A Probabilistic Algorithm for Decentralized Extrema-Finding in a Circular Configuration of Processors

by

E. Korach*, D. Rotem** and N. Santoro***

RESEARCH REPORT CS-81-19

University of Waterloo
Department of Computer Science
Waterloo, Ontario, Canada

May 1981

* University of Waterloo
Department of Combinatorics and Optimization
Waterloo, Ontario, Canada

** University of Waterloo
Department of Computer Science
Waterloo, Ontario, Canada

*** University of Ottawa
Department of Computer Science
Ottawa, Ontario, Canada

A Probabilistic Algorithm for Decentralized Extrema-Finding in a Circular Configuration of Processors

by

E. Korach*, D. Rotem** and N. Santoro***

Abbreviated Title: Extrema Finding in Circular Configurations

* University of Waterloo
Department of Combinatorics and Optimization
Waterloo, Ontario, Canada

** University of Waterloo
Department of Computer Science
Waterloo, Ontario, Canada

*** University of Ottawa
Department of Computer Science
Ottawa, Ontario, Canada

Abstract

In this note we present and analyze a probabilistic algorithm for finding all k highest numbered processors arranged in a circular configuration. The algorithm requires $O(n(\log_e n)^k)$ message transmissions on the average and may be completed in n time units. This generalizes the algorithms of Chang [2] and Hirschberg [4] which find only one unique highest numbered processor.

Key words and phrases: Decentralized Algorithms, Distributed System, Operating Systems, Records in Permutations.

1. Introduction

In this paper we consider a system of n processors arranged in a circular configuration. Each processor is given a unique number (of which it alone is aware), and we would like to devise election algorithms which designate by concensus a set of one or more unique processors. Situations in which such algorithms are important are presented in Lelann [5] and Chang and Roberts [2].

The efficiency of these algorithms can be measured by the total number of messages exchanged between the processors and also by the total time required to complete the election process. The time is measured under the assumption that there is maximum overlap in the message transmission. The algorithm of Chang and Roberts, which we call Algorithm-C, finds the highest numbered processor in the circle in n time units and requires the transmission of nH_n ($H_n \sim \log_e n$ is the n-th harmonic number) messages on the average with a worst case of $n(n+1)/2$ transmitted messages. Recently, Hirschberg and Sinclair [4] found an algorithm which requires $8(n + \lceil n \log_2 n \rceil)$ messages in its worst case. However this new algorithm requires between $4n - 2$ and $6n - 6$ units of time to complete the election process. The number of message transmissions by Hirschberg and Sinclair has been slightly improved by Burns [1] who also showed that $O(n \log_2 n)$ is a lower bound for this problem.

In Section 2 we propose a new solution to the extrema-finding problem which uses a probabilistic method. This algorithm, called Algorithm-P, has a lower number of messages transmitted on the average compared with Algorithm-C, while its running time is still n .

In Section 3 we propose an extension to Algorithm-C which enables us to select the k unique processors with the k highest labels. This algorithm is analyzed using the theory of records in permutations [3] and when $k = 1$ we get a new proof of the nH_n result given in [2] for the average case analysis of Algorithm-C.

2. The Algorithm

For reference purposes we include here a description of Algorithm-C. The assumptions made in all the above mentioned algorithms are that each processor can send messages only to its neighbors in the circle and that it can distinguish between the clockwise and anticlockwise directions.

Algorithm-C

```
Step 1:  Initially each processor $i$ , $1 \le i \le n$ , sends the message $i$
         to its clockwise neighbor.
Step 2:  When processor $i$ receives the message $j$ then;
         a)  if $j > i$ then $i$ sends the message $j$ to its clockwise
             neighbor.
         b)  if $j = i$ then $i$ wins the election since it is the
             highest numbered processor.
         c)  if $j < i$ the message $j$ is discarded.                □
```

The correctness of this algorithm follows immediately from the fact that only the highest message transmitted will return back to its originator.

The time required to complete the election process when maximum overlap of transmissions is allowed will be n .

