Two Lower Bounds In Asynchronous Distributed Computation

Pavol Duris, Zvi Galil

Technical Report
CUCS-304-87

Two Lower Bounds in Asynchronous Distributed Computation

Pavol Duris

Slovak Academy of Sciences

and

Zvi Galil*

Columbia University

and Tel-Aviv University

* The work of the first author was supported in part by NSF Grants DCR-85-11713 and CCR-86-05353.

Abstract

We introduce new techniques for deriving lower bounds on message complexity in asynchronous distributed computation. These techniques combine the choice of specific patterns of communication delays and crossing-sequence arguments with consideration of the speed of propagation of messages, together with careful counting of messages in different parts of the network. They enable us to prove the following results, settling two open problems:

— An $\Omega(n \log^* n)$ lower bound for the number of messages sent by an asynchronous algorithm for computing **any** non-constant function on a bidirectional ring of n anonymous processors.

— An $\Omega(n \log n)$ lower bound for the **average** number of messages sent by any maximum-finding algorithm on a ring of n processors, in case n is known.

0. Introduction

Consider the following model. We have a bidirectional asynchronous anonymous ring of n processors ([1],[5]). There is no leader among the processors. All processors run the same program, which may depend on the size of the ring. All processors compute the same function $f : \Sigma^n \rightarrow \{0,1\}$, where Σ is an arbitrary finite alphabet. The input of each processor is a letter of Σ, and the processors compute $f(x)$, where x is the concatenation of the n inputs beginning with any processor on the ring. We assume that for every input $x \in \Sigma^n$ and for any possible pattern of communication delays (or scheduling of the messages sent) all the processors eventually stop. Upon termination all processors are in one of two states: either they all *accept* (which corresponds to $f(x) = 1$) or they all *reject* (which corresponds to $f(x) = 0$).

In [1] Attiya, Snir and Warmuth considered the following function: $f(x) = 1$ if x is a cyclic shift of a string in $0(01)^*$ and is 0 otherwise. They showed that if n, the size of the ring, is assumed to be odd, then the function can be computed in $O(n)$ messages. Similar non-constant functions computable in $O(n)$ messages can be defined if the size of the ring is assumed to have any fixed constant non-divisor. They left as an open problem whether a similar result can be obtained **without** restrictions on the size of the ring. In [5] Moran and Warmuth defined a non-constant function and proved that its message complexity is at most $O(n \log^* n)$. Our main result answers the open problem in [1] in the negative and shows that the upper bound in [5] is best possible:

Theorem 1. Let $n = m!$, $m > 1$. If $f : \Sigma^n \to \{0, 1\}$ is a non-constant function, then any asynchronous algorithm for computing f on a bidirectional ring of n anonymous processors requires at least $\Omega(n \log^* n)$ messages in the worst case.

In [5], Moran and Warmuth showed that any non-constant function requires $\Omega(n \log n)$ bits on an anonymous ring of size n, while it is easy to construct non-constant functions with $O(n \log n)$ bit complexity on such a ring. They refer to this phenomenon as a gap in complexity between constant and non-constant functions. (When the function is constant the bit complexity is 0.) They also left open the question whether a gap exists when we consider the message complexity. Their lower bound techniques were not sufficient for establishing the gap which Theorem 1 exhibits. Theorem 1 deals with the more general setting that allows general messages. Moreover the result in [5] did not exclude the possibility of $O(n)$ message complexity of non-constant functions. For example, the algorithm of [1] mentioned above has $O(n)$ message complexity but $\Theta(n \log n)$ bit complexity.

In Section 1 we prove Theorem 1. Our arguments consider the speed of propagation of certain messages as well as crossing sequences (i.e. cut and paste) and specific choices of communication delays to fool the algorithm and derive a contradiction.

Our second main result concerns the problem of maximum-finding on a ring of proces-

sors, which is one of the basic problems in distributed computation. Its solutions are used as building blocks in other more complicated algorithms. It has been studied quite extensively. We consider a ring of n processors $p_1, p_2, \ldots, p_n$, and let $L = \{s_1, s_2, \ldots, s_m\}$ be a set of labels (distinct integers). Assume that for $i = 1, \ldots, n$, p_i is labeled by $r_i \in L$ and every two processors are labeled by distinct labels. We consider asynchronous message-driven algorithms in which all processors start simultaneously, the communication channels are first-in first-out, and all processors eventually stop after computing the maximum label (see [6]).

There are two different versions of the problem, depending on whether or not n is known to the processors. Also, one can consider the worst-case message complexity or the average message complexity. In the latter case we average the message complexity over all possible distinct label assignments to the n processors. For each such assignment we consider the worst pattern of communication delays. Consequently, we have four cases to consider. There is yet another distinction between unidirectional and bidirectional rings, but similar results were obtained for both subcases: usually lower bounds are proved in the bidirectional case and upper bounds for the unidirectional case. This distinction is only important for determining the best constants in the bounds.

