NEW UPPERBOUNDS FOR DECENTRALIZED EXTREMA-FINDING IN A RING OF PROCESSORS

H.L. Bodlaender and J. van Leeuwen

RUU-CS-85-15

May 1985

Rijksuniversiteit Utrecht

Vakgroep informatica

Budapestlaan 6 3584 CD Utrecht
Corr. adres: Postbus 80.012 3508 TA Utrecht
Telefoon 030-531454
The Netherlands

NEW UPPERBOUNDS FOR DECENTRALIZED EXTREMA-FINDING IN A RING OF PROCESSORS

H.L. Bodlaender and J. van Leeuwen

Technical Report RUU-CS-85-15

May 1985

Department of Computer Science
University of Utrecht
P.O.Box 80.012, 3508 TA Utrecht
the Netherlands

NEW UPPERBOUNDS FOR DECENTRALIZED EXTREMA-FINDING IN A RING OF PROCESSORS

H.L. Bodlaender* and J. van Leeuwen

Department of Computer Science, University of Utrecht
P.O.Box 80.012, 3508 TA Utrecht, the Netherlands

Abstract. We show that decentralized extrema-finding ("election") is more efficient in bidirectional rings than in unidirectional rings of processors, by exhibiting a (non-probabilistic) algorithm for distributed extrema-finding in bidirectional rings that requires fewer messages on the average than any such algorithms for unidirectional rings. Previously only an efficient probabilistic algorithm of the same characteristic was known. Both algorithms are shown to require an average (c.q. expected) number of less than $\frac{3}{4}nH_n$ messages for rings of n processors, where H_n denotes the n^{th} harmonic number. For both algorithms the bound is improved to about $0.7nH_n$ messages.

Keywords and phrases: distributed algorithms, ring topology, election, extrema-finding, order statistics, message complexity.

1. Introduction.

Consider n processors connected in a network, and distinguished by unique identification numbers. Every processor only has local information about the network topology, viz. it only knows the processors to which it is connected through a direct link. In a number of distributed algorithms it is required that the active processors elect a central coordinator (a "leader"), e.g. as part of an initialisation or restart procedure. The problem arises to design a

---

* The work of this author was supported by the Foundation for Computer Science (SION) of the Netherlands Organisation for the Advancement of Pure Research (ZWO).

protocol by means of which any active processor can incite the election and every processor will learn the identification number of the leader in as small a number of message-exchanges as possible. Because the active processor with the largest identification number is normally designated as the leader, the election problem is also known as the "decentralized extrema-finding" problem. We assume no faults in the communication subsystem, and only consider the problem for a ring of processors.

The decentralized extrema-finding problem for rings of n processors has received considerable attention, after it was proposed by LeLann [16] in 1977. The problem has been studied for unidirectional rings as well as for general, bidirectional rings. Figures 1 and 2 summarize the solutions presently known for both cases, together with the worst case or average number of messages requires for each algorithm. In 1981 Korach, Rotem, and Santoro [15] gave a probabilistic algorithm for decentralized extrema-finding in bidirectional rings that uses a smaller (expected) average number of messages than any deterministic algorithm for the problem in unidirectional rings requires. In this paper we consider the key question of whether decentralized extrema-finding can be solved more efficiently in bidirectional rings than in unidirectional rings by a deterministic algorithm. (The question was first posed by Pachl, Korach, and Rotem [17], who proved a lowerbound of nH_n on the average number of messages

Algorithm	Lowerbound	Average	Worst Case
LeLann (1977)		n^2	n^2
Chang & Roberts (1979)		nH_n	$0.5n^2$
Peterson (1982)			1.44..nlogn
Dolev, Klawe, & Rodeh (1982)			1.356 nlogn
Pachl, Korach, & Rotem (1982)	(aver.) nH_n		

Fig. 1 Election Algorithms for Unidirectional Rings, and known General Bounds ($H_n \approx 0.69 \log n$).

Algorithm	Lowerbound	Average	Worst Case
Gallager et.al. (1979)			5nlogn
Hirschberg & Sinclair (1980)			8nlogn
Burns (1980)	$\frac{1}{4}$nlogn		3nlogn
Franklin (1982)			2nlogn
Korach, Rotem, & Santoro (1981)		(prob.)$\frac{3}{4}nH_n$	(prob.) $\frac{1}{2}n^2$
Pachl, Korach, & Rotem (1982)	(aver.) $\frac{1}{8}$nlogn		
Santoro, Korach, & Rotem (1982)			1.89nlogn
this paper (1985)		(det.)$<\frac{3}{4}nH_n$	(det.)$\frac{1}{4}n^2$

Fig. 2 Election Algorithms for Bidirectional Rings, and known General Bounds ($H_n \approx 0.69$ logn).

required by any reasonable algorithm for leader-finding in unidirectional rings.)

Consider a ring of n processors with identification numbers X_1 through X_n. Without loss of generality we may assume each X_i to be an integer between 1 and n, hence $X \equiv X_1 X_2 \ldots X_n$ is a permutation. We also assume that f is "random", i.e., we assume that every permutation can occur with an equal probability of $\frac{1}{n!}$. One technique of decentralized extrema-finding in bidirectional rings makes use of the "peaks" in a circular permutation. Assume that initially all processors are active.

Definition. A peak in a ring of active and non-active processors is an active processor X_i that is larger that the active processors immediatey to the left and to the right of X_i, assuming a fixed clock-wise orientation of the ring.

A typical algorithm due to Franklin [10] operates in the following way. During one stage of the algorithm all active processors X_i send their identification number to the nearest active processors to the left and to the right. (Intermediate, inactive processors simply relay messages onwards.) When an active processor finds out that it has a larger active "neighbour" to the left or to the right, it becomes non-active. It is clear that in one stage only 2n messages need to be exchanged, and that precisely the peaks of the current permutation pass on to the next stage. As the number of peaks is not larger than half the number of currently active processors, Franklin's algorithm requires at most logn stages and (hence) 2nlogn messages*. The experimentally observed, smaller number of messages on the average in Franklin's algorithm might be explained as follows.

