JOURNAL OF COMPUTER AND SYSTEM SCIENCES 42, 97–118 (1991)

Some Lower Bound Results for Decentralized Extrema-Finding in Rings of Processors*

H. L. BODLAENDER

Department of Computer Science, University of Utrecht, P. O. Box 80.089, 3508 TB Utrecht, The Netherlands

Received June 16, 1986; revised July 20, 1989

We consider the problem of finding the largest of a set of n uniquely numbered processors, arranged in a ring, by means of an asynchronous distributed algorithm without a central controller. Processors are identical, except for their unique number (identity). Using a technique of Frederickson and Lynch we show that arbitrary algorithms that solve this problem on rings where processors know the ring size cannot have a better worst-case number of messages than algorithms that use only comparisons between identities. We show a similar type of result for rings, where the ring size is not known. We use these results to answer a question, posed by Korach, Rotem, and Santoro in 1981 whether each extrema-finding algorithm that uses time n on a ring of n processors must use a quadratic number of messages; and to show a lower bound of $0.683\, n \log(n)$ on the worst-case number of messages for unidirectional rings with known ring size n. Also, we give a lower bound of $\frac{1}{2} n \log(n)$ on the average number of messages for algorithms that use only comparisons on rings with known ring size n.

1. INTRODUCTION

Consider a ring of n processors, distinguished by unique identification numbers. In general it is assumed that the size n of the ring is not known to the processors. There is no central controller. The problem is to design a distributed algorithm for finding the processor with the highest number, using a minimum number of messages. The elected processor can act as a "leader" (central controller). Every processor (possibly several or all processors simultaneously) can start the "election," and every processor has to use the same algorithm. We further assume that the processors work fully asynchronously and cannot use clocks and/or timeouts. (Frederickson and Lynch [11], and Vitanyi [21] analysed the case where this strict assumption of asynchronicity does not hold and show that in that case a significantly smaller number of messages is needed, if one is willing to spend considerably more time.) This latter assumption allows us to assume that the algorithm is message-driven: except for the initialisation of an election, a processor can only

* This work was supported by the Foundation for Computer Science (SION) of the Netherlands Organization for Scientific Research (N.W.O).

0022-0000/91 $3.00

Algorithm	Average number of messages	Worst-case number of messages
Le Lann (1977)	n^2	n^2
Chang & Roberts (1979)	nH_n	$0.5n^2$
Peterson (1982)		$1.44n \log n$
Dolev, Klawe & Rodeh (1982)		$1.356n \log n$

FIG. 1.1. Election algorithms for undirectional rings.

perform actions upon receipt of a message. We also assume that there are no faulty processors and no faults in the communication subsystem.

The leader-finding or "election" problem has received considerable attention, after it was proposed by Le Lann [15] in 1977. It has been studied for unidirectional rings as well as for general, bidirectional rings. Figures 1.1 and 1.2 summarize the solutions presently known for both cases, together with the worst case or average number of messages required for each algorithm. (All logarithms are taken to the base 2.)

For bidirectional rings we assume a global sense of orientation; i.e., each processor knows the left and right direction on the ring. This only strengthens our lower bound results. For most of the bidirectional algorithms of Fig. 1.2 this assumption is unnecessary.

There are some lower bounds known for the election problem on a ring. Burns [6] showed a lower bound of $\frac{1}{4}n \log n$ messages for the worst-case for bidirectional rings. This lower bound was improved to $\frac{1}{2}nH_n$ in [4].[1] Pachl, Korach and Rotem [17] proved a lower bound of $nH_n \approx 0{,}69\, n \log n$ on the average number of

Algorithm	Average number of messages	Worst-case number of messages
Gallager et. al. (1979)		$5n \log n$
Hirschberg & Sinclair (1980)		$8n \log n$
Burns (1980)		$3n \log n$
Franklin (1982)		$2n \log n$
Korach, Rotem & Santoro (1981)	(prob.) $0.70..nH_n$ see also [14]	(prob.) $0.5n^2$
Santoro, Korach & Rotem (1982)		$1.89n \log n$
Bodlaender & van Leeuwen (1985)	(det.) $0.70..nH_n$ see also [14]	(det.) $0.25n^2$
Van Leeuwen & Tan (1985)		$1.44n \log n$
Moran, Shalom & Zaks (1985)		$1.44n \log n$

FIG. 1.2. Election algorithms for bidirectional rings.

[1] H_n is the nth harmonic number, i.e., $H_n = \sum_{i=1}^{n} (1/n) \approx 0.69 \log n$.

Problem	Unidirectional	Bidirectional
Every processor must know the maximum.	$\geq \frac{1}{2}n(n+1)$	$\Omega(n^2)$
One or more arbitrary processors must know the maximum.	$\leq 1.356n \log n + O(n)$	$\leq 1.356n \log n + O(n)$
The maximum must know it is the maximum.	$\geq \frac{1}{2}n(n+1)$	$\Omega(n^2)$
One processor must declare itself as a leader.	$\leq 1.356n \log n + O(n)$	$\leq 1.356n \log n + O(n)$
One processor must declare itself as a leader, and all processors must know the id of the leader.	$\geq \frac{1}{2}n(n+1)$	$\Omega(n^2)$

FIG. 1.3. General bounds for the number of messages for problems that have to be solved in time $\leq n$.

messages required by a unidirectional algorithm (hence the Chang–Roberts algorithm is optimal with respect to the average number of messages) and a lower bound of $\frac{1}{8}n \log n + O(n)$ on the average number of messages for bidirectional rings. For the case that the size of the ring n is initially known to the processors, they proved lower bounds of $(6/5 \log 5)\, n \log n + O(n)$ and $\frac{1}{4}n \log n + O(n)$ on the worst-case number of messages for unidirectional rings and for bidirectional rings, respectively. Recently, Duris and Galil [9] proved $\Omega(n \log n)$ lower bounds for the average number of messages for rings with known ring size. Very recently, these lower bounds were improved in [5] to $\frac{1}{2}n \log n$ for the unidirectional case, and $0.173\, n \log n$ for the bidirectional case.

For the synchronous case, Frederickson and Lynch [11] prove a lower bound of $\frac{1}{2}n \log n + O(n)$ on the worst-case number of messages for (bidirectional) comparison algorithms (a comparison algorithm uses only mutual comparisons between identification numbers $(=, \neq, <, >, \leqslant, \geqslant)$), and for (bidirectional) algorithms, that run in time, bounded by some constant t_n on all rings with size n. This lower bound is also valid for the asynchronous case: every algorithm that runs on a asynchronous ring can also run on a synchronous ring, and the time the algorithm uses is bounded, for instance, by the number of messages that is sent.

This paper is organized as follows. In Section 2 we give some preliminary definitions and results. In Section 3 we relate the worst-case number of messages for arbitrary (asynchronous) algorithms to this number for (asynchronous) comparison algorithms. Our results and proof techniques that we use in this section are very similar to results proven by Frederickson and Lynch [11] for the synchronous case. The results of Section 3 will be used in Section 4 to answer a question, posed by Korach, Rotem, and Santoro in 1981 [13], whether algorithms that use time n, have to send a quadratic number of messages. The answer will be positive or negative, depending on the precise assumptions made about whether we want to find the maximum or whether we want to find a leader (which does not have to be

the maximum) and which processors have to "know" the maximum or leader after completion of the algorithm. In Fig. 1.3 the results of Section 4 are summarized. In Section 5 we consider the case where the size of the ring n is initially know to the processors for unidirectional rings. We improve a result of Pachl, Korach, and Rotem [17] and show an $0.689\, n \log n$ lower bound on the worst-case number of messages for rings with known ring size. We also give a $\frac{1}{2} n \log n$ lower bound on the average number of messages for rings with known ring size, for comparison algorithms only.

2. Definitions and Preliminary Results

Pachl, Korach, and Rotem [17] introduced the notion of full-information algorithms (for unidirectional rings). In full-information algorithms, when a processor sends a message, it sends *everything* it knows. In this way, algorithms have to specify only *when* to send (and no longer *what* to send). Every algorithm A corresponds to an "equivalent" full-information algorithm A': if during execution of algorithm A', a processor p receives a message s from a neighbouring processor, p can decide from its own knowledge, the information that is contained in the message s, and the direction from which it received s, whether or not it would have sent a message on the corresponding moment during execution of algorithm A. A and A' use the same number of messages (although messages sent by A' can be considerably longer than messages sent by A) and the same time. (We assume that in one time unit each processor can send one message. We ignore the time used for calculations in processors.) We also assume that identification numbers are chosen from Z, the set of integers. N^+ denotes the set of positive integers. R^+ denotes the set of positive real numbers.