$O(n \log n)$ upper bounds for all four cases have been known for some time (see for example [4],[7]). Burns [3] and Pachl, Korach and Rotem [6] proved $\Omega(n \log n)$ lower bounds in all cases but one. Their techniques did not suffice for determining the average message complexity in case n is known, and no nontrivial (nonlinear) lower bound was known. Bodlaender [2] proved an $\Omega(n \log n)$ lower bound for unidirectional algorithms which use only comparisons between labels. Our second result completes the picture:

Theorem 2. If the label set L is sufficiently large, then any maximum-finding algorithm for a bidirectional ring of size n labeled by L in which the processors know n has average message complexity at least $\Omega(n \log n)$.

In Section 2 we prove Theorem 2. We choose two types of specific communication delays: the first lead to contradiction by forcing the algorithm to terminate without the correct answer, and the second force the algorithm to send many messages needed for the desired lower bound. The proofs also use arguments that consider the speed of propagation of certain messages, as well as a special way of counting the messages in different parts of the ring.

1. The Proof of Theorem 1

For convenience we denote the anonymous processors by $p_1, p_2, \ldots, p_n$. (Processor p_i does not know i.) Without loss of generality we assume that the ring R of size $n = m!$ is oriented, i.e. all processors in it agree on the same "right" and "left" directions. Consider an algorithm A which computes a non-constant function f on the ring R (where both are arbitrary but fixed). For input $w = w_1 w_2 \ldots w_n$ we choose a particular pattern of communication delays: All processors start at time zero, internal computations take no time and links are "synchronized"; i.e., it takes one unit of time to traverse the link. Therefore we can speak of time $0, 1, 2, \ldots$ in the computation of A on w. Recall that $n = m!$ and assume first that $m \geq 192$.

In the proof we use the notion of a "segment" of the ring R and the notion of a "crossing sequence" at a link of R. For $1 \leq i < j \leq n$, the *segment* $[i, j]$ consists of the processors $p_i, p_{i+1}, \ldots, p_j$ and all the links associated with them including the first (leftmost) and last (rightmost) links. A segment is a segment $[i, j]$ for some $1 \leq i < j \leq n$. The *length* of a segment $s = [i, j]$ is $|s| = j - i + 1$ and the input to s is the string $w_i w_{i+1} \ldots w_j$.

Let M be the set of all possible messages and let $\epsilon \notin M$ denote the empty message (not counted in the message count). The *crossing sequence* at a link b until time $T \geq 0$ is a pair (r, l), $r, l \in (M \cup \{\epsilon\})^{T+1}$, where for $0 \leq t \leq T$, r_t (l_t) is the message sent right (left)

on b at time t. If $r_t = l_t = \epsilon$ we say that the link b is *passive* at time t in the computation of A on w. Otherwise we say that b is *active*. Note that in every time unit until every processor halts at least one link is active.

The *configuration* of a processor p at time $t \geq 0$ in the computation of A on w is the triple (q, m_L, m_R) where q is the state of the processor p (running A) immediately after performing its internal computation, and m_L (m_R) is the message (if there is any) sent by p to the left (to the right) at time t in the computation of A on w; if there is no such message then m_L (m_R) is ϵ. The configuration of a segment $[i, j]$ at time $t \geq 0$ in the computation of A on w is the $(j - i + 1)$-tuple $(a_i, a_{i+1}, \ldots, a_j)$, where a_l is the configuration of p_l at time t in the computation of A on w for $l = i, i + 1, \ldots, j$.

The proof of Theorem 1 requires some lemmas. Lemma 1 follows by considering the propagation speed of messages.

Lemma 1.

(a) Let i, T, T' be positive integers such that $0 \leq i - T$, $i + T \leq n$ and $T < T'$. If the link connecting p_i and p_{i+1} is passive at every time $t = T, T+1, \ldots, T'$ in the computation of A on w, then the crossing sequence at this link until time T' is uniquely determined only by the string $w_{i-T+1} w_{i-T+2} \ldots w_{i+T}$ and by the algorithm A.

(b) Let $1 \leq i < j \leq n$ and $i + 1 < j - 1$. If all processors of the segment $[i, j]$ are passive at time t in the computation of A on w, then all processors of the segment $[i+1, j-1]$ are passive at time $t + 1$.

Proof. (a) The proof is obvious for $T = 1$. Assume $T \geq 2$. First, observe that the configuration of the segment $[i - T + t + 2, i + T - t - 1]$ at time $t+1$ in the computation of A on w is uniquely determined by the configuration of the segment $[i - T + t + 1, i + T - t]$ at time t and by A for every $t = 0, 1, 2, \ldots, T - 2$. Clearly, all these configurations are determined by the configuration of $[i - T + 1, i + T]$ at time $t = 0$, i.e. by the string $w_{i-T+1} \ldots w_{i+T}$ and by algorithm A. On the other hand, they uniquely determine the

crossing sequence mentioned above until time $T-1$. To complete the proof, note that the link is passive after time $T-1$ and until time T'.