Theorem (Bienaymé [2], 1874). The average number of peaks and troughs in a permutation of n elements is $\frac{1}{3}(2n-1)$.

It follows that one stage of Franklin's algorithm will leave about $\frac{1}{3}n$ processors ("peaks") active on the average. Assuming that the order type of the resulting configuration is again random, repetition shows that Franklin's algorithm requires only $\log_3 n$ stages and hence $2n\log_3 n$ $\approx$ 1.26nlogn messages on the average.

In another technique of decentralized extrema-finding, identification numbers are send on the ring in some direction and travel until a processor is encountered that "knows" that the passing identification number cannot be the largest. In a typical algorithm due to Chang and Roberts [5] identification numbers are all send in the same direction and are annihilated by the first larger processor that is encountered. Thus all identification numbers except the largest are annihilated on their way around the ring, and the "leader" is identified as the only processor that eventually receives its own identification number as a message again. Knowing it is elected, the leader will send its identification number around the ring in another n messages to inform the processors of the result.

*All logarithms are taken to the base 2, unless stated otherwise.

Given a random sequence (e.g. a time-series), an "upper record" is any element that is larger that all the preceeding ones. The study of records was pioneered by Chandler [4] in 1952, as part of the general theory of "order statistics" (see e.g. Galambos [11], Sect. 6.3). Let X be a random sequence. Let $v_o=1$, and let v_i be the index of the first upper record with index larger than v_{i-1} ($i\geq 1$). Thus v_i is a random variable for the position of the i^{th} upper record in the sequence. It is well known that the distribution of each v_i does not depend on the distribution function of the elements of the random sequence (cf. Galambos [11], lemma 6.3.1) and that we may assume in fact that the elements are uniformly distributed. Observe that v_1 is the distance to the 'first' upper record of the sequence. The following result repeatedly occurs in the theory (see e.g. [4], [9], [13]).

Theorem A. The average distance to the first upper record in a random sequence of length n is $H_n-1 \approx 0.69\log n$.

Proof. (sketch).

One can show that $P(v_1=j) = \frac{1}{j(j-1)}$ ($j\geq 2$). Thus the average distance to v_1 is equal to

$$\sum_{j=2}^{n} \frac{j-1}{j(j-1)} = \sum_{j=2}^{n} \frac{1}{j} = H_n-1 \approx 0.69\log n. \quad \square$$

The theory of record distributions in random sequences was considerably advanced by Renyi [19] in 1962. He also derived the following useful fact, proved later by David and Barton [6] (p.181) by a combinatorial argument.

Theorem B. For every $k\geq 1$ and $1<j_1<\ldots<j_k$ one has that $P(v_1=j_1;\ldots;v_k=j_k) = \frac{1}{j_k \prod_{i=1}^{k}(j_i-1)}$.

Renyi [19] proved that the number of upper records in a random permutation of n elements has the same distribution as the number of cycles in a random permutation. It follows from results of Feller [8] from

1945 that this number is normally distributed, with expected value $H_n \approx 0.69 \log n$ (see also [19]).

The results from the theory of order statistics apply to decentralized extrema-finding by observing that e.g. in the algorithm of Chang and Roberts the message generated by an X_i is propagated to the first upper record in the random sequence $X_i X_{i+1} \dots$. (Because the message can travel all the way around the ring, the sequence is considered to have length n.) By theorem A a message will travel over H_n links "on the average", before it is annihilated. It follows that the algorithm of Chang and Roberts uses $nH_n \approx 0.69\, n\log n$ messages on the average. (This fact was proved by Chang and Roberts [5] without reference to the theory of order statistics.) By a result of Pachl, Korach, and Rotem [17] the algorithm is optimal for unidirectional rings. In this paper we will show that the algorithm is not optimal for bidirectional rings, i.e., bidirectional rings are "faster".

The paper is organised as follows. In section 2 we review a probabilistic algorithm for decentralized extrema-finding due to Korach, Rotem, and Santoro [15] and derive a deterministic algorithm for the problem that uses only $\frac{3}{4}nH_n \approx 0.52\, n\log n$ messages on the average. In Section 3 we improve the analyses to obtain a bound about $0.7nH_n \approx 0.48\, n\log n$ messages for both algorithms.

2. <u>Decentralized extrema-finding in a bidirectional ring using a small number of messages on the average</u>. We begin by describing a probabilistic algorithm for extrema-finding in a bidirectional ring due to Korach, Rotem, and Santoro [15] that uses an "expected" number of $\frac{3}{4}nH_n$ messages. We subsequently derive a deterministic algorithm for the problem that uses the same number of messages on the average (over all rings of n processors).

The probabilistic algorithm employs the second technique described in Section 1 but, instead of all X_i sending their identification number in the same direction on the ring like in Chang and Roberts' method, the processors randomly decide to send their identification number to the left or to the right. With messages going clockwise and counterclockwise on the ring, it is expected that many messages run into "larger" messages and (hence) are annihilated sooner, thus resulting in the smaller message complexity of the algorithm. The algorithm in every processor consists of three successive stages, as described below.

<u>Algorithm-P</u>

Each processor X_i keeps the largest identification number it has seen in a local variable MAX_i $(1 \le i \le n)$. Each processor X_i goes through the following stages.