We will first consider unidirectional full-information algorithms. In a unidirectional full-information algorithm, the "knowledge" of a processor consists of its identification number (abbreviated: its id) and the id's of a number of processors (initially zero) "before" it on the ring. Thus, we may assume, that when a processor sends its "knowledge," it does this by sending a string of identities (see also [17]). So, if a processor with identification number id_* receives a message $\langle \mathrm{id}_1 \cdots \mathrm{id}_k \rangle$, then either it sends a message $\langle \mathrm{id}_1 \cdots \mathrm{id}_k\, \mathrm{id}_* \rangle$ to the next processor, or it does nothing. If a processor id_* receives a message $\langle \mathrm{id}_1 \cdots \mathrm{id}_n \rangle$ and $\mathrm{id}_* = \mathrm{id}_1$, then it knows the id of every processor on the ring. It depends on what exactly we want the algorithm to do whether we have to send some more messages. If we want an arbitrary processor to know the maximum, then we are done. If we want that the processor whose id is the maximum to become aware of this fact or if we want every processor to know the maximum, then some extra messages may have to be sent. In both cases, it is not difficult to find the most efficient way to finish the algorithm, and at most n extra messages in total have to be used. We therefore only consider messages with length $\leqslant n =$ number of processors on the ring.

DEFINITIONS. (i) Let $X \subseteq Z$. $D(X)$ is the set of finite, non-empty sequences of distinct elements of X, i.e.,

$$D(X) = \{\langle s_1 \cdots s_k \rangle \mid k \geqslant 1,\ 1 \leqslant i \leqslant k \Rightarrow s_i \in X;\ i \neq j \Rightarrow s_i \neq s_j\}.$$

(ii) $D = D(Z)$.

(iii) Let $s \in D$. $\text{len}(s)$ is the length of s, i.e., $\text{len}(\langle s_1 \cdots s_k \rangle) = k$.

(iv) Let $s \in D$. $C(s)$ is the set of cyclic permutations of s.

(v) Let $s, t \in D$. t is a subsequence of s, if there are $u, v \in D \cup \{\varepsilon\}$ (ε is the empty sequence), with $s = utv$.

(vi) For $s \in D$, $E \subseteq D$, let $N(s, E) = |\{t \in E \mid t$ is a subsequence of an element of $C(s)\}|$.

DEFINITION. Let $E \subseteq D$. E is exhaustive, iff

(i) $\forall t, u \in D$: $tu \in E \Rightarrow t \in E$

(ii) $\forall s \in D$: $C(s) \cap E \neq \varnothing$.

Theorem 2.1 is a minor extension of a result of Pachl, Korach, and Rotem [17]. We say that a ring is labeled with $t \in D$, if the ring has $\text{len}(t)$ processors with consequtive identities $t_1, t_2, \ldots, t_{\text{len}(t)}$.

THEOREM 2.1 [17]. *Let A be a (unidirectional) full-information algorithm. Let $E = \{t \in D \mid$ a message t will be transmitted when A executes on a ring labeled with $t\}$. Then*

(i) *E is exhaustive*

(ii) *A requires exactly $N(s, E)$ messages with length $\leqslant \text{len}(s)$, when executed on a ring labeled s (and possibly some extra messages with greater length).*

Conversely every *effective computable* (i.e., recursive) exhaustive set $E \subseteq D$ corresponds to a maximum finding algorithm; namely, use the full information algorithm that sends a message if and only if it is an element of E.

An algorithm is said to be a *comparison* algorithm iff no other operations on the id's are used except *mutual* comparisons ($=, \neq, <, >, \leqslant, \geqslant$). We now give the corresponding notion for sets.

DEFINITION. Let $s, t \in D$. $s \equiv t$ (s and t are *order equivalent*), iff $\text{len}(s) = \text{len}(t)$ and $\forall i, j,\ 1 \leqslant i, j \leqslant \text{len}(s)$: $s_i < s_j \Leftrightarrow t_i < t_j$.

DEFINITION. Let $E \subseteq D$. E is comparison-based, iff $\forall s, t \in D$: $s \equiv t \Rightarrow (s \in E \Leftrightarrow t \in E)$.

DEFINITION. Let $E \subseteq D$. E is comparison-exhaustive, iff

(i) E is exhaustive

(ii) E is comparison-based.

THEOREM 2.2. *Let A be a (unidirectional) full-information algorithm, corresponding to a comparison algorithm. Then $E = \{t \in D \mid$ message t will be transmitted when A executes on a ring labeled $t\}$ is comparison-exhaustive.*

Again, conversely, effective computable *comparison*-exhaustive sets $E \subseteq D$ corresponds to *comparison* maximum finding algorithms.

The notion of full information algorithms can be extended to bidirectional algorithms. Our notion of a bidirectional full-information algorithm is very similar to the notion of free algorithms of Frederickson and Lynch [11]. (Free algorithms run on bidirectional *synchronous* rings.) In bidirectional algorithms the behavior of a processor does not necessarily depend fully on the id's of the processors in the neighbourhood it knows, but it can also depend on the order in which it received messages from its neighbours, etc.

DEFINITION. Let $X \subseteq Z$. $D_b(X)$ is the smallest set of strings, such that

(i) $\mathrm{id} \in X \Rightarrow \langle \mathrm{id} \rangle \in D_b(X)$

(ii) $\mathrm{id} \in X$, $k \geqslant 1$, $\{s_1, ..., s_k\} \subseteq D_b(X)$, $\{d_1, ..., d_k\} \subseteq \{l, r\} \Rightarrow (\langle \mathrm{id} \rangle, \langle (d_1, s_1), (d_2, s_2), ..., (d_k, s_k) \rangle) \in D_b(X)$.

We abbreviate $D_b(Z)$ as D_b.

The information that a certain processor has on a certain moment during execution of a bidirectional full-information algorithm can be represented by an element of D_b. For instance, read the string $(\langle \mathrm{id} \rangle, \langle (l, s_1), (r, s_2) \rangle)$ as: my own identification number is id, the first message I received came from my left neighbour and contained information s_1; the second and last message I received came from my right neighbour and contained information s_2.

So if a processor sends a message, it sends all its information, that is, an element of D_b. If a processor p with information $(\langle \mathrm{id} \rangle, \langle (d_1, s_1), ..., (d_k, s_k) \rangle)$ receives a message s, its new information becomes $(\langle \mathrm{id} \rangle, \langle (d_1, s_1), ..., (d_k, s_k), (d_{k+1}, s) \rangle)$, with $d_{k+1} = l$ if the message came from p's left neighbour and $d_{k+1} = r$, if the message came from p's right neighbour.

Now every bidirectional full-information algorithm corresponds to a pair (L, R), L, R subsets of D_b. L corresponds to the messages that are sent to left neighbours, R corresponds to the messages that are sent to right neighbours, in a manner similar to the undirected case. Note that a message that is sent to both neighbours will be a member of $L \cap R$. We will not give a bidirectional variant of the notion of exhaustiveness.

DEFINITION. Let $s \in D_b$. $\tilde{s}$ is obtained by taking the successive integers that appear in s, ignoring other characters:

— if s is of the form $\langle \mathrm{id} \rangle$, with $\mathrm{id} \in Z$, then $\tilde{s} = \mathrm{id}$

— if s is of the form $\langle \mathrm{id}, \langle (d_1, s_1), (d_2, s_2), ..., (d_k, s_k) \rangle \rangle$, then $\tilde{s} = \mathrm{id} \circ \tilde{s}_1 \circ \tilde{s}_2 \circ \cdots \circ \tilde{s}_k$ (where $a_1 \cdots a_n \circ b_1 \cdots b_m = a_1 \cdots a_n b_1 \cdots b_m$).

DEFINITION. Let $s, t \in D_b$. $s \equiv t$ (s and t are order equivalent), iff $\mathrm{len}(\tilde{s}) = \mathrm{len}(\tilde{t})$ and $\tilde{s}_i < \tilde{s}_j \Leftrightarrow \tilde{t}_i < \tilde{t}_j$ for all i, j, $1 \leqslant i, j \leqslant \mathrm{len}(\tilde{s})$. ($\tilde{s}_i$ is the ith integer in the string $\tilde{s}$.)