Next, we present our algorithm for this problem,

Algorithm-P

Assume that a number p where $0 < p < 1$ is known apriori to all processors on the circle. Also, each processor i stores the maximum number it has seen so far in MAX_i . Each processor i , $1 \leq i \leq n$ performs the following.

Step 0: Set $i \to MAX_i$.

Step 1: Initially processor i chooses randomly a direction $d(i) \in \{\text{clockwise, anticlockwise}\}$ where $\text{prob}\{d(i) = \text{clockwise}\} = p$, and then sends the message i in the chosen direction.

Step 2: If processor i receives a message j from one of its neighbors then if $j > MAX_i$ the message j is sent to the other neighbor of i and $MAX_i \leftarrow j$ else if $j = MAX_i$, processor i wins the election.

Step 3: If processor i receives two messages, say k and j , from both its neighbors at one time then it ignores the smaller message, $\min(k, j)$, and proceeds as in Step 2. □

Again, correctness and finiteness of this algorithm follow from the fact that once a direction $d(i)$ is chosen for message i , this message will continue to traverse the circle in this direction. The message i will return to its originator if and only if it is the highest number on the circle. Clearly, the worst and best case for the number of transmitted messages and also the running time is as in Algorithm-C.

The following definitions are useful in the analysis of the average number of messages transmitted by Algorithm-P. Let $\Pi = \langle \sigma_1, \sigma_2, \ldots, \sigma_n \rangle$ be a permutation on $\{1, 2, \ldots, n\}$. The ordered set of <u>records</u> of Π, $R(\Pi)$, is defined as

$$R(\Pi) = \langle \sigma_i \in \Pi \mid i > 1 \text{ and for all } j < i , \sigma_j < \sigma_i \rangle .$$

The set $R(\Pi) \cup \sigma_1$ is also called left to right maxima in Π ([6]). Let D be a permutation of $1, 2, \ldots, n$ written on a circle. We denote by $\vec{D}(i)$ the permutation obtained by reading the elements of D in the clockwise direction starting with element i and $\overleftarrow{D}(i)$ is defined similarly where the reading is performed anticlockwise (see Fig. 1). For an element j in the permutation $\vec{D}(i)$ $(\overleftarrow{D}(i))$ we denote by $\vec{\Delta}(i, j)$ $(\overleftarrow{\Delta}(i, j))$ the distance of j from i, i.e. $\vec{\Delta}(i, j)$ is one less than the position of j in $\vec{D}(i)$.

In Algorithm-P, a processor i may be eliminated from the election process by processor j where $j > i$ in two ways:

a) the message i arrives at processor j and it is discarded.

b) the message i arrives at processor ℓ, $\ell < i$ and is found to be smaller than $MAX_\ell = j$ or is ignored because it arrives at ℓ at the same time as j.

In both cases processor j is called an <u>eliminator</u> of i. In a given circular arrangement D, let $\vec{E}_D(i)$ be the sequence of all possible eliminators of i if $d(i) =$ clockwise, and $\overleftarrow{E}_D(i)$ the sequence of possible eliminators of i if $d(i) =$ anticlockwise (see Fig. 1). Clearly, the first records in $\vec{D}(i)$ and $\overleftarrow{D}(i)$ which we call r_1 and s_1 respectively satisfy $r_1 \in \vec{E}_D(i)$ and $s_1 \in \overleftarrow{E}_D(i)$. For the purpose of the following analysis we will assume that the processors work synchronously and that all processors start the election process at the same time.

<u>Lemma 1</u>. Let $R(\vec{D}(i))$ and $R(\overleftarrow{D}(i))$ be the sequence of records of $\vec{D}(i)$ and $\overleftarrow{D}(i)$ respectively. For $j > i$,

$$\text{(a)}\quad r_j \in \vec{E}_D(i) \leftrightarrow \left\{ r_j \in R(\vec{D}(i)) \text{ and } \left\lceil \frac{\vec{\Delta}(i, r_j)}{2} \right\rceil^{\dagger} < \vec{\Delta}(i, r_1) \right\}$$

$$\text{(b)}\quad s_j \in \overleftarrow{E}_D(i) \leftrightarrow \left\{ s_j \in R(\overleftarrow{D}(i)) \text{ and } \left\lceil \frac{\overleftarrow{\Delta}(i, s_j)}{2} \right\rceil < \overleftarrow{\Delta}(i, s_1) \right\}.$$

where r_k and s_k are the k-th records of $\vec{D}(i)$ and $\overleftarrow{D}(i)$ respectively.