<u>Stage 1</u> (initialisation)

$MAX_i := X_i$;
choose a direction $d \in \{left, right\}$ with probability $\frac{1}{2}$;
send message $\langle X_i \rangle$ in direction d on the ring;

<u>Stage 2</u> (election)

repeat the following steps, until the end of the election is signalled by receipt of a <!> message:
if two messages are received from the left and the right simultaneously, then ignore the smaller message and proceed as if only the larger message is received;
if message $\langle X_j \rangle$ is received from a neighbour, then

    <u>if</u> $X_j > MAX_i$ <u>then</u> $MAX_i := X_j$;
                    pass messages $\langle X_j \rangle$ on
    <u>elif</u> $X_j = MAX_i$ <u>then</u> {X_i has won the election}
                    send message <!> on the ring
    <u>fi</u>;

<u>Stage 3</u> (inauguration)

if a message <!> is received, the election is over and MAX_i holds the identification number of the leader;
if this processor was elected in stage 2 then the inauguration is over, otherwise pass message <!> on and stop.

One easily verifies that a processor X_i wins the election if and only if its identification number succeeds in making a full round along the ring in a direction chosen in stage 1. Thus, at the moment that a unique processor X_i finds out that it is the leader, all processors must have set their local MAX-variable to X_i. It follows that it is sufficient to send a simple <!> message around the ring for an inauguration and as a signal that the election is over and that the algorithm is correct. We assume that all processors start the election simultaneously, otherwise the first message a processor receives serves to wake it up and trigger its stage 1, before it actually processes the message. For the analyses we will assume that the processors work synchronously.

<u>Theorem</u> 2.1 (Korach, Rotem, and Santoro [15]).

(i) Algorithm-P uses $\approx \frac{1}{2}n^2$ messages in the worst case,

(ii) Algorithm-P uses (at most) $\approx \frac{3}{4}nH_n \approx$ o.52 nlogn messages in the expected case.

<u>Proof</u>.

(i) The worst case occurs in a ring X=n n-1...2 1, when all processors decide to send their identification numbers to the right (as in the algorithm of Chang and Roberts [5]). The number of messages adds up to $\frac{1}{2}n(n-1)+n \approx \frac{1}{2}n^2$.

(ii) Observe that the message generated by X_i (in stage 1) will be annihilated by the first upper record in the chosen direction on the ring. If the first upper record had decided to send its identification number in the opposite direction, i.e., towards X_i, then the messages meet "half way" and the $\langle X_i\rangle$-message is killed right there. There is probability $\frac{1}{2}$ that the $\langle X_i\rangle$-message needs to travel only half the distance to the first upper record in either direction on the ring. Using theorem A, the expected number of $\langle X_i\rangle$-messages will be $\frac{1}{2}H_n+\frac{1}{2}.\frac{1}{2}H_n = \frac{3}{4}H_n$. (In case the first upper record decides to send its identification number away from X_i, it is possible that the second upper record decides to send its identification number towards X_i. If this happens it will kill the message of the first upper record, and it can conceivably stop the $\langle X_i\rangle$-message even before reaching the position of the first upper record. Thus the expected number of messages will be slightly less than $\frac{3}{4}H_n$ per processor, cf. Section 3.) It follows that the total number of messages exchanged is less than $\frac{3}{4}nH_n+n \approx \frac{3}{4}nH_n \approx$ 0.52 nlogn in the expected case. □

Observe that Algorithm-P is probabilistic and, hence, no proof in itself that decentralized extrema-finding is more efficient for bidirectional rings than for unidirectional rings. To resolve the problem we devise a version of Algorithm-P in which stage 1 is replaced by a purely deterministic step. The idea is to let a processor X_i send its $\langle X_i \rangle$-message in the direction of its smallest neighbour, instead of letting it decide the initial direction by random choice. If X_i is beaten by a neighbour rightaway in the first exchange, it is made "inactive" for the remainder of the election.

Algorithm-D

Similar to Algorithm-P, except that for each processor X_i stage 1 is replaced by the following stage.

Stage 1*

send message $\langle *X_i \rangle$ to both neighbours on the ring;
wait for the messages $\langle *X_{i-1} \rangle$ and $\langle *X_{i+1} \rangle$ of both neighbours (with the indices "i-1" and "i+1" interpreted in the usual circular sense as indices of the left and right neighbour, resp.);

```
MAX_i := max(X_{i-1}, X_i, X_{i+1});
if MAX_i = X_i then
      if X_{i-1} < X_{i+1} then send messages <X_i> to the left
                           else send message <X_i> to the right
      fi
fi;
```

(Stages 2 and 3 are unchanged.)

Algorithm-D is correct by the same argument as used for Algorithm-P. Note that in Algorithm-D, stage 1* uses only 2n messages and eliminates at least $\frac{1}{2}n$ processors from active participation in the election. The active processors that remain and send an $\langle X_i \rangle$-message on the ring, will always have an inactive neighbour to the left and to the right.

Theorem 2.2.

(i) Algorithm-D uses $\approx \frac{1}{4}n^2$ messages in the worst case,

(ii) Algorithm-D uses (at most) $\approx \frac{3}{4}nH_n \approx 0.52$ nlogn messages in the average case.

Proof.

(i) At most $\frac{1}{2}n$ processors are still active after stage 1*, and the active processor are separated by at least one inactive processor. Suppose the largest processor sends its identification number to the right. The worst case occurs if every second processor sends its identification number in the same direction and is not annihilated before it reaches the largest. This generates at most $n + \sum_{i=1}^{n/2} (n-2i) \approx \frac{1}{4}n^2$ messages. The worst case occurs in a ring of the form X≡n 1 n-1 $\lceil\frac{1}{2}n\rceil$ n-2 $\lceil\frac{1}{2}n\rceil$-1... (the shuffle of n n-1 ... $\lceil\frac{1}{2}n\rceil$+1 and 1 $\lceil\frac{1}{2}n\rceil$ $\lceil\frac{1}{2}n\rceil$-1 ... 2).