DEFINITION. Let $s, t \in D_b$. $s \square t$ (s and t are equally typed), iff

(a) s and t are both of the form $\langle \mathrm{id} \rangle$, with $\mathrm{id} \in Z$, or

(b) there exists a $k \in Z$ and $d_1, ..., d_k \in \{l, r\}$, such that

(i) s is of the form $\langle \mathrm{id}_s, \langle (d_1, s_1), ..., (d_k, s_k) \rangle \rangle$;

(ii) t is of the form $\langle \mathrm{id}_t, \langle (d_1, t_1), ..., (d_k, t_k) \rangle \rangle$;

(iii) For all i, $1 \leqslant i \leqslant k$, $s_i \square t_i$.

We call a set $E \subseteq D_b$ comparison based, iff for all s, t with $s \square t$ and $s \equiv t$: $s \in E \Leftrightarrow t \in E$. Similar to the undirected case one has:

THEOREM 2.3. *Let A be a bidirectional full-information comparison algorithm. Let $L, R \subseteq D_b$ be the sets of all messages that could possibly be sent by algorithm A to left and right neighbours, respectively. Then L and R are comparison based sets.*

3. ARBITRARY VERSUS COMPARISON ALGORITHMS

In this section we relate the worst case number of messages needed in arbitrary and comparison algorithms. The results and proof techniques are very similar to results, proven by Frederikson and Lynch [11] for the synchronous case. In the proof we will use a well-known Ramsey theorem (Theorem 3.1). We first consider the unidirectional case.

DEFINITION. For $n \in N^+$ and a set A, let $P_n(A)$ denote the collection of subsets of A with cardinality n, i.e., $P^n(A) = \{X \subseteq A \mid |x| = n\}$.

THEOREM 3.1 (Ramsey's theorem, see, e.g., [1]). *Let A be an infinite set, $k, n \in N^+$, and $C_1, ..., C_k$ a partition of $P^n(A)$. ($\bigcup_{i=1}^{k} C_i = P^n(A)$, $i \neq j \Rightarrow C_i \cap C_j = \varnothing$). Then there exists an infinite homogeneous subset $B \subseteq A$, i.e., $\exists i$: $P^n(B) \subseteq C_i$.*

DEFINITIONS. (i) Let $E \subseteq D$ be exhaustive, and let $A \subseteq Z$, $n \in N^+$. We let $E(n, A)$ denote the set of all strings in E with elements in A and length at most n, i.e., $E(n, A) = \{s \in E \cap D(A) \mid \text{len}(s) \leqslant n\}$.

(ii) Let $n \in N^+$. We fix an enumeration of all permutations of the set $\{1, ..., n\}$, denoted by $\pi_1^n, \pi_2^n, ..., \pi_{n!}^n$.

(iii) Let $X \subseteq Z$, $n = |X|$, and $1 \leqslant i \leqslant n!$. We let $\pi_i^n(X)$ denote the string, obtained by placing the elements of X, in the order, prescribed by π_i^n, i.e., $\pi_i^n(X) \in D(X)$; $\pi_i^n(X) \equiv \pi_i^n$.

THEOREM 3.2. *Let $E \subseteq D$ be exhaustive. There exists a collection of infinite sets $A_1, A_2, ..., A_n, ...$, with*

(i) $A_1 = Z$

(ii) $\forall n \in N^+ : A_n \supseteq A_{n+1}$

(iii) $\forall n \in N^+ : E(n, A_n)$ *is comparison based.*

Proof. We use induction to n. First note that E contains every string $\langle v \rangle$, $v \in Z$. (We use that E is exhaustive.) So $E(1, Z) = E(1, A_1)$ is comparison based.

Now let an infinite set A_n be given, with $E(n, A_n)$ comparison based. We will now show that A_{n+1} can be chosen such that it fulfils the requirements. With induction we define a row of infinite sets $B_0^{n+1}, ..., B_{(n+1)!}^{n+1}$, as follows:

— $B_0^{n+1} = A_n$.

— Let with induction an infinite set B_j^{n+1} be given. Let $C_1 = \{X \subseteq P^{n+1}(B_j^{n+1}) \mid \pi_{j+1}^{n+1}(X) \in E\}$ and $C_2 = \{X \subseteq P^{n+1}(B_j^{n+1}) \mid \pi_{j+1}^{n+1}(X) \notin E\}$. Ramsey theorem 3.1 with $k = 2$ gives that we can choose B_{j+1}^{n+1} such that:

(i) $B_{j+1}^{n+1} \subseteq B_j^{n+1}$

(ii) B_{j+1}^{n+1} is infinite

(iii) $P^{n+1}(B_{j+1}^{n+1}) \subseteq C_1$ or $P^{n+1}(B_{j+1}^{n+1}) \subseteq C_2$.

Finally choose $A_{n+1} = B_{(n+1)!}^{n+1}$. It is obvious that A_{n+1} is infinite and $A_n \supseteq A_{n+1}$.

Now suppose $s_1, s_2 \in D(A_{n+1})$ and $\text{len}(s_1) = \text{len}(s_2) \leqslant n+1$ and $s_1 \equiv s_2$. If $\text{len}(s_1) \leqslant n$, then $s_1 \in E \Leftrightarrow s_2 \in E$, because $A_n \supseteq A_{n+1}$ and $E(n, A_n)$ is comparison based. Now suppose $\text{len}(s_1) = n+1$. There exists a unique π_k^{n+1}, with $s_1 \equiv \pi_k^{n+1} \equiv s_2$. Let S_1, S_2 be the sets of integers, appearing in s_1, s_2, respectively ($s_i = \langle \text{id}_1 \cdots \text{id}_n \rangle \Rightarrow S_i = \{\text{id}_i, ..., \text{id}_n\}$), $S_1, S_2 \subseteq B_k^{n+1}$. The construction of B_k^{n+1} shows that $\pi_k^{n+1}(S_1) = s_1 \in E \Leftrightarrow S_1 \in C_1 \Leftrightarrow P^{n+1}(B_k^{n+1}) \subseteq C_1 \Leftrightarrow S_2 \in C_2 \Leftrightarrow \pi_k^{n+1}(S_2) = s_2 \in E$. So again $s_1 \in E \Leftrightarrow s_2 \in E$. This shows that $E(n+1, A_{n+1})$ is comparison based. ∎

THEOREM 3.3. *Let $E \subseteq D$ be exhaustive. Then there exists a comparison-exhaustive set $F \subseteq D$, such that*

$$\forall n: \max\{N(s, F) \mid s \in D, \text{len}(s) = n\} \leqslant \max\{N(s, E) \mid s \in D, \text{len}(s) = n\}.$$

Proof. Let an exhaustive set $E \subseteq D$ be given, and let the sets $A_1, A_2, ..., A_i, ...$, as implied by Theorem 3.2, be given. Now choose $F = \{s \in D \mid \exists t \in E \mathrm{aD}(A_{\mathrm{len}(s)}): t \equiv s\}$. F is comparison-exhaustive:

(i) Let $t, u \in D$ and $tu \in F$. Then $\exists s \in D$: $\exists v \in D$: $sv \in E \cap D(A_{\mathrm{len}(tu)}) \wedge sv \equiv tu \wedge \mathrm{len}(s) = \mathrm{len}(t) \Rightarrow \exists s \in E \cap D(A_{\mathrm{len}(s)})$: $s \equiv t \Rightarrow t \in F$.

(ii) Let $s \in D$. One can choose $v \in D(A_{\mathrm{len}(s)})$, with $v \equiv s$. (Choose $\mathrm{len}(s)$ different elements from $A_{\mathrm{len}(s)}$ and place them in the right order.) Now $C(v) \cap E \neq \varnothing$. Let $w \in C(v) \cap E$. We can find a $t \in C(s)$, with $t \equiv w$. (Use the same cyclic permutation that is necessary to obtain w from v to obtain t from s.)

So $t \equiv w$ and $w \in D(A_{\mathrm{len}(t)}) \cap E$. This implies $t \in F$ and $C(s) \cap F \neq \varnothing$.

(iii) Let $s, t \in D$ and $s \equiv t$. Then

$$\begin{aligned} s \in F &\Leftrightarrow \exists v\text{: } v \equiv s \text{ and } v \in D(A_{\mathrm{len}(s)}) \\ &\Leftrightarrow \exists v\text{: } v \equiv t \text{ and } v \in D(A_{\mathrm{len}(t)}) \\ &\Leftrightarrow t \in F. \end{aligned}$$

This concludes the proof that F is comparison-exhaustive.