<u>Proof</u>:

(a) In this case the message i moves clockwise. If $i = n$ then it has no possible eliminators and $R(\vec{D}(i))$ is empty so the lemma holds. Otherwise, the message i can travel at most the distance $\vec{\Delta}(i, r_1)$. Clearly, a processor r_j may eliminate i if and only if

(i) $d(r_j) =$ anticlockwise

(ii) $\overleftarrow{\Delta}(r_j, r_1) < \vec{\Delta}(i, r_1) + 2$ which is equivalent to

$$\left\lceil \frac{\vec{\Delta}(i, r_j)}{2} \right\rceil < \vec{\Delta}(i, r_1)$$

† $\lceil x \rceil$ is the ceiling of x, i.e. the smallest integer larger than x.

(iii) the message r_j is not eliminated by any other processor on its way which is equivalent to $r_j \in R(\vec{D}(i))$.

Part (a) follows from (i), (ii) and (iii).

Part (b) may be proved in a similar way. $\square$

Theorem 1. The average number of messages transmitted during the execution of Algorithm-P is smaller than $\frac{3}{4}n(H_n + \frac{2}{3} - \frac{1}{3n})$ for $n > 3$ and $p = \frac{1}{2}$.

Proof: Given a pair of circular configurations, D and its reverse D^R (see Fig. 1(b)), the total distance travelled by message i , $(i \neq n)$ under Algorithm-C in this pair is

$$\vec{\Delta}(i, r_1) + \overleftarrow{\Delta}(i, s_1) .$$

Turning now to Algorithm-P, let $\vec{E}_D(i) = \langle r_1, \ldots, r_\ell \rangle$ and $\overleftarrow{E}_D(i) = \langle s_1, \ldots, s_k \rangle$. The probability that i will be eliminated by $r_t \in \vec{E}_D(i)$ is $(1-p)\cdot p^t$ and the distance travelled in this case by meassage i is $\left\lceil \frac{\vec{\Delta}(i, r_t)}{2} \right\rceil$, also, i will travel the distance $\vec{\Delta}(i, r_1)$ with probability $p^{\ell+1}$. We obtain the following expression for the average distance travelled by message i in D .

$$\begin{aligned} A = \; & (1 - p) \sum_{j=1}^{\ell} p^j \left\lceil \frac{\vec{\Delta}(i, r_j)}{2} \right\rceil + p^{\ell+1}\vec{\Delta}(i, r_1) \\ & + p \sum_{j=1}^{k} (1 - p)^j \left\lceil \frac{\overleftarrow{\Delta}(i, s_j)}{2} \right\rceil + (1 - p)^{k+1}\overleftarrow{\Delta}(i, s_1) \end{aligned} \tag{1}$$

Similarly in D^R the average distance is,

$$B = p\sum_{j=1}^{\ell}(1-p)^j\left\lceil\frac{\vec{\Delta}(i, r_j)}{2}\right\rceil + (1-p)^{\ell+1}\vec{\Delta}(i, r_1)$$

$$+ (1-p)\sum_{j=1}^{k}p^j\left\lceil\frac{\overleftarrow{\Delta}(i, s_j)}{2}\right\rceil + p^{k+1}\overleftarrow{\Delta}(i, s_1)$$

By setting $p = \frac{1}{2}$ and noting that the the sum $A + B$ may only increase if we ignore the possible eliminators $r_2, \ldots, r_\ell$ and $s_2, \ldots, s_k$ if follows that,

$$A + B \le \frac{1}{2}(\vec{\Delta}(i, r_1) + \overleftarrow{\Delta}(i, s_1)) + \frac{1}{2}\left(\left\lceil\frac{\vec{\Delta}(i, r_1)}{2}\right\rceil + \left\lceil\frac{\overleftarrow{\Delta}(i, s_1)}{2}\right\rceil\right)$$

$$\le \frac{3}{4}(\vec{\Delta}(i, r_1) + \overleftarrow{\Delta}(i, s_1)) + \frac{1}{2} \quad . \tag{2}$$