(b) The proof follows from the fact that if three consecutive links are passive at some time in the computation of A of w, then the middle link is passive at the next time in the computation of A on w. □

Corollary 1. Let $1 \leq i < n$. If the link connecting p_i and p_{i+1} is active at time t, then at least one link of the segment $[i-l, i+l+1]$ is active at time $t-l$ for every $l = 0, 1, 2, \ldots, \min\{t, i-1, n-i-1\}$.

Proof. Suppose that Corollary 1 is false and apply part (b) of Lemma 1 to derive a contradiction. □

Corollary 2. Let $1 \leq j_1 < i < j_2 \leq n$. If the link connecting p_i and p_{i+1} is active at time t, then at least one link of the segment $[j_1, j_2]$ is active at time $t-l$ for every $l = 0, 1, 2, \ldots, \min\{t, i-j_1, j_2-i-1\}$.

Proof. Observe that all links of the segment $[i-l, i+l+1]$ are some links of the segment $[j_1, j_2]$ for every $l = 0, 1, 2, \ldots, \min\{t, i-j_1, j_2-i-1\}$ and apply Corollary 1. □

Lemma 2. There is an input $z \in \{0,1\}^n$ such that no processor of the ring accepts or rejects before time $n/4$ in the computation of A on z.

Proof. We use the method of [1] or [5]. Consider the computation of A on input 0^n. The input is completely symmetric. All processors run the same algorithm and thus are in the same state of the algorithm at any given time. At least one message is sent by each processor at each time until some time T at which no message is sent. From now on the processor cannot change any more due to new messages. Thus all the processors terminate at time T after sending at least nT messages altogether. If $T \geq n/4$, then the message complexity is at least $n^2/4$, which is much more than we need. So assume $T \leq n/4 - 1$. Since the function f is non-constant, there is an input z in $\{0,1\}^n$ such that

$f(z) \neq f(0^n)$. We will show that z has the desired property. Assume for contradiction that there is a processor p_j which terminates at time $T' \leq n/4 - 1$ in the computation of A on z. Since the ring is invariant under circular shifts we may assume that $j = T' + 1$. Now consider the computation of A on input $z_1 0^{2T+1}$, where z_1 is the prefix of z of length $n-2T-1 > n/2 > 2T'+1$. In this computation the processor p_j terminates with the result $f(z)$ but the processor p_{n-T} terminates with the result $f(0^n)$, which is a contradiction. □

The next lemma requires the following definitions:

$$d_1 = 4,\ d_{i+1} = 3d_i 2^{d_i},\ \text{for } i \geq 1 \tag{1}$$

$$T_i = d_i/4,\ \text{for } i \geq 1 \tag{2}$$

Let k be the integer such that

$$d_k \leq m < d_{k+1} \tag{3}$$

Recall that $n = m!$ and $m \geq 192$. Hence $d_2 = 192 \leq m$ and $k \geq 2$. Theorem 1 will follow from the following lemma.

Lemma 3. Let $1 \leq i \leq k-1$, let S be an arbitrary segment of length d_{i+1}, and let z be the string of Lemma 2. Then there are at least $|S|/12$ messages sent in internal links of S during the time period that starts at $t = T_i$ and ends at $t = T_{i+1} - 1$.

Proof. By (1) the segment S consists of $b = 3 \cdot 2^{d_i}$ consecutive segments of length d_i which we denote by $s_1, s_2, \ldots, s_b$. We say that a segment s_j is *rich* if there is a time t, $2T_i \leq t < T_{i+1}$, such that the middle link of s_j was active at time t, and we say that s_j is *poor* otherwise. Let $g = 2^{d_i - 1} + 1$ and $h = g + 2^{d_i+1} - 1$ and consider the segments $s_g, s_{g+1}, \ldots, s_h$ (the middle $2 \cdot 2^{d_i}$ of the $3 \cdot 2^{d_i}$ segments) and distinguish between two cases: Case 1: at least 2^{d_i} of them are rich, and Case 2: more than 2^{d_i} of them are poor. Case 1: There are at least 2^{d_i} rich segments among the segments $s_g, s_{g+1}, \ldots, s_h$. Let s_j be an arbitrary rich segment. By (2), $|s_j| = d_i = 4T_i$. Thus, by Corollary 2, there is at

least one message in the segment s_j at time $t - l$ in the computation of A on z for every $l = 0, 1, 2, \ldots, T_i - 1$. Note that $T_{i+1} - 1 \geq t - l \geq T_i$, by the definition of rich segment. Summing up these messages over all rich segments among the segments $s_g, s_{g+1}, \ldots, s_h$, we obtain the desired number of messages.