Finally we will show that for all n:

$$\max\{N(s, F) \mid s \in D, \mathrm{len}(s) = n\} \leqslant \max\{N(s, E) \mid s \in D, \mathrm{len}(s) = n\}.$$

Let n be given. Let $s \in D$, with $\mathrm{len}(s) = n$. There exists a $t \in D(A_n)$, with $t \equiv s$. It follows directly from the exhaustiveness of F that $N(s, F) = N(t, F)$. Further, for every subsequence t_0 of t: $t_0 \in E \Leftrightarrow t_0 \in F$ (use that $t_0 \in D(A_n) \subseteq D(A_{\mathrm{len}(t_0)})$); hence $N(t, F) = N(t, E)$. So for all $s \in D$, $\mathrm{len}(s) = n$ there exists a $t \in D$, $\mathrm{len}(t) = N$, with $N(s, F) = N(t, E)$. So

$$\max\{N(s, F) \mid s \in D, \mathrm{len}(s) = n\} \leqslant \max\{N(s, E) \mid s \in D, \mathrm{len}(s) = n\}. \quad \blacksquare$$

The following is an algorithmic variant of Theorem 3.2.

THEOREM 3.4. *Let A be a (unidirectional) extrema finding algorithm for rings. There exist algorithms $B_1, B_2, ..., B_i, ...$ for every $i \in N^+$:*

(i) *algorithm B_i works correctly on every ring* with size $\leqslant i$.

(ii) *algorithm B_{i+1} is an extension of algorithm B_i (i.e., algorithm B_{i+1} does the first i steps the same as algorithm B_i.)*

(iii) *B_i is a comparison algorithm.*

(iv) *For every $j \leqslant i$, the worst case number of messages used by B_i on rings with size j is less than or equal to the worst case number of messages used by algorithm A on rings with size j.*

Proof. Let A' be the full information variant of algorithm A. Let $E = \{t \in D \mid$ a

message t will be transmitted when A' executes on a ring labeled $t\}$. E is exhaustive (cf. Theorem 2.1). Let the sets $A_1, A_2, ..., A_i, ...$ be given, as implied by Theorem 3.2. Now let algorithm B_i be the (full information) algorithm, that sends a message s, if and only if there is a $t \equiv s$, with $t \in D(A_i) \cap E$. We first have to show that B_i indeed is a comparison algorithm, i.e., that it is computable using only comparisons whether there exists a $t \equiv s$ with $t \in D(A_i) \cap E$. Note that if a certain message s is sent by B_i, then also all messages t, with $t \equiv s$ are sent. B_i can use a list of all permutations of i or fewer elements, and with each permutation π_k^j it is stored whether messages s, with $s \equiv \pi_k^j$ are to be sent or not. For each s, one can find, using comparisons only, for which π_k^j $s \equiv \pi_k^j$, and then one can look up whether to send s or not.

It is not difficult to check that (i)—(iv) hold. (Compare the proof of Theorem 3.2.) ▮

Unfortunately, Theorem 3.3 does *not* imply the following conjecture:

Conjecture. Let A be a (unidirectional) extrema finding algorithm for rings. There exists a (unidirectional) comparison algorithm B (for the extrema finding problem for rings) such that for every n, the worst-case number of messages used by algorithm B on rings with size n is at most the worst-case number of messages used by algorithm A on rings with size n.

The key difference between this conjecture and Theorem 3.3 is the computability of F. To let F "yield an algorithm," it must be effectively computable (i.e., recursive) whether a given $s \in D$ is an element of F or not. This is not necessarily true. However, the results of this section show that some worst-case lower bound proofs for comparison algorithms are also valid for arbitrary algorithms. This is when the fact that the comparison is really an *algorithm* (that is, effectively computable) is not used. For instance, a proof can show a lower bound for $\max\{N(s, F) \mid s \in D, \text{len}(s) = n\}$ for all comparison-exhaustive sets, not only for *recursive* comparison-exhaustive sets. The results of this section show that this not only implies the same lower bound on the worst-case number of messages that are sent on rings with size n by comparison algorithms, but also by arbitrary algorithms.

The results can be generalized to the bidirectional case. A bidirectional variant of Theorem 3.2 is:

THEOREM 3.5. *Let A be a bidirectional full-information algorithm, and let L and R be the sets of all messages that could possibly be sent by algorithm A to left and right neighbours, respectively. There exists a collection of infinite sets $A_1, A_2, ...,$ with*

(i) $A_1 \subseteq Z$

(ii) *for all* $n \in N^+$: $A_n \supseteq A_{n+1}$

(iii) *for all* $n \in N^+$ *and all strings* $s, t \in D_b(A_n)$, *with* $s \square t$, $\tilde{s} \equiv \tilde{t}$, *and* $\text{len}(\tilde{s}) = \text{len}(\tilde{t}) \leqslant n$, *one has* $s \in L \Leftrightarrow t \in L$ *and* $s \in R \Leftrightarrow t \in R$.

Proof. The proof is more or less similar to the proof of Theorem 3.2. We will

only stress the differences. As in the proof of Theorem 3.2, we use induction to n. Where we had to use one "Ramsey-step" for each permutation of n elements in the unidirectional case (i.e., for each equivalence class of the strings in D, induced by the equivalence relation $\equiv$), here we have to use the Ramsey theorem for every combination of a *type* of strings s, with $\text{len}(\tilde{s}) = n$, together with an ordering of n elements, i.e., for each equivalence class of the strings in D_b, induced by the equivalence relation $\cong$, where $\cong$ is defined by

$$s \cong t \Leftrightarrow (s \sqsupset t \wedge s \equiv t).$$

In every "Ramsey-step" we divide the subsets of the current B_i^{n+1} with the right cardinality in four classes: those that correspond with messages sent to the left and to the right neighbours, those that correspond with messages, sent only to the left neighbour, etc. Now use Theorem 3.1 with $k = 4$. ▮

Similar to Theorem 3.4 one now can prove:

THEOREM 3.6. *Let A be a (bidirectional) extrema finding algorithm for rings. There exists algorithms $B_1, B_2, ..., B_i, ...$ with for every $i \in N^+$:*

(i) *algorithm B_i works correctly on every ring with size $\leqslant i$*

(ii) *algorithm B_{i+1} is an extension of algorithm B_i (on rings with size $\leqslant i$ algorithm B_i and B_{i+1} behave exactly the same)*

(iii) *B_i is a comparison algorithm*

(iv) *for every $j \leqslant i$, the worst case number of messages used by B_i on rings with size j, is less than or equal to the worst-case number of messages used by algorithm A on rings with size j.*

Proof. The proof is similar to that of Theorem 3.4. The main difference is that we can no longer use the set A_i to obtain algorithm B_i, because messages on rings with size i can have a size much larger than i. Suppose t_i is an upper bound on the number of time steps algorithm A takes on any ring with size i. Note that the size of the largest knowledge of any processor can at most triple in one time step (that is, when the processor receives equally large messages from both its neighbours). So 3^{t_i} is an upper bound on the size of the largest message ever used by algorithm A on rings with size i, and we can use set $B_{3^{t_i}}$ to obtain algorithm A_i, similar to the unidirectional case. ▮

The same remarks we made about lower bound proofs concerning unidirectional rings are also valid for bidirectional rings.

For rings with known ring size, one can prove the following result in the same way as for theorems 3.4 and 3.6.

THEOREM 3.7. *For every extrema finding algorithm A for rings with known ring size n, there exists an extrema finding algorithm B, that uses at most the same worst-case number of messages as A. If A is unidirectional, then B is unidirectional.*

4. Algorithms That Use Time $\leqslant n$

The (running) time of an algorithm is the worst-case time, if each message can take at most one time step and the time needed for internal computations is not counted. In [13] Korach, Rotem, and Santoro posed the question "whether algorithms with running time of n must have a quadratic number of messages in their worst case." The answer to this question depends on what exactly we want the algorithm to do in time n. There are several possibilities:

(A) every processor must know the maximum (i.e., the id of the largest numbered node)

(B) there are one or more arbitrary processors that know the maximum

(C) the processor that has the largest id must know that it is the node with the largest id

(D) exactly one processor must declare itself as a leader

(E) exactly one processor must declare itself as a leader; all other processors must know the id of the leader.