The above analysis holds for every message i , $i < n$, and for every pair D and D^R . The message n always travels a distance of n , hence the average number of messages transmitted under Algorithm-P where each of the $(n-1)!$ circular configurations is equally probable is less than

$$\frac{3}{4}(nH_n - n) + \frac{n-1}{4} + n = \frac{3}{4}n(H_n + \frac{2}{3} - \frac{1}{3n}) \quad . \tag{3}$$

□

3. Finding the k largest numbers

In this section we analyze a modification to Algorithm-C for the case where all k processors with highest labels must be identified, we call the new algorithm Algorithm-C_k.

Algorithm-C_k

```
Step $1_k$: Each processor j sends to its clockwise neighbor a message
            which consists of a pair of numbers (j, c) where c is a
            counter which is initially set to zero.
Step $2_k$: When processor i receives the message (j, c)
            a) If $j > i$ then i sends (j, c) to its clockwise neighbor.
            b) If $j = i$ then i is the c+1-st highest numbered processor.
            c) If $j < i$ then
                   $c \leftarrow c + 1$
                   if $c = k$ the message is discarded otherwise
                   the message (j, c) is sent to the clockwise neighbor of i.
```

□

The correctness of this algorithm follows from the fact that exactly the messages which originated from one of the k highest numbered processors will return back to their originators. We now give a bound on the average number of message transmissions in this case.

Clearly all messages which are one of the k largest will be transmitted n times in any circular configuration. We therefore consider only messages i where $i \leq n-k$. Given a circular configuration D with $R(D(i)) = \langle r_1, r_2, \ldots, r_s \rangle$, the message i will travel a distance of at most $\vec{\Delta}(i, r_k)$ before it is discarded. We can now use the following result given in [3].

Lemma 2. Let $A_n(k)$ be the average position of the k-th record in a random permutation on $\{1, 2, \ldots, n\}$ and let $p_n(k)$ be the probability that such a permutation contains exactly k records. Then for k fixed and $n \to \infty$

$$A_n(k) \sim \frac{1}{k!}(\log_e n)^k$$

$$p_n(k) \sim \frac{1}{N\cdot k!}(\log_e n)^k .$$

$\square$

Remark 1: The contribution to $A_n(k)$ of permutations which do not have a k-th record is taken as zero.

Theorem 2. For k fixed, the average number of messages transmitted under Algorithm-C_k is asymptotically bounded by G where

$$G = O(n(\log_e n)^k) .$$

Proof: We calculate an asymptotic bound on the total number of messages transmitted over all $(n-1)!$ possible circular configurations as follows. For each fixed configuration D we consider the n permutations $\vec{D}(1), \vec{D}(2), \ldots, \vec{D}(n)$ which can be derived from it. We note that the distance travelled by message i in D is at most one less than the position of the k-th record in $\vec{D}(i)$ if this record exists. In case $\vec{D}(i)$ does not have a k-th record, the distance travelled by i is bounded by n. By repeating the above argument for all $(n-1)!$ possible circular configurations and using Lemma 2 it follows that the total number of messages transmitted is bounded by

$$n!\, A_n(k) + n!\, k\cdot p_n(k-1)\cdot n . \qquad (4)$$

The first term in (4) bounds the total number of messages in these permutations which have a k-th record while the second term bounds the total number of messages in all the remaining permutations. By dividing (4) by $(n-1)!$ we get an upper bound G on the average number of messages transmitted where

$$G = \frac{n(\log_e n)^k}{k!} + \frac{n \cdot k(\log_e n)^{k-1}}{(k-1)!} = O(n(\log_e n)^k) \quad . \tag{5}$$

□

The exact result nH_n of [2] is derived by noting that exactly $(n-1)!$ permutations on $\{1, 2, \ldots, n\}$ do not have a first record. Hence the total number of messages transmitted

$$n!(A_n(1)-1) + (n+1)(n-1)! \tag{6}$$

the second term is a correction introduced because of Remark 1. The average is obtained by dividing (6) by $(n-1)!$ and the known result ([3]) that $A_n(1) = H_{n-1}$.