Case 2: There are at least $2^{d_i} + 1$ poor segments among the segments $s_g, s_{g+1}, \ldots, s_h$. Since there are 2^{d_i} strings over $\{0,1\}$ of length d_i, we have that there are two indices p, q $(g \leq p < q \leq h)$ and there is a string u over $\{0,1\}$ of length d_i such that the segments s_p and s_q are poor, and u is the input to both segments s_p and s_q in the computation of A on z. In order to prove Case 2 we will prove the following claim.

Claim 1. At least one link among all links of the segments $s_g, s_{g+1}, \ldots, s_h$ is active at time $T_{i+1} - 1$ in the computation of A on z.

Proof. Assume for contradiction that all these links are passive at time $T_{i+1} - 1$ in the computation of A on z. By v we denote the string over $\{0,1\}$ of length $(q-p-1)2^{d_i}$ which is the input to the segment formed by the segments $s_{p+1}, s_{p+2}, \ldots, s_{q-1}$ in the computation of A on z. Note that $z = xuvuy$ for some x and y. Let u_1 be the left half and u_2 the right half of u. Let s'_p be the right half of the segment s_p and let s'_q be the left half of the segment s_q. Let $\bar{s}$ be the segment of length $(q-p)2^{d_i}$ formed by the segments $s'_p, s_{p+1}, \ldots, s_{q-1}, s'_q$. Note that the string $u_2 v u_1$ is the input to the segment $\bar{s}$ in the computation of A on z. For $0 \leq t \leq T_{i+1} - 1$ we denote by c_t the configuration of the segment $\bar{s}$ at time t in the computation of A on z.

We will use below two simple observations. The first one is that the crossing sequences at the middle link of s_p and at the middle link of s_q until time $T_{i+1} - 1$ in the computation of A on z are the same. To prove it use the facts that s_p and s_q are poor (their middle links are passive at t, $2T_i \leq t \leq T_{i+1} - 1$), the string u is the input to s_p and to s_q in the computation of A on z, and apply (a) of Lemma 1 twice with $T = 2T_i$, $T' = T_{i+1} - 1$ and the two middle links. The second observation is obvious: The configuration of an arbitrary

segment at time $t+1$ is uniquely determined by the configuration of this segment at time t, by the message received (from the left) by the leftmost processor of this segment at time t, by the message received (from the right) by the rightmost processor of this segment at time t and by the algorithm A.

Now consider the computation of A on the input $z' \equiv (u_2vu_1)^{n/|u_2vu_1|}$. Note that $|u_2vu_1|$ divides $n = m!$, since $|u_2vu_1| = |\bar{s}| \leq |S| = d_{i+1} \leq d_k \leq m$, by (3). Divide the whole ring into $n/|u_2vu_1|$ consecutive segments of length $|u_1vu_2|$, such that the string u_2vu_1 is the input to every one of them in the computation of A on z'. One can show, by induction on t and by the two observations above, that every such segment is in the same configuration c_t (introduced above) at time t in the computation of A on z' for every $t = 0, 1, 2, \ldots, T_{i+1} - 1$. In particular, all these segments are in the configuration $c_{T_{i+1}-1}$ at time $T_{i+1} - 1$ in the computation of A on z'. Recall that the segment $\bar{s}$ is in the configuration $c_{T_{i+1}-1}$ at time $T_{i+1} - 1$ in the computation of A on z. Since $T_{i+1} - 1 < T_k = d_k/4 \leq m/4 \leq n/4$, there is neither an accepting nor a rejecting state in the configuration $c_{T_{i+1}-1}$, by Lemma 2. By the assumption at the beginning of the proof of the claim and by the definition of $\bar{s}$, all the links of $\bar{s}$ are passive at time $T_{i+1} - 1$ in the computation of A on z, i.e. there is no message in $c_{T_{i+1}-1}$. Consequently, all links of the whole ring are passive at time $T_{i+1} - 1$ in the computation of A on z'. Thus, the computation of A on z' has stopped before time T_{i+1}. Moreover, since the configuration $c_{T_{i+1}-1}$ contains neither an accepting nor a rejecting state, this computation has stopped without any output — a contradiction, which completes the proof of the claim. □

In order to complete the proof of Case 2 (and the proof of Lemma 3), observe, using the definitions of the segment S and the numbers g, h, that the active link mentioned in the claim connects two processors p_j and p_{j+1} such that all links (and all processors) of the segment $[j - d_{i+1}/6 + 1, j + d_{i+1}/6]$ are some links (some processors) of the segment S. Consequently, by Corollary 2, there is at least one message in the segment S at time $T_{i+1} - 1 - l$ for every $l = 0, 1, 2, \ldots, d_{i+1}/6 - 1$. Note that $T_{i+1} - 1 \geq T_{i+1} - 1 - l \geq T_i$ by

(1) and (2). This completes the proof of Lemma 3. □

Theorem 1. Let $n = m!$, $m > 1$. If $f : \Sigma^n \to \{0,1\}$ is a non-constant function, then any asynchronous algorithm for computing f on a bidirectional ring of n anonymous processors requires at least $\frac{1}{24}n(\log^* n - 6)$ messages in the worst case.