These five problems are closely related. In Fig. 4.1 it is shown which problems are subproblems of which other problems. Also, without much difficulty one can show the following relation:

Theorem 4.1. *If there exists an algorithm that solves one of the five problems* A–E, *has a worst-case running time* $t(n)$, *and uses a worst-case number of messages* $m(n)$, *then for each of the other four problems there exists an algorithm that has a worst-case running time of at most* $t(n)+2n$ *and uses a worst-case number of messages of at most* $m(n)+2n$.

Proof. Suppose we have an algorithm that solves (B). Every processor that knows the maximum sends a message {final, maximum} to its neighbour. Processors that did not know the maximum forward this message. Within n time steps, every processor knows the maximum (A) and (C). If the processor with the maximum id declares itself leader, we have also solved (D) and (E).

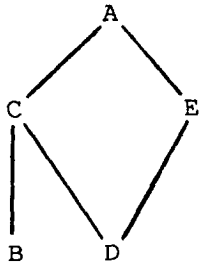

Fig. 4.1. The partial ordering of the problems A–E. A line indicates that the lower problem is a subproblem of the upper problem.

Problem	Unidirectional	Bidirectional
A	$\geq \frac{1}{2}n(n+1)$	$\Omega(n^2)$
B	$\leq 1.356n \log n + O(n)$	$\leq 1.356n \log n + O(n)$
C	$\geq \frac{1}{2}n(n+1)$	$\Omega(n^2)$
D	$\leq 1.356n \log n + O(n)$	$\leq 1.356n \log n + O(n)$
E	$\geq \frac{1}{2}n(n+1)$	$\Omega(n^2)$

FIG 4.2. Bounds for the worst case number of messages for algorithms that use time $\leqslant n$.

Suppose we have an algorithm that solves (D). The leader can initiate a message that makes a full tour around the ring and determines the maximum id. The resulting algorithm uses n extra time steps, and solves (B). To solve the other problems, use the transformation given above. All the other cases now follow directly. ▮

In Fig. 4.2 we summarize the bounds we prove on algorithms that use time $\leqslant n$.

THEOREM 4.2. *Every unidirectional algorithm for problem C that uses time n, must use at least* $\frac{1}{2}n(n+1)$ *messages in the worst-case.*

Proof. If the processor with the largest id must be aware of the fact that it has the largest id after time n, then it must have sent a message to its successor, and this message must have been propagated around the whole ring, until it returned to the processor that originated the message. The exhaustive set, induced by the algorithm must therefore contain all messages $s_1, ..., s_n$, with $\forall i,\ 2 \leqslant i \leqslant n$: $s_1 \geqslant s_i$. (This means essentially that every message, used by the Chang–Roberts algorithm must be used here too.) Now this implies a worst case lower bound of $\frac{1}{2}n(n+1)$ messages on rings with size n. (Consider rings labeled with $n,\ n-1,\ n-2, ..., 2, 1$. Compare this result with the results of Chang and Roberts [7].) ▮

THEOREM 4.3. *For every bidirectional algorithm for problem C, that uses time* $\leqslant n$, *there is a* $c \in R^+$, *such that the algorithm uses at least* cn^2 *messages in the worst-case.*

Proof. Let A be a bidirectional algorithm for problem C that uses time $\leqslant n$ on rings with size n and suppose there does not exist a $c \in R^+$, such that the algorithm uses at least cn^2 messages in the worst-case (on rings with size n). Without loss of generality we can assume that A is a full-information algorithm. We first assume A is a comparison algorithm.

Let rings $r^{k,l}$ be defined as follows: the size of a ring $r^{k,l}$ is kl, and the id of the ith node of $r^{k,l}$ is

$$((-i-1) \bmod l)\, k + \left\lceil \frac{i}{l} \right\rceil \qquad (1 \leqslant i \leqslant kl).$$

So, for instance, the processors of $r^{3,5}$ have successive id's:

12 9 6 3 15 11 8 5 2 14 10 7 4 1 13.

Note $r^{k,l}$ consists of k pieces of l nodes. Each piece consists of a decreasing row of $l-1$ id's; the last id is larger than every other id in the piece. The pieces are—in a certain sense—mutually placed in a decreasing order: the ith processor in the jth piece has a larger id than the ith processor in the $(j+1)$th piece.

We now suppose the ring is fully synchronous: all message transmission times are equal and we ignore time needed for calculations in processors. This means, that when a processor sends a message on a time step $t+1$, this message can contain the id's of at most $2t+1$ processors: the id of the processor itself, the id's of t processors immediately to the left of the processor, and the id's of t processors immediately to the right of the processor. This also means that, when for two processors p_0, p_1 the strings consisting of these $2t+1$ id's are equally ordered, we may assume that p_0 sends a message on time step $t+1$ if and only if p_1 sends a message on time step $t+1$. (We use that the ring is fully synchronous and the algorithm is a comparison algorithm.)

Let $\Phi(k, l)$ be the smallest number t, such that on the ring $r^{k,l}$ all messages that are sent by the full information algorithm A after time t (under the assumption of synchronicity) contain the value of the maximum (and, of course, many other data).

CLAIM 4.3.1. $\forall l: \exists c_l: \forall k: \Phi(k, l) \leqslant c_l$.

Proof. Suppose not. Let l be given, such that $\forall c: \exists k: \Phi(k, l) \geqslant c$. Now we first claim that on every ring $r^{k,l}$ at least the first $\lfloor \frac{1}{4}kl \rfloor$ time steps messages are sent that do not contain the maximum identity. This we can see by taking k' such that $\Phi(k', l) \geqslant \lfloor \frac{1}{4}kl \rfloor$. On the ring $r^{k',l}$ on each of the first $\lfloor \frac{1}{4}kl \rfloor$ time steps there are messages sent that do not contain the value of the maximum ($=k'l$). Now use that, for each part of the ring $r^{k',l}$ with length $\leqslant 2\lfloor \frac{1}{4}kl \rfloor + 1$ that does not contain the maximum id, there exists an equally ordered part of the ring $r^{k,l}$ that also does not contain the maximum id of this ring ($=kl$). So on $r^{k,l}$ the first $\lfloor \frac{1}{4}kl \rfloor$ time steps there are messages sent that do not contain the maximum id.

Finally we use, that for each part of $r^{k,l}$ with length $\leqslant 2\lfloor \frac{1}{4}kl \rfloor + 1$ that does not contain the maximum id, there are at least $\frac{1}{2}k + O(1)$ equally ordered parts. So on each of the first $\lfloor \frac{1}{4}kl \rfloor$ time steps there are at least $\frac{1}{2}k + O(1)$ messages sent; so in total at least approximately $\frac{1}{8}k^2l$ messages. When we take l as a constant, and let k grow to infinity, we see that the algorithm costs a quadratic number of messages in the worst case; contradiction. ∎

As the algorithm is "message-driven" (a processor can only send a message upon reception of a message, except for the first time step), each message m, except those sent on the first time step, has a "predecessor" message, that is, the message that "triggered" the processor to send message m. The processor that sent the message

on time step 1, obtained by taking recursively the predecessor message of m is called the originator of m; the successive messages between the originator of m and m we call the chain of messages of m.

Now look at the last message m, that arrives at the processor with the maximum id, and look at its originator and chain of messages. Claim 4.3.1 shows that the distance between the originator and the processor with the maximum id is at most c_l on a ring $r^{k,l}$. As the processor $p_{\max}$ with the maximum id must know all id's of all processors after time step n, there are basically the following possibilities:

— $p_{\max}$ is the originator of m, the chain of messages of m "goes around the whole ring": each processor sends one message of this chain; the successive messages have either travelled around the whole ring in positive, or in negative direction. (We say "m has travelled around the ring in positive/negative direction") (see Fig. 4.3a).

— the originator of m is $p_{\max}$ or a processor with distance $\leqslant c_l$ to $p_{\max}$; the chain of messages goes from the originator to a node, approximately halfway around the ring and then returns. So m can inform $p_{\max}$ of the id's of $\pm \frac{1}{2}n + c_1$ processors, all on approximately the same half of the ring (seen from $p_{\max}$). Some other messages(s) must inform $p_{\max}$ of the other id's (see Fig. 4.3b).