We can transform Algorithm-P into Algorithm-P_k by adding a counter c to each message in a similar way to the transformation of Algorithm-C into Algorithm-C_k . In this case all k highest numbered processors will be identified.

It is easy to show that for every k $(k < n)$, the performance of Algorithm-P_k is never worse than that of Algorithm-C_k . In fact we can prove the following: Let AC_k and AP_k be the average number of messages transmitted under Algorithm-C_k and Algorithm-P_k respectively.

Lemma 3. For fixed k , $p = \frac{1}{2}$ and $n \to \infty$,

$$\frac{AP_k}{AC_k} \leq 1 - (\tfrac{1}{2})^{k+1} \tag{7}$$

Proof: For a fixed configuration D and a message $1 \le i \le n-k$, let $\langle b_1, b_2, \ldots, b_k \rangle$ be the k numbers which cause the counter c to be incremented if $d(i) =$ anticlockwise. Note that in this case not all the b_i's (a_i's) are necessarily records in $\overrightarrow{D}(i)(\overleftarrow{D}(i))$. We now repeat the same argument as in Theorem 1. In the configuration D and its reverse D^R, the message i will travel a distance of $\overrightarrow{\Delta}(i, b_k) + \overleftarrow{\Delta}(i, a_k)$ under Algorithm-C_k. In Algorithm-P_k the average distance for this pair of configurations is bounded by

$$2\left\{\frac{1}{2}\left[\left(\frac{1}{2}\right)^k \left\lceil\frac{\overrightarrow{\Delta}(i, b_k)}{2}\right\rceil + \left(1 - \left(\frac{1}{2}\right)^k\right)\overrightarrow{\Delta}(i, b_k)\right]\right. \\ \left. + \frac{1}{2}\left[\left(\frac{1}{2}\right)^k \left\lceil\frac{\overleftarrow{\Delta}(i, a_k)}{2}\right\rceil + \left(1 - \left(\frac{1}{2}\right)^k\right)\overleftarrow{\Delta}(i, a_k)\right]\right\} . \quad (8)$$

from which the result follows since AC_k is at least $n \log_e n$.

$\square$

4. Summary

The analysis of Algorithm-P has been carried under the assumption that the processors work synchronously. However, Algorithm-P will perform better than Algorithm-C also in the asynchronous case since there is always a positive probability that message transmissions will be saved. Furthermore, the bounds derived on the performance of Algorithm-P are conservative, it is an open questions whether the bounds given in (3) and (7) can be improved.

Recently, Dolev and Rodeh [7] have devised a unidirectional algorithm for finding the largest numbered processor which uses $2n\log_2 n$ messages in its worst case. Again, their algorithm has a running time which is worse than n. It is an open problem whether algorithms with running time of n must have a quadratic number of message transmissions in their worst case.

The following table summarizes the known algorithms for the extrema finding problem in circular configurations of processors.

	Name	Number of Messages		Time
		Average Case	Worst Case	
1	LeLlan's Algorithm [5]	n^2	n^2	n
2	Algorithm-C [2]	nH_n	$\binom{n+1}{2}$	n
3	Hirschberg's Algorithm [4]	?	$8n \log_2 n$	between $4n-2$ and $6n-6$
4	Algorithm-P ($p = \frac{1}{2}$)	$\le \frac{3}{4} nH_n$	$\binom{n+1}{2}$	n
5	Algorithm-C_k	$\le G=O(n(\log_e n)^k)$	$\binom{n}{2}-\binom{k}{2}+kn$	n
6	Algorithm-P_k ($p = \frac{1}{2}$)	$\le(1-(\frac{1}{2})^{k+1})AC_k$	$\binom{n}{2}-\binom{k}{2}+kn$	n