(ii) Note that stage 1* only requires 2n messages and leaves at most $\frac{1}{2}n$ processors (peaks) that will send a message on the ring at the end of the stage. To allow for an analysis bases on random sequences, we note that this is only an optimized version of the algorithm in which every processor X_i sends a message on the ring in a direction as determined in stage 1*. By pairing every permutation with one in which the neighbours of X_i are interchanged, one easily sees that X_i sends its messages to the left or to the right with probability $\frac{1}{2}$ (averaged over all permutations). The message sent by X_i will be annihilated by the first upper record X_j in the direction determined in stage 1*, or by the message of the first upper record that is a peak in the same direction (in case this message was sent towards X_i and collided with the $\langle X_i\rangle$-message between X_i and X_j). We ignore the case that X_i does not have an upper record. Without loss of generality we may in fact assume that X_i and X_j are more than two steps apart, otherwise the $\langle X_i\rangle$-message certainly travels only O(1) steps. As a result we may assume that X_j sends a message towards X_i with probability $\frac{1}{2}$, where we note that the complementary case with probability $\frac{1}{2}$ consists of X_j

sending its message away from X_i or not sending a message at all. (This is seen by the following argument, where we use X_j^l and X_j^r to denote the left and right neighbours of X_j as seen from X_i in the direction of X_j. Note that $X_j^l < X_i$, by the assumption that X_j is the first upper record. If $X_j^r < X_i$ then pair the current permutation with the one in which X_j^l and X_j^r are switched. If $X_j^r > X_i$ then pair the current permutation with the one in which X_j and X_j^r are switched. Of every pair precisely one permutation will give a case in which the first upper record of X_i sends a message towards X_i. Note that the pairing of permutations is independent of the choice of a direction by X_i.) By theorem A we know that the average distance of a random processor to its first upper record is H_n. It follows that Algorithm-D uses (at most) $\frac{3}{4}nH_n+O(n)$ messages on the average. (One should observe that, as in the proof of theorem 2.1. (ii), the analysis ignores the possible effect of higher order upper records. Thus the average number of messages used by Algorithm-D will actually be less than the claimed $\frac{3}{4}H_n+O(1)$ messages per processor, cf. Section 3.) □

Corollary 2.3. Decentralized extrema-finding can be achieved strictly more efficiently (i.e., with fewer messages on the average) for bidirectional rings than for unidirectional rings.

Note that Algorithm-P and Algorithm-D use "time" n and n+1, respectively, when executed under ideal assumptions, not counting the time for the inauguration of an elected leader.

3. An improved analysis of Algorithm-P and Algorithm-D. In the proofs of theorem 2.1 and 2.2 it was argued that the bound of $\frac{3}{4}H_n$ on the average (c.q. expected) number of propagations of an $\langle X_i \rangle$-message is only an upperbound, because the possible effect of higher order upper records was ignored. In this section we will improve the analyses and derive a bound of about $0.7H_n$ on the average (c.q. expected) number of propagations of a message in both algorithms.

For an analysis of Algorithm-P, we assume without loss of generality that i=1 and that the $\langle X_1 \rangle$-message is sent to the right. Let

$v_1, v_2, \ldots$ be random variables denoting the position of the first and higher order upper records (cf. Section 1). If X_{v_1} to $X_{v_{j-1}}$ randomly choose to send their <X>-message to the right as well but X_{v_j} sends its message to the left, then the $\langle X_1\rangle$-message is annihilated by the $\langle X_{v_j}\rangle$-message if the messages meet before X_{v_1} is reached, i.e., at processor $X_{1+\lceil\frac{v_j-1}{2}\rceil}$ provided $v_j<2v_1$. (For v_j-1 odd, the messages will not meet but pass over the same link before $\langle X_1\rangle$ is annihilated at the next processor.) Otherwise the $\langle X_1\rangle$-message is simply killed at X_{v_1}.

__Definition.__ $K_n(j)= \sum\limits_{\substack{1<t_1<\ldots<t_j\leq n \\ t_j<2t_1}} \frac{t_1-\frac{1}{2}t_j}{(t_1-1)\ldots(t_j-1)t_j}$.

Suppose that we take the effect of up to l upper records into account. (Further upper records could only lower the bound on the expected message complexity.)

__Theorem__ 3.1 The expected number of messages used by Algorithm-P is bounded by $n(H_n - \sum\limits_{j=1}^{l} (\frac{1}{2})^j K_n(j)) + O(n)$.

__Proof.__

The expected number of $\langle X_1\rangle$-messages propagated by Algorithm-P is bounded by the expected value of

$$\sum_{j=1}^{l} (\tfrac{1}{2})^j\left\lceil\frac{v_j-1}{2}\right\rceil + (\tfrac{1}{2})^l(v_1-1) =$$

$$= (v_1-1)-\{\sum_{j=1}^{l} (\tfrac{1}{2})^j(v_1-1)-\sum_{j=1}^{l} (\tfrac{1}{2})^j\left\lceil\frac{v_j-1}{2}\right\rceil\}$$

$$= (v_1-1)-\sum_{j=1}^{l} (\tfrac{1}{2})^j(v_1-\frac{v_j}{2})+O(1)$$

, where each term in the summation arises with the probability of v_j being less than $2v_1$ (and thus of the $\langle X_{v_j}\rangle$-message serving as the annihilator of the $\langle X_1\rangle$-message). We ignore the effect of rings without a first upper record. The expected value is given by

$$\sum_{1<t_1\le n} P(v_1=t_1)(t_1-1)-\sum_{j=1}^{l}(\tfrac{1}{2})^j \sum_{\substack{1<t_1<\ldots<t_j\le n\\ t_j<2t_1}} P(v_1=t_1;\ldots;v_j=t_j)(t_1-\tfrac{1}{2}t_j)+O(1)=$$

$$=\sum_{1<t_1\le n}\frac{t_1-1}{(t_1-1)t_1}-\sum_{j=1}^{l}(\tfrac{1}{2})^j K_n(j)+O(1)=$$

$$=H_n-\sum_{j=1}^{l}(\tfrac{1}{2})^j K_n(j)+O(1),$$

using theorem B. Accumulating this bound for all $\langle X_i\rangle$-messages yields the result. □

Note that the term $-(\frac{1}{2})^j K_n(j).n$ denotes the savings in the expected number of messages used by the algorithm when the effect of the j^{th} upper record is taken into account.