CLAIM 4.3.2. *For each l, there are only finitely many k such that the last message m, that is received at the maximum on ring $r^{k,l}$ has travelled around the ring in negative direction.*

Proof. Suppose not. For a ring $r^{k,l}$ there are $\frac{1}{2}k + O(1)$ pieces, that are equally ordered to the piece, consisting of the maximum and the $\lfloor \frac{1}{2}kl \rfloor$ id's of the processors before the maximum. We again use that A is comparison based, hence if on $r^{k,l}$ the last message received at the maximum has travelled around the ring in negative direction, then on each of the first $\lfloor \frac{1}{2}kl \rfloor$ time steps at least $\frac{1}{2}k + O(1)$ messages are sent, so in total at least approximately $\frac{1}{4}k^2l$. Now we can keep l fixed, and let k grow to infinity: the algorithm uses a quadratic number of messages in the worst-case; contradiction. ∎

CLAIM 4.3.3. *For each l, there is at least one k such that the last message m that is received at the maximum on ring $r^{k,l}$ has travelled around the ring in positive direction.*

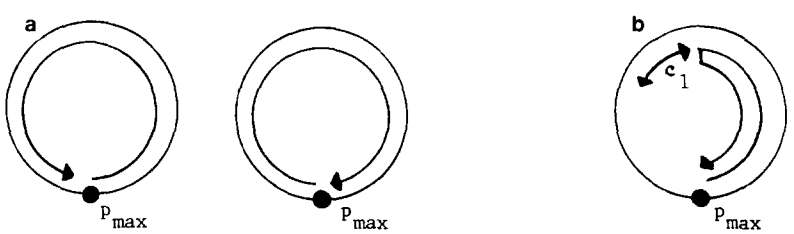

FIGURE 4.3.

Proof. Suppose not. Then there are l and k', such that for all $k \geqslant k'$ the chain of messages of m goes from the originator to a node approximately halfway around the ring and then returns. Because the maximum must be informed of the id's of all nodes on the ring, the turning point has a distance of $\frac{1}{2}kl + O(c_l)$. Further, notice that if a message chain goes halfway around the ring and then back to the maximum on a ring $r^{k,l}$, then a message chain, with a similar length and behavior is sent on $r^{m,l}$, for all $m \geqslant k$. So when k grows to infinity, the number of messages sent grows at least linearly to $k^2 l/c_l$: the algorithm uses a quadratic number of messages in the worst-case; contradiction. ∎

Now consider rings $r^{1,l}$, i.e., rings labeled $l\ l-1\ l-2 \ldots 2\ 1$. The processor labeled l must send a message in positive direction, this message will continue to travel in this direction, for *at least* the first $\lceil \frac{1}{2}l \rceil$ time steps. (This follows from Claim 4.3.3—the fact that the algorithm is a comparison algorithm and the fact that during the first $\lceil \frac{1}{2}l \rceil$ time steps, processors cannot distinguish between the cases $k = 1$ and $k > 1$.) Because the algorithm is a comparison algorithm, this means that at least $l + l - 1 + l - 2 + \cdots + \lceil \frac{1}{2}l \rceil$ messages $\approx \frac{3}{8}l^2$ messages are sent on the ring $r^{1,l}$, for every l. Hence the algorithm sends a quadratic number of messages.

We now have shown that every comparison algorithm for problem C that uses time $\leqslant n$ must use a quadratic number of messages. Because we have never used in our proof that the way in which processors decide to send or not to send a certain message must be by an *algorithm* in the processors, i.e., must be effectively computable by a processor, the results of the previous section show that the obtained result is also valid for arbitrary algorithms. ∎

The bounds proven in Theorems 4.2 and 4.3 are of course also valid for problem A: it contains problem C as a subproblem. Theorem 4.4 shows that these bounds are also valid for problem E.

THEOREM 4.4. *Every algorithm for problem E that uses time $\leqslant n$ must use at least $\frac{1}{2}n(n+1)$ messages in the worst-case for unidirectional rings or $\Omega(n^2)$ messages in the worst-case for bidirectional rings.*

Proof. First suppose we have a comparison algorithm for problem E that uses time $\leqslant n$, but also uses a smaller number of messages. We will derive a contradiction with Theorem 4.2 and Theorem 4.3 for the unidirectional and the bidirectional case, respectively. We also assume the algorithm is a full-information algorithm. We claim that when a processor knows what id the leader has, then it must know the id's of all the processors. Suppose this is not the case: a processor p decides that the processor with identification number id is the leader (possibly p itself is the leader), and the information of p does not contain all id's of all processors. Suppose p knows the id's of k successive processors. We now define a ring r' with $2k$ nodes. The first k processors have the id's of the successive processors p knows the id of, the second k processors have id's, such that this part of the ring is equally ordered to the first part of k id's. Now the processor p_0 with the same id as p, and the pro-

cessor p_1 with distance k to it can behave similarly (the algorithm is a comparison algorithm), so it is possible that p_0 and p_1 will both decide on the id of the leader, but then these id's will not be the same; contradiction.

Because when a processor knows the id's of all the processors it also knows the maximum id, the full-information comparison algorithm can solve problem A in the same time and with the same number of messages. This contradicts Theorem 4.2 or Theorem 4.3. The results of Section 3 show that the obtained bounds are also valid for arbitrary algorithms. ▮

We assume that the number of bits, needed to express an id is m.

THEOREM 4.5. *There exists a unidirectional algorithm for problem B that uses time n, and* $\leqslant 1.356\, n \log n + O(n)$ *messages in the worst-case, each consisting of* $O(m + \log n)$ *bits.*

Proof. Basically we use the algorithm of Dolev, Klawe, and Rodeh [8]. To each message we add two fields: one contains the id of the originator, the other the current maximum known id. If a processor id receives a message $\{\mathrm{id}_{0r}, \mathrm{id}_{\max}, \text{otherdata}\}$, and $\mathrm{id} \neq \mathrm{id}_{0r}$, then it uses the field "otherdata" to execute the algorithm of [8]. If this algorithm decides to let processor id send a message {newotherdata}, then instead it sends a message $\{\mathrm{id}_{0r}, \max(\mathrm{id}_{\max}, \mathrm{id}), \text{newotherdata}\}$. (Initialisation messages of the algorithm of [8] {data} are changed in {id, id, data}.) If a processor id receives a message $\{\mathrm{id}_{0r}, \mathrm{id}_{\max}, \text{otherdata}\}$ and $\mathrm{id} = \mathrm{id}_{0r}$, then $\mathrm{id}_{\max}$ is the maximum id of all the processors. It is easy to see that the algorithm works correctly, uses time n, does not use more messages than the algorithm of [8], and has a message size of $O(m + \log n)$ bits. ▮

Note that the use of the algorithm of [8] is not essential in the proof. For instance, when a better upper bound is found for unidirectional extrema finding, then this upper bound is also valid for problem B and algorithms that use time n.

THEOREM 4.6. *There exists a unidirectional algorithm for problem E that uses time n and* $1.356\, n \log n + O(n)$ *messages in the worst-case.*

Proof. Use the full-information variant of the algorithm of Dolev, Klawe, and Rodeh [8], during the first n time steps. When a processor receives a message that contains its own id in the first field, then it knows the successive id's of all the processors; hence it can decide what processors also receive messages that contain their own id in the first field and whether it has the largest id of this set of processors. If this is the case, then it declares itself as a leader. In this way in time n exactly one processor will declare itself as a leader; the algorithm does not use more messages than the algorithm of [8]. ▮

Note that the size of the messages becomes as large as $O(nm)$ bits. It is an open question whether there exists an algorithm for problem E that uses time n, and $O(n \log n)$ messages of $O(m + \log n)$ bits each in the worst-case.

5. Lower Bounds for Algorithms on Rings with Known Ring Size

In this section we consider the extrema-finding problem on rings "that know the ring size"; i.e., the number of processors n is initially known to the processors. For this problem a worst-case lower bound of $(6/5 \log 5)\, n \log n + O(n) \approx 0.51\, n \log n + O(n)$ messages was proved by Pachl, Korach, and Rotem [17]. We will improve on this result by looking at comparison algorithms.

To start our analysis, again we replace an algorithm by its full-information variant. Notice that if a processor receives a message with length $n-1$ then it knows all the id's of the processors, so it also knows the maximum. So now the notion of exhaustiveness is replaced by the following definition:

Definition. Let $E \subseteq D$. E is *exhaustive for known ring size n*, iff

(i) $\forall t, u \in D$: $tu \in E \Rightarrow t \in E$

(ii) $\forall s \in D$ *with* $\text{len}(s) = n$: $\exists t \in C(s)$: $\exists u, v \in D$: $t = uv$, $\text{len}(u) = n-1$, $\text{len}(v) = 1$, $u \in E$.

