit to see if $a_{|w| \mod 2^k - 1}$ appears an even number of times. In one round we need to check the parity of all the $a_i$'s concurrently, hence we need $(k + 2^k - 1)n$ bits ($k$ for $|w| \mod 2^k - 1$, and one bit for each $a_i$ $0 \leq i \leq 2^k - 2$). It is easy to see from these that in the general case, there are regular languages requiring $cn$ bits for its recognition in one pass, and in $\log c$ passes can be recognized in $n \log c$ bits.

6. SUMMARY AND OPEN PROBLEMS

In this paper we studied the bit complexity of language recognition on a unidirectional ring with a leader. We showed that a language is recognized in $O(n)$ bits if and only if it is regular, and that a non-regular language requires at least $\Omega(n \log n)$ bits. The range of $O(n \log n)$ to $\Theta(n^2)$ still remains unexplained, but it was shown that it does not correspond to the Chomsky hierarchy. From our results it follows that only regular languages can be recognized without the knowledge of $n$, and that for every non-regular language we can assume that $n$ is known.

In the introduction we mentioned similar results ([HA], [HE], [T]) for Turing machines. One might argue that every algorithm $A$ can be transformed in to a Turing machine $TM$, in the following way:
(1) The set of messages $M_A$ will be the set of states of TM.

(2) The transition function is the function that from a given information state and input message creates the output message.

Two main problems arise:

(1) The set $M_A$ can be infinite.

(2) Even if the set $M_A$ is finite, still the transition function can be infinite.

Because of these two problems the transformation of an algorithm $A$ into a Turing machine TM, such that the bit complexity of $A$ equals the time complexity of TM, does not seem to be straightforward. As an example consider an algorithm $A$ that counts, the number of processors in one pass; clearly $A$ uses $O(n \log n)$ bits. Although there exists a TM that performs the same task in $O(n \log n)$ moves, there is no simple way of directly constructing it from the algorithm, so that the bit complexity equals the time complexity. On the other hand, a TM can move in both directions, so a direct simulation of it on a unidirectional ring can increase the bit complexity by a factor of $n$.

We list a few open problems:

(1) Characterize the non-regular languages by their bit complexity (The range $\Theta(n \log n)$ to $\Theta(n^2)$).

(2) How does the knowledge of $n$ affect the results?

(3) Given a regular language $L$, construct an optimal algorithm that recognizes $L$.

(4) Extend the results for bidirectional ring.

(5) Given an algorithm $A$ with an infinite set of messages, does there exist an equivalent algorithm $\hat{A}$ that uses a finite set of messages, and have the same bit complexity?

Acknowledgement: We would like to thank Manfred Warmuth for a helpful discussion.
REFERENCES