Lemma 3.2.

(i) $K_n(1) = \frac{1}{2}H_n + O(1)$,

(ii) $K_n(2) = (\frac{1}{2} - \frac{1}{2}\ln 2)\ H_n + O(1)$,

(iii) $K_n(3) = (\frac{1}{2} - \frac{1}{2}\ln 2 - \frac{1}{4}\ln^2 2)\ H_n + O(1)$,

(iv) $K_n(j+1) < K_n(j)$.

Proof.

(i) $K_n(1) = \sum_{1<t_1\le n}\frac{\frac{1}{2}}{(t_1-1)} = \frac{1}{2}H_n + O(1)$.

(ii) See the appendix.

(iii) See the appendix.

(iv) The result is obvious by the interpretation of $K_n(j+1)$ and $K_n(j)$, but can be proved formally as follows.

$$K_n(j+1) = \sum_{\substack{1<t_1<\ldots<t_{j+1}\leq n \\ t_{j+1}<2t_1}} \frac{t_1-\frac{1}{2}t_{j+1}}{(t_1-1)\ldots(t_{j+1}-1)t_{j+1}} =$$

$$= \sum_{1<t_1<\ldots<t_j\leq n} \left\{\frac{1}{(t_1-1)..(t_j-1)} \sum_{\substack{t_j<t_{j+1}\leq n \\ t_{j+1}<2t_1}} \frac{t_1-\frac{1}{2}t_{j+1}}{(t_{j+1}-1)t_{j+1}}\right\} \leq$$

$$\leq \sum_{\substack{1<t_1<\ldots<t_j\leq n \\ t_j<2t_1}} \left\{\frac{1}{(t_1-1)..(t_j-1)t_j} \sum_{t_{j+1}=t_j+1}^{2t_1-1} \frac{t_j\cdot(t_1-\frac{1}{2}t_{j+1})}{(t_{j+1}-1)t_{j+1}}\right\} <$$

$$< \sum_{\substack{1<t_1<\ldots<t_j\leq n \\ t_j<2t_1}} \left\{\frac{t_1-\frac{1}{2}t_j}{(t_1-1)..(t_j-1)t_j} \sum_{t=t_{j+1}}^{2t_1-1} \frac{t_j}{(t-1)t}\right\} <$$

$$< K_n(j),$$

where we use that $\sum_{t=t_j+1}^{2t_1-1} \frac{t_j}{(t-1)t}$ consists of at most t_1-2 terms of size less than $\frac{1}{t_1}$ and thus has a sum <1 (but positive). □

Theorem 3.3. The expected number of messages used by Algorithm-P is equal to $0.70\ldots nH_n + O(n)$.

Proof.

By theorem 3.1. the expected number of messages used by Algorithm-P is equal to $n(H_n - \sum_{j=1}^{L} (\frac{1}{2})^j K_n(j)) + O(n)$, for the largest L possible. (Note that no ring has a j^{th} upper record with $t_j<2t_1$ for $j\geq \frac{1}{2}n+1$.) Using lemma 3.2. this number is bounded from above by

$$(1-\tfrac{1}{2}(\tfrac{1}{2})-\tfrac{1}{4}(\tfrac{1}{2}-\tfrac{1}{2}\ln 2)-\tfrac{1}{8}(\tfrac{1}{2}-\tfrac{1}{2}\ln 2-\tfrac{1}{4}\ln^2 2))nH + O(n) =$$
$$= (\tfrac{9}{16}+\tfrac{3}{16}\ln 2+\tfrac{1}{32}\ln^2 2)nH_n + O(n) =$$
$$= 0.7075nH_n + O(n),$$

and from below by

$$(\tfrac{9}{16}+\tfrac{3}{16}\ln 2+\tfrac{1}{32}\ln^2 2)nH_n - (\tfrac{1}{2}-\tfrac{1}{2}\ln 2-\tfrac{1}{4}\ln^2 2)\sum_{j=4}^{\infty}(\tfrac{1}{2})^j.nH_n + O(n)$$
$$= (\tfrac{1}{2}+\tfrac{1}{4}\ln 2+\tfrac{1}{16}\ln^2 2)nH_n + O(n) =$$
$$= 0.7033\, nH_n + O(n). \quad \square$$

Because of the analogy to Algorithm-P (cf. theorem 2.2), it is intuitive that the improved bound of $0.70..nH_n + O(n)$ messages also holds for the average number of messages used by Algorithm-D. We give a more rigorous proof of this fact. Note that in its first stage, Algorithm-D expends $O(n)$ messages to eliminate every processor that has a larger neighbour. The active processors that remain are at least one position apart, but must be at least two positions apart if we want to claim the independence of their choice of direction at the end of stage 1* (cf. theorem 2). This motivates the definition of a modified record concept.

<u>Definition</u>. Given a random sequence, and "upper *-record" is any element that is larger than all the preceeding ones (thus an upper record in the traditional sense) for which the two immediately preceeding elements are not upper *-records.

We will bound the average number of messages used by Algorithm-D by taking only the effect of upper *-records into account, by an analysis that is very similar to the analysis of Algorithm-P. (The effect of upper records that are no upper *-records will only lower the resulting estimate.)

<u>Definition</u>. $L_n(j) = \Sigma\, P^*(v_1=t_1;\ldots;v_j=t_j).(t_1-\tfrac{1}{2}t_j)$, where the summation is taken over all $t_1,\ldots,t_j$ with $1<t_1<\ldots<t_j\leq n$ and $2<t_1$,

$t_1+2<t_2,\ldots,t_{j-1}+2<t_j$ and $t_j<2t_1$.