Definition. Let $E \subseteq D$. E is comparison-exhaustive for known ring size n, iff E is exhaustive for known ring size n and E is comparison based. Again we let $N(s, E)$ denote $|\{t \in E \mid t \text{ is a subsequence of an element of } C(s)\}|$. We let $K(n)$ ($K^{cb}(n)$) denotes the worst-case number of messages sent by the "best" algorithm (comparison algorithm) for rings with known ring size n.

Definition. $K(n) = \min\{\max_{s \in D,\, \text{len}(s)=n} N(s, E) \mid E \subseteq D \text{ is exhaustive for known ring size } n\}$.

$K^{cb}(n) = \min\{\max_{s \in D,\, \text{len}(s)=n} N(s, E) \mid E \subseteq D$ is comparison exhaustive for known ring size $n\}$.

Theorem 5.1. (i) *$K(n)$, $K^{cb}(n)$ are lower bounds for the worst-case number of messages sent by unidirectional algorithms for known ring size n.*

(ii) $K(n) = K^{cb}(n)$.

Proof. (i) This follows from the definitions. Compare with Section 2.

(ii) Compare with Theorems 3.2 and 3.4. ∎

Lemma 5.2. $K(5) = K^{cb}(5) \geq 12$.

Proof. Let E be comparison exhaustive for known ring size 5. If $\langle 1, 2\rangle$ and $\langle 2, 1\rangle \in E$ then $N(\langle 1, 2, 3, 4, 5\rangle, E) \geq 12$: $\langle 1\rangle$, $\langle 2\rangle$, $\langle 3\rangle$, $\langle 4\rangle$, $\langle 5\rangle$, $\langle 1, 2\rangle$, $\langle 2, 3\rangle$, $\langle 3, 4\rangle$, $\langle 4, 5\rangle$ $\langle 5, 1\rangle \in E$ (use that E is comparison based), and at least one string with length 3 and one string with length 4, that are substrings of an element of $C(\langle 1, 2, 3, 4, 5\rangle)$, must be an element of E. Without loss of generality suppose $\langle 1, 2\rangle \in E$, $\langle 2, 1\rangle \notin E$. If $\langle 1, 2, 3\rangle \in E$ then $N(\langle 1, 2, 3, 4, 5\rangle) \geq 13$. (Use

again that E is comparison based.) So assume $\langle 1, 2, 3\rangle \notin E$. There must be a substring with length 3 of an element of $C(\langle 1, 2, 3, 4, 5\rangle)$ an element of E, so $\langle 4, 5, 1\rangle \in E$. Also $\langle 1, 5, 4\rangle \in E$. (Use ring $\langle 5, 4, 3, 2, 1\rangle$. $\langle 3, 2, 1\rangle \notin E$, because $\langle 3, 2, 1\rangle \in E \Rightarrow \langle 3, 2\rangle \in E \Rightarrow \langle 2, 1\rangle \in E$, etc.)

Now consider the ring labeled $\langle 3, 5, 2, 4, 1\rangle$. $\langle 3, 5, 2\rangle \in E$, $\langle 2, 4, 1\rangle \in E$ and $\langle 5, 2, 4\rangle \notin E$, $\langle 4, 1, 3\rangle \notin E$, $\langle 1, 3, 5\rangle \notin E$. This means that $\langle 3, 5, 2, 4\rangle \in E$ or $\langle 2, 4, 1, 3\rangle \in E$. Because $\langle 3, 5, 2, 4\rangle \equiv \langle 2, 4, 1, 3\rangle$, both are elements of E. Hence $N(\langle 3, 5, 2, 4, 1\rangle, E) \geqslant 5+3+2+2 = 12$. So $\max_{s \in d,\, \mathrm{len}(s)=n} N(s, E) \geqslant 12$. ∎

LEMMA 5.3. $\forall k, n \in N^+$: $K^{cb}(kn) \geqslant kK^{cb}(n) + nK^{cb}(k) - kn + k + n - 1$.

Proof. Let π^n be a permutation of the elements $\{1, ..., n\}$, and let π^k be a permutation of the elements $\{1, ..., k\}$. Let $\pi^k \circ \pi^n$ be the permutation of $\{1, ..., kn\}$, defined by

$$(\pi^k \circ \pi^n)(i) = n \cdot \pi^k(((i-1) \bmod k) + 1) + \pi^n\left(\left\lceil \frac{i}{k} \right\rceil\right) \quad (1 \leqslant i \leqslant k - n).$$

In this way the ring labeled by the successive values of $\pi^k \circ \pi^n$ consists of n pieces of k elements. Every piece of k successive elements is order equivalent to an element of $C(\pi^k)$; the relative ordering we obtain by dividing the ring in n pieces of k elements each and then comparing these pieces is given by π^n.

We now claim, that for every (full-information) comparison algorithm A the worst-case number of messages that is sent by A, taken over all rings $\pi^k \circ \pi^n$ is at least $k \cdot K^{cb}(n) + nK^{cb}(k) - kn + k + n - 1$. First note that we obtain a correct (full-information) comparison algorithm for rings with known ring size k, by letting A^k send a message, iff it has length at most $k-1$ and is sent by algorithm A. Hence A sends at least $nK^{cb}(k)$ messages with length $\leqslant k-1$. It is easy to see that A sends $\geqslant n$ messages with length k. Now, for each π^k, let A^{n,π^k} be the full information algorithm for rings with size n, that sends a message $s = \langle s_1, ..., s_m\rangle$ $(m \leqslant n)$, iff A sends a message $t = \langle t_1, ..., t_r\rangle$ $mk \leqslant r \leqslant nk$ with t is a subsequence of an element of $C(\pi^k \circ \pi_k^n)$ for certain π^n, and $s \equiv \langle t_k, t_{2k}, t_{3k}, ..., t_{mk}\rangle$. One can check that $A^{n,\pi}$ is a correct algorithm; let the number of messages sent by A^{n,π^k} be α, then the number of messages sent by A with length between $k+1$ and $(n-1)k$ (inclusive) is $k \cdot \alpha - kn \geqslant kK^{cb}(n) - kn$. Finally, for each i, with $(n-1)k + 1 \leqslant i \leqslant nk - 1$, A will send at least one message with length i.

The total number of messages is hence at least $kK^{cb}(n) + nK^{cb}(k) - nk + k + n + 1$. ∎

A similar, but weaker result was proved by Pachl, Korach, and Rotem [17]. From Lemma 5.3 it follows that for any fixed $k \geqslant 1$ and infinitely many n,

$$K(n) \geqslant \frac{(K(k) - k + 1)\, n \log n}{k \log n} + O(n).$$

So now we have proved:

COROLLARY 5.4. *There are infinitely many n, for which the worst-case number of messages, sent by any unidirectional algorithm on a ring with known ring size n, is at least* $(8/5 \log 5)\, n \log n + O(n) \approx 0.689\, n \log n + O(n)$.

This result improves the lower bound of $(6/5 \log 5)\, n \log n + O(n) \approx 0.51\, n \log n + O(n)$ messages, proved by Pachl, Korach, and Rotem [17], and is quite close to the best known lower bound for the worst-case number of messages in the case that the ring size is not known (i.e., $nH_n \approx 0.693\, n \log n + O(n)$).

Until recently no results were known on lower bounds for the average number of messages for algorithms that can use a known ring size. We will prove a result for comparison algorithms.