Proof. If $m \leq 192$, the lower bound is immediate since the number of messages is at least $n/4$ by Lemma 2. So assume $m \geq 192$. By Lemma 3, the total number of all messages in the computation of A on z is at least $(k-1)n/12$. We now estimate k. Let $e_1 = 4$, $e_{i+1} = 2^{2^{e_i}}$, for each $i \geq 1$. Thus, by (1), $d_i \leq e_i$ for each $i \geq 1$. Now, $\log^* e_i = 2i$ for each $i \geq 1$. These facts with (3) yield:

$$\log^* n = \log^*(m!) \leq \log^* m + 2 \leq \log^* d_{k+1} + 2 \leq \log^* e_{k+1} + 2 = 2k + 4$$

Theorem 1 follows from this estimate. □

2. The Proof of Theorem 2

In this section we also allow segments $[i,j]$ with $i = j$ as well as those with $i > j$ which "go around the ring". A segment is *waiting* if all messages in all its interior links have arrived and no processor in it is able to send any message before receiving a message in the first or last link.

We fix a constant α, $0 < \alpha < 1/2$. We consider A, an arbitrary maximum-finding algorithm on a ring of size n, where $n \geq 8$ is power of two, and with a label set L that contains m distinct labels. We choose m large enough so that the following inequality holds for $l = 2, 4, 8, \ldots n/4$:

$$(n-1)^{(n+l)} \sum_{k=l}^{n-1} \binom{m-l}{k-l} < (1-2\alpha)^{n/l} \frac{(m(m-1)\cdots(m-2l+1))^{n/l}}{(m(m-1)\cdots(m-l+1))^{n/l+1}} \qquad (4)$$

For a string $r = r_1 r_2 \dots r_t$, $r_i \in L$ for $1 \le i \le t$, we denote by $set(r)$ the set of all the different r_i's in r. For $l = 1, 2, 4, \dots, n$ we define

$S_l = \{r |\ r = r_1 r_2 \dots r_l,\ r_i \in L \text{ for } 1 \le i \le l, \text{ and } r_i \ne r_j \text{ for } i \ne j\}$. Note that $|S_l| = |L|(|L|-1)\dots(|L|-l+1)$.

For $l = 1, 2, 4, \dots, n$ and for each $r \in S_l$ we denote by $c(r)$ the following computaion of A in the segment labeled by r. If $r \in S_1$, then $c(r)$ is "do nothing". If $r = ss' \in S_l$, $s, s' \in S_{l/2}$, then $c(r)$ is defined by keeping the first (leftmost), last (rightmost), and middle links of the segment labeled by r very slow, executing $c(s)$ and $c(s')$. Then we increase the speed of the middle link and continue the execution of A in an arbitrary (but fixed) way till the segment labeled by r is waiting. Let $|c(r)|$ be the number of messages sent during $c(r)$. In the proof below we use the following sets for $l = 2, 4, 8, \dots, n/4$:

$C_{2l} = \{ss' |\ s, s' \in S_l,\ ss' \in S_{2l},\ |c(ss')| - |c(s)| - |c(s')| \ge l/2\}$ and

$H_{2l} = \{ss' |\ s, s' \in S_l,\ ss' \in S_{2l},\ ss' \notin C_{2l} \text{ and } s's \notin C_{2l}\}$.

Lemma 4. For $l \in \{2, 4, 8, \dots n/4\}$ and any fixed integer $h \ge 2$, (a) and (b) hold for the sets

$V = \{s_1 s_2 \dots s_h |\ s_i \in S_l \text{ for } 1 \le i \le h,\ s_i s_{i+1} \in H_{2l} \text{ for } 1 \le i \le h-1\}$ and

$W = \{s_1 s_2 \dots s_{2h-1} |\ s_i \in S_l \text{ for } 1 \le i \le 2h-1,\ s_i s_{i+1} \in H_{2l} \text{ for } 1 \le i \le 2h-1\}$.

(a) $|V|^2/|S_l| \le |W|$; and

(b) there is a string $r \in S_l$ such that W contains at least $|V|^2/|S_l|^3$ strings of the form $s_1 s_2 \dots s_{2h-1}$, where $s_1 = s_{2h-1} = r$.