In the definition, $P^*(v_1=t_1;\ldots;v_j=t_j)$ denotes the probability that the i^{th} upper *-record occurs at position t_i $(1i\leq j)$. Note that for $2<t_1$, $t_1+2<t_2,\ldots,t_{j-1}+2<t_j$ one has $P^*(v_1=t_1;\ldots;v_j=t_j) \geq P(v_1=t_1;\ldots;v_j=t_j) = \frac{1}{(t_1-1)\ldots(t_j-1)t_j}$, because the upper records in positions t_1 through t_j will automatically be upper *-records. Suppose we only take the effect of 1 upper *-records into account. In analogy to theorem 3.1. one obtains the following fact.

<u>Theorem</u> 3.4. The average number of messages used by Algorithm-D is bounded by (at most) $n(H_n - \sum_{j=1}^{1} (\frac{1}{2})^j L_n(j)) + O(n)$.

For simplicity define $L_n'(j) = \sum \frac{t_1-\frac{1}{2}t_j}{t_1-1)\ldots(t_j-1)t_j}$, where the summation extends over all $t_1,\ldots,t_j$ as in $L_n(j)$.

<u>Lemma</u> 3.5.

(i) $L_n(j) \geq L_n'(j)$,

(ii) $0\leq K_n(j) - L_n'(j) = O(j)$.

<u>Proof</u>.

(i) This follows immediately from the definitions.

(ii) Obviously $K_n(j) \geq L_n'(j)$, as L_n' equals the expression for $K_n(j)$ over a more restricted range. To show that $K_n(j)-L_n'(j) = O(j)$ we define the following auxiliary quantity $M_n(j,i)$ for $0\leq i\leq j$:

$M_n(j,i) = \sum \frac{t_1-\frac{1}{2}t_j}{(t_1-1)\ldots(t_j-1)t_j}$, where the summation is taken over all $t_1,\ldots,t_j$ with $1<t_1<\ldots<t_j\leq n$ and $t_i+2<t_{i+1},\ldots,t_{j-1}+2<t_j$ and $t_j<2t_1$.

Observe that $K_n(j) = M_n(j,j)$ and $L_n'(j)=M_n(j,0)$, and clearly $M_n(j,i+1)\geq M_n(j,i)$. Note that $\frac{t_1-\frac{1}{2}t_j}{t_j} \leq 1$, and that for "fixed" t_1 to t_i one has $\sum \frac{t_1-\frac{1}{2}t_j}{(t_1-1)\ldots(t_j-1)(t_{i+1}-1)\ldots(t_j-1)t_j} = O\left(\frac{1}{(t_1-1)\ldots(t_i-1)}\right)$, where the summation extends over all $1<t_{i+1}<\ldots<t_j\leq n$ with $t_i+2<t_{i+1},\ldots,t_{j-1}+2<t_j$ and $t_j<2t_1$. Next observe that

$$M_n(j,i+1)-M(j,i)=\Sigma\frac{t_1-\frac{1}{2}t_j}{(t_1-1)\ldots(t_j-1)t_j}$$

where the summation is taken over all $t_1,\ldots,t_j$ with $1<t_1<\ldots<t_j\leq n$ and $t_{i+1}+2<t_{i+2},\ldots,t_{j-1}+2<t_j$ and $t_j<2t_1$, and $t_{i+1}=t_i+1$ or $t_{i+1}=t_i+2$. It follows that

$$M_n(j,i+1)-M(j,i)=O\Bigg(\sum_{\substack{1<t_1<\ldots<t_{i+1}\leq n\\ t_{i+1}<2t_1\\ t_{i+1}=t_i+1\text{ or }=t_i+2}}\frac{1}{(t_1-1)\ldots(t_i-1)(t_{i+1}-1)}\Bigg)=$$

$$=O\Bigg(\sum_{\substack{(1<t_1<..<t_i\leq n)\\ t_i<2t_1}}\frac{1}{(t_1-1)..(t_i-1)t_i}\Bigg)=$$

$$=O\Big(\sum_{1<t_1\leq n}\frac{1}{t_1^2}\Big)=O(1).$$

Hence $K_n(j)-L_n'(j)=M_n(j,j)-M_n(j,o)=\sum_{i=0}^{j-1}(M(j,i+1)-M(j,i))=O(j)$. □

<u>Theorem</u> 3.6. The average number of messages used by Algorithm-D is bounded by (at most) $0.7075nH_n+O(n)$.

<u>Proof</u>.

By theorem 3.4. and lemma 3.5. the average number of messages used by Algorithm-D can be bounded as follows (for any fixed l):

$$n(H_n-\sum_{j=1}^{l}(\tfrac{1}{2})^jL_n(j))+O(n)\leq n(H_n-\sum_{j=1}^{l}(\tfrac{1}{2})^jL_n'(j))+O(n)=$$

$$=n(H_n - \sum_{j=1}^{l} (\tfrac{1}{2})^j K_n(j)) + O(n. \sum_{j=1}^{l} (\tfrac{1}{2})^j j) + O(n) =$$

$$=n(H_n - \sum_{j=1}^{l} (\tfrac{1}{2})^j K_n(j)) + O(n).$$

The result now follows by using the upper bound on the latter expression, derived in the proof of theorem 3.3. □

4. References.

(Reference [1] is not cited in the text.)

[1] Barton, D.E., and C.L. Mallows, Some aspects of the random sequence, Ann. Math. Stat. 36(1965) 236-260.

[2] Bienaymé, J., Sur une question de probabilités, Bull. Soc. Math.France 2(1874) 153-154.

[3] Burns, J.E., A formal model for message passing systems, Techn. Rep. 91, Computer Sci. Dept., Indiana University, Bloomington, IN., 1980.

[4] Chandler, K.N., The distribution and frequency of record values, J. Roy. Stat. Soc., Series B, 14(1952) 220-228.

[5] Chang, E., and R. Roberts, An improved algorithm for decentralized extrema-finding in circular configurations of processes, C. ACM 22 (1979) 281-283.