LEMMA 5.5. *Let E be a comparison exhaustive set for known ring size n. Then,*

$$k \mid n,\ k \neq n \Rightarrow (\forall s \in D: \operatorname{len}(s) = k \Rightarrow C(s) \cap E \neq \varnothing).$$

Proof. Let $k \mid n$, $k \neq n$, $s \in D$, $\operatorname{len}(s) = k$. Because E is comparison based, we may assume without loss of generality, that $s \in D(\{1, ..., k\})$; i.e., s can be written as a permutation of the elements $\{1, ..., k\}$. Now label a ring r as

$$r_i = (s_{i \bmod k}) \cdot \frac{n}{k} + \left\lceil \frac{i}{k} \right\rceil \qquad (1 \leqslant i \leqslant n).$$

The size of r is n, and for all $t \in D$, with t as a subsequence of an element of $C(r)$ and $\operatorname{len}(t) = k$, there is a $u \in C(s)$ with $t \equiv S$. There must be at least one $t \in E$, with t a subsequence of an element of $C(r)$ and $\operatorname{len}(t) = k$. Now there is a $u \in C(s)$, with $t \equiv s$, $t \in E$; hence $u \in E$, so $C(s) \cap E \neq \varnothing$.) ∎

THEOREM 5.6. *Let $n = \prod_{i=1}^{l} p_i$; let $p_1, ..., p_l$ be prime numbers. For every unidirectional comparison algorithm A for known ring size n the average number of messages sent is at least*

$$n \cdot \sum_{i=1}^{l} \left(1 - \frac{1}{p_i}\right) - 1.$$

Proof. Let E be the comparison exhaustive set for known ring size n, corresponding to A. We can now use a technique similar to a technique used in [17].

We are going to estimate the total number of messages sent in all rings, labeled with permutations of $\{1, ..., n\}$. The total number of contiguous label sequences of length k in all these rings is $n(n-1)! = n!$ These can be gathered together in groups of size k, where every group consists of all the cyclic permutations of one sequence. If $k \mid n$, $k \neq n$, then E intersects each of these groups (Lemma 5.5.) So $k \mid n$, $k \neq n$ implies that there are at least $n(n-1)!/k$ messages with length k sent.

Furthermore, notice that on any ring, for $k_1 \leqslant k_2$ the number of messages with length k_1 that are sent is at least the number of messages with length k_2 that are sent. There must also be at least one message with length $n-1$ sent.

Let $M(k)$ denote the total number of messages sent with length k, over all rings, labeled with a permutation of $\{1, ..., n\}$. Write $f(j) = \prod_{i=1}^{j} p_i$:

$$\sum_{k=1}^{n-1} M(k) \geqslant \sum_{j=1}^{l} \sum_{k=f(j-1)+1}^{f(j)} M(k)$$

$$\geqslant \sum_{j=1}^{l} \left((f(j) - f(j-1)) \frac{n!}{f(j)} \right) n!$$

$$= \left(\sum_{j=1}^{l} \left(1 - \frac{1}{p_j} \right) n! \right) - n!$$

This means that the average number of messages sent by algorithm A is at least $n \cdot \sum_{j=1}^{l} (1 - (1/p_j)) - 1$. ∎

COROLLARY 5.7. *For every unidirectional comparison algorithm A for known ring size $n = 2^l$ ($l \in N$) the average number of messages that is sent is at least $\frac{1}{2} n \log n - 1$.*

THEOREM 5.8. *Let A be a unidirectional algorithm for known ring size $n = \prod_{i=1}^{l} p_i$ ($p_1, ..., p_l$ prime numbers). There exists an infinite set of labels X in Z, such that the number of messages sent, averaged over all rings with size n, labeled with elements of X, is at least*

$$n \cdot \sum_{i=1}^{l} \left(1 - \frac{1}{p_i} \right) - 1.$$

If $n = 2^l$ ($l \in N$), then this average number is at least $\frac{1}{2} n \log n - 1$..

Proof. One can easily prove a variant of Theorem 3.2 for rings with known ring size. Now let E be the exhaustive set for known ring size n, corresponding to A, apply the theorem and choose $X = A_n$. Then use Theorem 5.6 and Corollary 5.7. ∎

A stronger version of Theorem 5.8 was recently obtained, using powerful new techniques and results of extremal graph theory [5, 9]. Using more precise counting arguments, the lower bound can be improved to $(1 - \varepsilon) nH_n$, for the average number of messages of unidirectional comparison algorithms (and hence the worst-case number of messages for arbitrary algorithms) for rings with known ring size n (n large enough and of the form $m!$). A similar lower bound of $(\frac{1}{2} - \varepsilon) nH_n \approx 0.346\, n \log n$ can be proven for the bidirectional case (see [3]).

REFERENCES

1. J. BARWISE, (Ed.), "Handbook of Mathematical Logic," North-Holland, New York, 1977.
2. H. L. BODLAENDER AND J. VAN LEEUWEN, New upperbounds for decentralized extrema-finding in a ring of processors, *in* "Proceedings, 3rd Annu. Sympos. on Theoretical Aspects of Computer Science," Lect. Notes in Comput. Sci., Vol. 210, pp.119–129, Springer-Verlag, New York/Berlin, 1986.

3. H. L. BODLAENDER, New lower bounds for distributed leader finding in asynchronous rings of processors, *in* "Proceedings, GI-17, Jahrestagung, Informatik Fachberichte, Vol. 156, pp. 82–88, Springer-Verlag, New York/Berlin, 1987.
4. H. L. BODLAENDER, A better lower bound for distributed leader finding in bidirectional asynchronous rings of processors, *Inform. Proc. Lett.* **27** (1988), 287–290.
5. H. L. BODLAENDER, New lower bound techniques for distributed leader finding and other problems on rings of processors, *Theoret. Comput. Sci.*, to appear.
6. J. E. BURNS, "A Formal Model for Message Passing Systems," Technical Rep. 91, Computer Science Dept., Indiana University, Bloomington, IN, 1980.
7. E. CHANG AND R. ROBERTS, An improved algorithm for decentralized extrema-finding in circular configurations of processes, *Comm. ACM* **22** (1979), 281–283.
8. D. DOLEV, M. KLAWE, AND M. RODEH, An $O(n \log n)$ unidirectional distributed algorithm for extrema-finding in a circle, *J. Algorithms* **3** (1982), 245–260.
9. P. DURIS AND Z. GALIL, Two lower bounds in asynchronous distributed computation, *in* "Proceedings, 28th Annu. Sympos. on Fundamentals of Comput. Sci., 1987," pp. 326–330.
10. W. R. FRANKLIN, On an improved algorithm for decentralized extrema-finding in circular configurations of processors, *Comm. ACM* **25** (1982), 336–337.
11. G. N. FREDERICKSON AND N. A. LYNCH, The impact of synchronous communication on the problem of electing a leader in a ring, *in* "Proceedings, 16th ACM Sympos. Theory of Computing, 1984," pp. 493–503. Revised version "Electing a Leader in a Synchronous Ring" in *J. Assoc. Comput. Mach.* **34** (1987), 98–115.
12. D. S. HIRSCHBERG AND J. B. SINCLAIR, Decentralized extrema-finding in circular configurations of processors, *Comm. ACM* **23** (1980), 627–628.
13. E. KORACH, D. ROTEM, AND N. SANTORO, A probabilistic algorithm for decentralized extrema-finding in a circular configuration of processors, *Res. Rep.* CS-81-19, Dept. of Computer Science, University of Waterloo, Waterloo, 1981.
14. C. LAVAULT, Average number of messages for distributed leader-finding in rings of processors, *Inform. Proc. Lett.* **30** (1989), 167–176.
15. G. LE LANN, Distributed systems—Towards a formal approach, *in* "Information Processing," Vol. 77 (IFIP) (B. Gilchrist, Ed.), pp. 155–160, North-Holland, Amsterdam, 1977.
16. S. MORAN, M. SCHALOM, AND S. ZAKS, An algorithm for distributed leader finding in bidirectional rings without common sense of direction (draft), 1985.
17. J. PACHL, E. KORACH, AND D. ROTEM, A technique for proving lower-bounds for distributed maximum-finding algorithms, *in* "Proceedings, 14th ACM Sympos. Theory of Computing, 1982," pp. 378–382. Revised version, "Lowerbounds for Distributed Maximum-Finding Algorithms," in *J. Assoc. Comput. Mach.* **31** (1984), 905–918.
18. G. L. PETERSON, An $O(n \log n)$ unidirectional algorithm for the circular extrema problem, *ACM Trans. Program. Lang. Syst.* **4** (1982), 758–762.
19. N. SANTORO, E. KORACH, AND D. ROTEM, Decentralized extrema-finding in circular configurations of processor: An improved algorithm, *Congr. Numer.* **34** (1982), 401–412.
20. J. VAN LEEUWEN AND R. B. TAN, An improved upperbound for distributed election in bidirectional rings of processors, *Distr. Comput.* **2** (1987), 149–160.
21. P. M. B. VITÁNYI, Distributed elections in an archimedian ring of processors, *in* "Proceedings, 16th ACM Sympos. Theory of Computing, 1984," pp. 542–547.