Proof. (a) Let $S_l = \{r_1, r_2, \dots, r_q\}$, i.e. $q = |S_l|$. By b_i (c_i) we denote the number of strings in V with prefix (suffix) r_i for $i = 1, 2, \dots, q$. Hence $|V| = \sum_{i=1}^{q} b_i$. It is easy to see that $s_1 s_2 \dots s_h \in V$ if and only if $s_h s_{h-1} \dots s_1 \in V$, and that if $s_1 s_2 \dots s_h \in V$ and $s_h s_{h+1} \dots s_{2h-1} \in V$ then $s_1 s_2 \dots s_{2h-1} \in W$. Therefore $b_i = c_i$ for each i and $W \ge \sum_{i=1}^{q} c_i b_i = \sum_{i=1}^{q} b_i^2$. By Hölder's (or Cauchy-Bunakovski's) inequality we have $\sum_{i=1}^{q} b_i^2 \ge (\sum_{i=1}^{q} b_i)^2)/q = |V|^2/|S_l|$.

(b) By a_{ij} we denote the number of strings in V with prefix r_i and with suffix r_j for $i, j = 1, 2, \ldots, q$. Let b_k be the maximum number among $b_1, b_2, \ldots, b_q$. Obviously V contains at least $|V|/q$ strings with prefix r_k. Therefore $\sum_{j=1}^{q} a_{kj} = b_k \geq |V|/q$. Since $s_1 s_2 \ldots s_h \in V$ iff $s_h s_{h-1} \ldots s_1 \in V$, $a_{ij} = a_{ji}$ for each i and j. Again, by Hölder's (or Cauchy-Bunakovski's) inequality we have that the number of the strings of the form $s_1 s_2 \ldots s_{2h-1}$ such that $s_1 s_2 \ldots s_h \in V$, $s_h s_{h+1} \ldots s_{2h-1} \in V$, $s_1 = s_{2h-1} = r_k$ and $s_h = r_j$ (for $j = 1, 2, \ldots, q$) is at least $\sum_{j=1}^{q} a_{kj} a_{jk} = \sum_{j=1}^{q} a_{kj}^2 \geq (\sum_{j=1}^{q} a_{kj})^2/q \geq |V|^2/|S_l|^3$ and all of them belong to W. □

Corollary 3. For any given $l \in \{2, 4, 8, \ldots n/4\}$ (a) and (b) below hold for D_j, $j = 0, 1, \ldots$, where $D_0 = H_{2l}$ and for $j = 1, 2, \ldots$

$D_j = \{s_1 s_2 \ldots s_{2^j+1} |\ s_i \in S_l \text{ for } 1 \leq i \leq 2^j + 1,\ s_i s_{i+1} \in H_{2l} \text{ for } 1 \leq i \leq 2^j\}$.

(a) $|D_{j-1}|^2/|S_l| \leq |D_j|$; and

(b) for each $j = 1, 2, 3, \ldots$ there is a string $t_j \in S_l$ such that D_j contains at least $|D_{j-1}|^2/|S_l|^3$ strings of the form $s_1 s_2 \ldots s_{2^p+1}$, where $s_1 = s_{2^p+1} = t_j$.

Lemma 5. If for some $l \in \{2, 4, 8, \ldots n/4\}$, $|C_{2l}| < \alpha |S_{2l}|$, then $(1 - 2\alpha)|S_{2l}| < |H_{2l}|$.

Proof. Let

$E_{2l} = \{ss' \in S_{2l} | s, s' \in S_l,\ ss' \in C_{2l} \text{ and } s's \in C_{2l}\}$;

$F_{2l} = \{ss' \in S_{2l} | s, s' \in S_l,\ ss' \in C_{2l} \text{ and } s's \notin C_{2l}\}$; and

$G_{2l} = \{ss' \in S_{2l} | s, s' \in S_l,\ ss' \notin C_{2l} \text{ and } s's \in C_{2l}\}$.

Clearly $S_{2l} = E_{2l} \cup F_{2l} \cup G_{2l} \cup H_{2l}$, the sets E_{2l}, F_{2l}, G_{2l}, H_{2l} are pairwise disjoint, $E_{2l} \cup F_{2l} = C_{2l}$ and $|F_{2l}| = |G_{2l}|$. Consequently, if $|C_{2l}| < \alpha |S_{2l}|$ them $|E_{2l}| + |G_{2l}| = |E_{2l}| + |F_{2l}| = |C_{2l}| < \alpha |S_{2l}|$ and hence $|H_{2l}| = |S_{2l}| - |E_{2l}| - |F_{2l}| - |G_{2l}| > |S_{2l}| - 2\alpha |S_{2l}| + |E_{2l}| \geq (1 - 2\alpha)|S_{2l}|$. □

Lemma 6. If for some $l \in \{2, 4, 8, \ldots n/4\}$, $|C_{2l}| < \alpha |S_{2l}|$, then there is a string $s_1 s_2 \ldots s_{2^p+1}$ in D_p, where $p = \log(n/l)$ and D_p is as in Corollary 3, such that each $s_i \in S_l$ and for $i \neq j$ $set(s_i) \cap set(s_j) = \emptyset$.