[6] David, F.N., and D.E. Barton, Combinatorial chance, Charles Griffin & Comp., London, 1962.

[7] Dolev, D., M. Klawe, and M. Rodeh, An O(nlogn) unidirectional distributed algorithm for extrema finding in a circle, J. Algorithm 3 (1982) 245-260.

[8] Feller, W., The fundamental limit theorems in probability theory, Bull. Amer. Math. Soc. 51(1945) 800-832.

[9] Foster, F.G., and A. Stuart, Distribution-free tests in time-series based on the breaking of records, J. Roy. Stat. Soc., Series B, 16(1954) 1-13.

[10] Franklin, W.R., On an improved algorithm for decentralized extrema finding in circular configurations of processors,

C.ACM 25(1982) 336-337.

[11] Galambos, J., The asymptotic theory of extreme order statistics, J. Wiley & Sons, New York, 1978.

[12] Gallager, R.G., P.A. Humblet, and P.M. Spira, A distributed algorithm for minimum-weight spanning trees, ACM Trans. Prog. Lang. and Syst. 5(1983) 66-77.

[13] Haghighi-Talab, D., and C. Wright, On the distribution of records in a finite sequence of observations, with an application to a road traffic problem, J. Appl. Prob. 10(1973) 556-571.

[14] Hirschberg, D.S., and J.B. Sinclair, Decentralized extrema-finding in circular configurations of processors, C.ACM 23(1980) 627-628.

[15] Korach, E., 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, Univ. of Waterloo, Waterloo, 1981.

[16] LeLann, G., Distributed systems-towards a formal approach, in: B. Gilchrist (ed.), Information Processing 77 (IFIP), North- Holland Publ. Comp., Amsterdam, 1977, pp. 155-160.

[17] Pachl, J., E. Korach, and D. Rotem, A technique for proving lower bounds for distributed maximum-finding algorithms, Proc. 14th ACM Symp. Theory of Computing, 1982, pp.378-382.

[18] Peterson, G.L., An O(nlogn) unidirectional algorithm for the circular extrema problem, ACM Trans. Prog. Lang. and Syst. 4(1982) 758-762.

[19] Renyi, A., Egy megfigyeléssorozat kiemelkedö elemeiröl (On the extreme elements of observations), MTA III. Oszt. Közl. 12(1962) 105-121.

[20] Santoro, N., E. Korach, and D. Rotem, Decentralized extrema-finding in circular configurations of processors: and improved algorithm, Congr. Numer. 34(1982) 401-412.

Appendix

We give a proof of the estimates for $K_n(2)$ and $K_n(3)$ stated in lemma 3.2. through the following auxiliary results.

Lemma A.1. $K_n(2) = (\frac{1}{2}-\frac{1}{2}\ln 2)H_n + O(1)$.

Proof.

We use the following estimate: if $f(x)$ is non-negative and decreasing on $[a,b]$, then $\sum_{t=a}^{b} f(t) = \int_a^b f(x)dx+\varepsilon f(a)$ for some $0\leq\varepsilon\leq 1$.

Taking $f(x) = \frac{t_1-\frac{1}{2}x}{(x-1)x}$ we obtain

$$\sum_{t_2=t_1+1}^{2t_1-1} \frac{t_1-\frac{1}{2}x}{(x-1)x} = \int_{t_1+1}^{2t_1-1} \frac{t_1-\frac{1}{2}x}{(x-1)x}\,dx + O(\frac{1}{t_1}) =$$

$$= \left((t_1-\tfrac{1}{2})\ln(x-1)-t_1\ln x\right)\Big/_{x=t_1+1}^{x=2t_1-1} +O(\frac{1}{t_1}) =$$

$$= \tfrac{1}{2}-\tfrac{1}{2}\ln 2+O(\frac{1}{t_1}).$$

Now observe from the definition of $K_n(2)$ that

$$K_n(2)= \sum_{\substack{1<t_1<t_2\leq n\\ t_2<2t_1}} \frac{t_1-\frac{1}{2}t_2}{(t_1-1)(t_2-1)t_2} = \sum_{t_1=2}^{n-1}\ \sum_{t_2=t_1+1}^{\min(n,2t_1-1)} \frac{t_1-\frac{1}{2}t_2}{(t_1-1)(t_2-1)t_2}$$

and (thus)

$$\sum_{t_1=2}^{\frac{1}{2}n}\ \sum_{t_2=t_1+1}^{2t_1-1} \frac{t_1-\frac{1}{2}t_2}{(t_1-1)(t_2-1)t_2} \leq K_n(2) \leq \sum_{t_1=2}^{n-1}\ \sum_{t_2=t_1+1}^{2t_1-1} \frac{t_1-\frac{1}{2}t_2}{(t_1-1)(t_2-1)t_2}.$$

By substituting the previous estimate we obtain

$$(\tfrac{1}{2}-\tfrac{1}{2}\ln 2) \sum_{t_1=2}^{\frac{1}{2}n} \frac{1}{t_1-1}+O(\sum_{t_1=2}^{\frac{1}{2}n} \frac{1}{t_1^2}) \leq K_n(2) \leq (\tfrac{1}{2}-\tfrac{1}{2}\ln 2) \sum_{t_1=2}^{n-1} \frac{1}{t_1-1}+O(\sum_{t_1=2}^{n-1} \frac{1}{t_1^2})$$

, hence $K_n(2) = (\frac{1}{2}-\frac{1}{2}\ln 2)\, H_n + O(1)$. □

To derive an estimate for $K_n(3)$ we follow a similar approach, although the analysis becomes slightly more involved.

Lemma A.2. For $t_1 \le u < 2t_1-1$, $\sum_{t=u+1}^{2t_1-1} \frac{t_1-\frac{1}{2}x}{(t-1)t} = \frac{t_1}{u} - \frac{1}{2} - \frac{1}{2}\ln 2 - \frac{1}{2}\ \ln \frac{t_1}{u} +$ $O(\frac{1}{t_1})$.