Proof. By D'_p we denote the subset of D_p which contains at least $|D_{p-1}|^2/|S_l|^3$ strings of the form described in (b) of Corollary 3. We are looking for our string among those in D'_p. To the contrary, assume that there is no such string in D'_p, i.e. assume that for each string $v = s_1 s_2 \dots s_{2^p+1}$ in D'_p with $s_1 = s_{2^p+1} = t_p$ there is a pair of indices i, j, $1 \le i, j \le 2^p$, $i \ne j$ such that $set(s_i) \cap set(s_j) \ne \emptyset$. This means that $|set(v)| \le l2^p - 1$ for each v in D'_p. But the number of such v's (i.e. the cardinality of D'_p) can be bounded by $I \equiv (l2^p - 1)^{l(2^p+1)} \sum_{k=l}^{l2^p-1} \binom{|L|-l}{k-l} = (n-1)^{n+l} \sum_{k=l}^{n-1} \binom{|L|-l}{k-l}$. The factor $\sum_{k=l}^{l2^p-1} \binom{|L|-l}{k-l}$ denotes the number of different sets $set(v) - set(t_p)$ and the factor $(l2^p - 1)^{l(2^p+1)}$ denotes the number of different functions f mapping $\{1, 2, \dots, l(2^p+1)\}$ into $\{1, 2, \dots, l2^p - 1\}$, where $f(i) = j$ means that the i-th member of v is exactly the j-th largest element of $set(v)$. On the other hand $|D'_p| \ge |D_{p-1}|^2/|S_l|^3$ and by repeated applications of part (a) of Corollary 3 followed by a single application of Lemma 5 we have that $|D'_p| \ge |D_{p-1}|^2/|S_l|^3 \ge |H_{2l}|^{2^p}/|S_l|^{2^p+1} > ((1-2\alpha)|S_{2l}|)^{2^p}/|S_l|^{2^p+1} = (1-2\alpha)^{n/l}(|L|(|L|-1)\dots(|L|-2l+1))^{n/l}/(|L|(|L|-1)\dots(|L|-l+1))^{n/l+1} \equiv J$. But our upper and lower bounds on $|D'_p|$ (I and J) contradict (4) since $|L| \ge m_0$. □

Theorem 2. Let α be a real number, $0 < \alpha < 1/2$, and let n be a power of 2. Then there is a positive integer m_0 such that (a) and (b) hold.

(a) If L is a finite set of labels with $m_0 \le |L|$ and A is a maximum-finding algorithm for a bidirectional ring of size n labeled by L, then the average number of messages sent by A is at least $(\alpha/4)n \log n$.

(b) If L is a finite set of labels with $m_0 \le |L|$ and A is a maximum-finding algorithm for a unidirectional ring of size n labeled by L. then the average number of messages sent by A is at least $(\alpha/2)n \log n$.

Proof. (a) Assume $n \ge 8$. (The proof is obvious for $n \le 4$.) Let the C's and S's be defined for any particular L, n and A with $|L| \ge m_0$, where m_0 satisfies (4). First we will show that $|C_{2l}| \ge \alpha|S_{2l}|$ for each $l \in \{2, 4, 8, \dots, n/4\}$. To the contrary, assume that

there is $l \in \{2,4,8,\ldots,n/4\}$ with $|C_{2l}| < \alpha|S_{2l}|$. Choose p and $s_1 s_2 \ldots s_{2^p+1}$ according to Lemma 6. Let a ring of size n be labeled by $S = s_1 s_2 \ldots s_{2^p}$, with $s_i = u_i^1 u_i^2 \ldots u_i^l$ for each $i = 1,2,\ldots,2^p$, and $u_i^j \in L$ for each i and j. By Lemma 6, S is a valid input since all its labels are different. Consider the following computation of A in such ring. First execute $c(s_i)$ in the segment labeled by s_i for each i. Keep the transmission speed of the channels connecting the segments labeled by s_1 and $s_2,\ldots,s_{2^p-1}$ and s_{2^p}, and s_{2^p} and s_1 very slow during the execution of the $c(s_i)$'s.

The concatenation of each pair of consecutive s_j's on the ring is in H_{2l}, which implies that for $i = 1,2,\ldots 2^p$ $|c(s_i s_{i+1})| - |c(s_i)| - |c(s_{i+1})| \leq l/2 - 1$ and the number of messages in $c(s_i s_{i+1})$ after the completion of $c(s_i)$ and $c(s_{i+1})$ is smaller than $l/2$ (recall that $s_1 = s_{2^p+1}$). Hence there is a continuation of the computation that mimics $c(s_i s_{i+1})$ for all i, $1 \leq i \leq 2^p$, even though they overlap, as no message in the continuation can reach the processors in the middle of the segments labeled by s_i or s_{i+1} and no two consecutive continuations can interfere. At the end of the computations $c(s_i s_{i+1})$ for all i the whole ring is waiting, which means that the computation has terminated. But if the maximum label is in the segment labeled by s_i, this information cannot reach the middle processor of the segment labeled by s_{i+1} (the processor labeled by $u_{i+1}^{l/2}$, say) before the algorithm stops — contradiction. Hence $|C_{2l}| \geq \alpha|S_{2l}|$ for $l = 2,4,8,\ldots,n/4$.