Proof.

$$\sum_{t=u+1}^{2t_1-1} \frac{t_1-\frac{1}{2}t}{(t-1)t} = \int_{u+1}^{2t_1-1} \frac{t_1-\frac{1}{2}x}{(x-1)x}\, dx + O(\frac{1}{t_1}) =$$

$$= \left((t_1-\tfrac{1}{2})\ln(x-1)-t_1\ln x\right) \Big/_{x=u+1}^{x=2t_1-1} + O(\frac{1}{t_1}) =$$

$$= (t_1-\tfrac{1}{2})\ln(2t_1-2)-t_1\ln(2t_1-1)+t_1\ln(u+1)-(t_1-\tfrac{1}{2})\ln u+O(\frac{1}{t_1}) =$$

$$= \frac{t_1}{u} - \frac{1}{2} - \frac{1}{2}\ln 2 - \frac{1}{2}\ln \frac{t_1}{u} + O(\frac{1}{t_1}),$$

where we use that $\log(1+z) = z + O(z^2)$ for $z \le \frac{1}{2}$. □

It follows that

$$\sum_{t_1=3}^{n} \sum_{t_2=t_1+1}^{2t_1-2} \sum_{t_3=t_2+1}^{2t_1-1} \frac{t_1-\frac{1}{2}t_3}{(t_1-1)(t_2-1)(t_3-1)t_3} =$$

$$\sum_{t_1=3}^{n} \sum_{t_2=t_1+1}^{2t_1-2} \frac{\frac{t_1}{t_2}-\frac{1}{2}-\frac{1}{2}\ln 2-\frac{1}{2}\ln\frac{t_1}{t_2}}{(t_1-1)(t_2-1)} + \sum_{t_1=2}^{n} \sum_{t_2=t_1+1}^{2t_1-2} \frac{O(\frac{1}{t_1})}{(t_1-1)(t_2-1)}.$$

Note that $\sum_{t_2=t_1+1}^{2t_1-2} \frac{1}{t_2-1}$ consists of t_1-2 term of size $\le \frac{1}{t_1}$ and thus is $O(1)$. Consequently the second summand in the expression above is

$O(\sum_{t_1 \geq 2} \frac{1}{t_1^2}) = O(1)$. Rewrite the first summand as B-C, with

$$B = \sum_{t_1=3}^{n} \sum_{t_2=t_1+1}^{2t_1-2} \frac{t_1-\frac{1}{2}t_2}{(t_1-1)(t_2-1)t_2},$$

$$C = \sum_{t_1=3}^{n} \sum_{t_2=t_1+1}^{2t_1-2} \frac{\frac{1}{2}\ln 2+\frac{1}{2}\ln\frac{t_1}{t_2}}{(t_1-1)(t_2-1)}.$$

Lemma A.3. For $a \geq \frac{1}{2}\ln 2 + \frac{1}{2} \ln t_1$, $\sum_{t=t_1+1}^{2t_1-2} \frac{a-\frac{1}{2}\ln t}{(t-1)} = a\ln 2 - \frac{1}{4}\ln^2 2 - \frac{1}{2}\ln 2 \ln t_1 + O(\frac{a}{t_1})$.

Proof.

$$\sum_{t=t_1+1}^{2t_1-2} \frac{a-\frac{1}{2}\ln t}{(t-1)} = \sum_{t=t_1+1}^{2t_1-2} \frac{a-\frac{1}{2}\ln t}{t} + \sum_{t=t_1+1}^{2t_1-2} \frac{a-\frac{1}{2}\ln t}{t(t-1)} =$$

$$= \int_{t_1+1}^{2t_1-2} \frac{a-\frac{1}{2}\ln x}{x}\, dx + O(\frac{a}{t_1}) =$$

$$= (a\ln x-\frac{1}{4}\ln^2 x)\Big/_{x=t_1+1}^{x=2t_1-2} + O(\frac{a}{t_1}) =$$

$$= a\ln 2-\frac{1}{4}\ln^2 2-\frac{1}{2}\ln 2\ln t_1 + O(\frac{a}{t_1}). \quad \square$$

Lemma A.4. $K_n(3) = (\frac{1}{2} - \frac{1}{2} \ln 2 - \frac{1}{4} \ln^2 2)H_n + O(1)$.

Proof.

By the analysis in the proof of lemma A.1 it easily follows that we have

$$B = \sum_{t_1=3}^{n} \frac{\frac{1}{2}-\frac{1}{2}\ln 2+O(\frac{1}{t_1})}{(t_1-1)} = (\tfrac{1}{2}-\tfrac{1}{2}\ln 2)H_n + O(1).$$

Applying lemma A.3. with $a=\frac{1}{2}\ln 2 + \frac{1}{2}\ln t_1$ we obtain

$$C = \sum_{t_1=3}^{n} \frac{\frac{1}{4}\ln^2 2 + O(\frac{\ln t_1}{t_1})}{(t_1-1)} = \tfrac{1}{4}\ln^2 2\, H_n + O(1),$$

and thus $\displaystyle\sum_{t_1=3}^{n} \sum_{t_2=t_1+1}^{2t_1-2} \sum_{t_3=t_2+1}^{2t_1-1} \frac{t_1-\frac{1}{2}t_3}{(t_1-1)(t_2-1)(t_3-1)t_3} = B-C+O(1) =$

$= (\frac{1}{2} - \frac{1}{2}\ln 2 - \frac{1}{4}\ln^2 2)H_n + O(1)$. As in the proof of lemma A.1 one can now estimate $K_n(3)$ by this expression from above, and also from below using $\frac{1}{2}n$ instead of n. Thus $K_n(3) = (\frac{1}{2} - \frac{1}{2}\ln 2 - \frac{1}{4}\ln^2 2)\, H_n+O(1)$. □