We now consider the following computation in the ring labeled by $r_1 r_2 \ldots r_n$. Start with the computations in $c(r_1 r_2), c(r_3 r_4), c(r_5 r_6), c(r_7 r_8), \ldots$, then mimic the continuation of $c(r_1 r_2 r_3 r_4), c(r_5 r_6 r_7 r_8)\ldots$, then mimic the continuation of $c(r_1 r_2 \ldots r_8), \ldots$, etc. We call the segments involved *special segments*. Note that unlike the case above, special segments of the same size do not overlap. We charge a special segment labeled with ss' the messages sent in $c(ss')$ after the completion of $c(s)$ and $c(s')$. Hence each message sent is only charged once.

We now estimate the total number of messages received during the computations corresponding to all possible assignments of labels to the processors. Let $l \in \{2,4,8,\ldots n/4\}$ and

consider a special segment z of length $2l$. There are $|C_{2l}| \geq \alpha|S_{2l}| = \alpha|L|(|L|-1)\ldots(|L|-2l+1)$ different ways how to label z by strings in C_{2l} and there are $(|L|-2l)(|L|-2l-1)\ldots(|L|-n+1)$ different ways how to label the rest of the ring. If $ss' \in C_{2l}$, where $s, s' \in S_l$, then there are at least $l/2$ messages received (in z) after finishing $c(s)$ and $c(s')$ and until finishing $c(ss')$. It means that that the special segments of length $2l$ are charged at least $(n/2l)(l/2)|C_{2l}|(|L|-2l)(|L|-2l-1)\ldots(|L|-n+1) \geq (\alpha/4)n|L|(|L|-1)\ldots(|L|-n)$ for each $l = 2, 4, 8, \ldots, n/4$. For $l = n/2$ there are $|L|(|L|-1)\ldots(|L|-n+1)$ different ways how to label a special segment z of length n by strings in S_n. If $ss' \in S_n$, where $s, s' \in S_{n/2}$, then there are at least $n/2$ messages received (in z) after finishing $c(s)$ and $c(s')$ until finishing the computation, because each processor must know the maximum label. Summing up (over all l's), we have that there are at least $(\alpha/4)n \log n|L|(|L|-1)\ldots(|L|-n+1)$ messages. But the factor $|L|(|L|-1)\ldots(|L|-n+1)$ denotes the number of all possible assignments of labels to the processors. This completes the proof of (a).

(b) The proof for a unidirectional ring is the same as for a bidirectional ring except that $C_{2l} = \{ss' \in S_{2l} | s, s' \in S_l,\ |c(ss')| - |c(s)| - |c(s')| \geq l\}$ and to derive the contradiction in the first part of the proof (assuming $|C_{2l}| < \alpha|S_{2l}|$) we show that the last processor of the segment labeled by s_{i+1} (the one labeled by u^l_{i+1}) cannot know the maximum. □

Theorem 2 requires a very large label set. ($|L|$ is exponential in n.) Very recently Bodlaender (private communication) proved an $\Omega(n \log n)$ lower bound for the average case when n is known with a label set L satisfying $|L| \geq cn$. His elegant proof uses extremal graph theory.

References

[1] C. Attiya, M. Snir and M.K. Warmuth, Computing on an anonymous ring, to appear in *J. ACM*; a preliminary version appeared in *Proc. 4-th Annual ACM Symp. on Principles of Distributed Comutation*, Minaki, Ontario, pp. 196–203, August 1985.

[2] H.L. Bodlaender, Distributed algorithms, structure and complexity, Ph.D. Thesis, University of Utrecht, 1986.

[3] J.E.Burns, A formal model for message passing systems, Technical Report # 91, Computer Science Department, Indiana University, Bloomington Indiana, 1980.

[4] D. Dolev, M. Klawe, M. Rodeh, An $O(n \log n)$ unidirectional algorithm for extrema finding in a circle, *J. of Algorithms 3*, pp. 245–260, 1982.

[5] S. Moran and M.K. Warmuth, Gap theorems for distributed computation, *Proc. 5-th Annual ACM Symp. on Principles of Distributed Comutation*, Calgary, Canada, pp. 141–150, August 1985.

[6] J. Pachl, E. Korach and D. Rotem, Lower bounds for distributed maximum-finding algorithms, *JACM 31*, 905-918, 1984.

[7] G.L. Peterson, An $O(n \log n)$ unidirectional algorithm for the circular extrema problem, *Trans. on Programming Languages and Systems 4*, pp. 758–762, 1982.