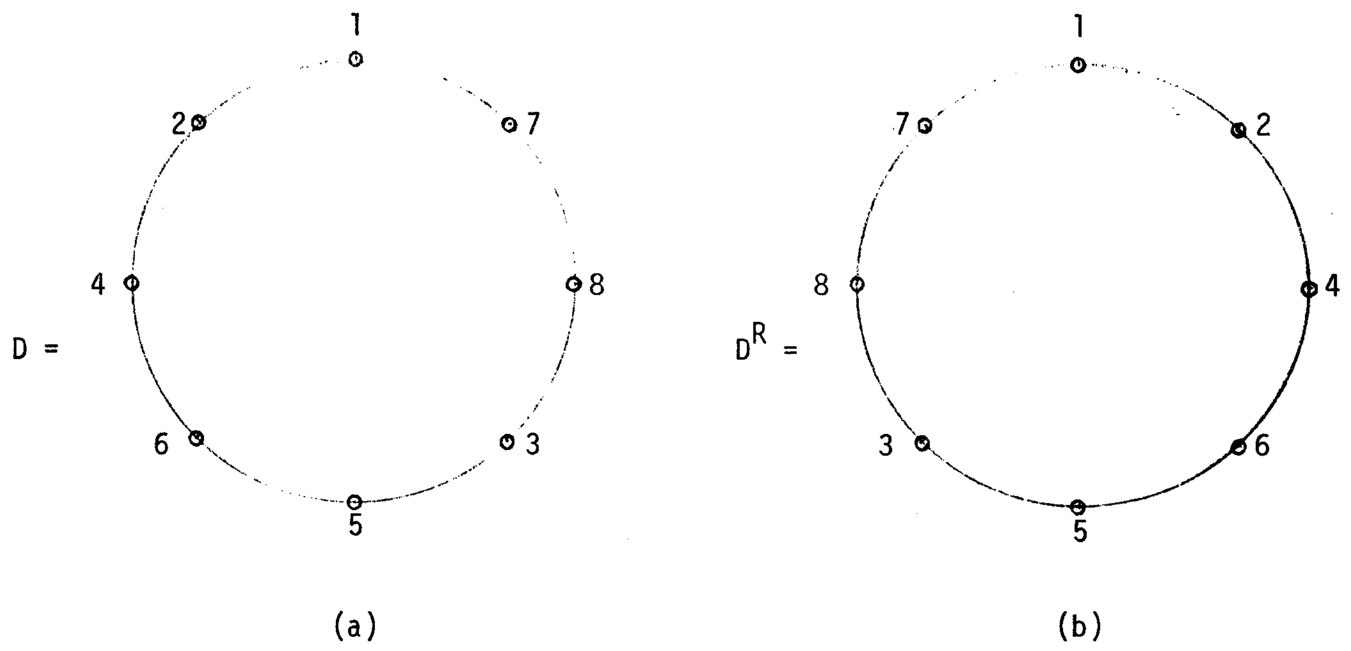

Figure 1

$\overrightarrow{D}(1) = \langle 1,7,8,3,5,6,4,2 \rangle$; $\overleftarrow{D}(3) = \langle 3,8,7,1,2,4,6,5 \rangle$;

$R(\overrightarrow{D}(3)) = \langle 5,6,7,8 \rangle$; $R(\overleftarrow{D}(3)) = \langle 8 \rangle$;

$\overrightarrow{E}_D(6) = \langle 7,8 \rangle$; $\overleftarrow{E}(5) = \langle 8 \rangle$.

References

[1] J.E. Burns, A formal model for message passing systems. Tech. Rep. No. 91, Computer Science Department, Indiana University, September 1980.

[2] E. Chang and R. Roberts, An improved algorithm for decentralized extrema-finding in circular configuration of processors. Comm. ACM, 22, 5 (May 1979), pp. 281-283.

[3] F.N. David and D.E. Barton, Combinatorial Chance. Charles Griffin & Co. Ltd., 1962.

[4] D.S. Hirschberg, and J.B. Sinclair, Decentralized extrema finding in circular configuration of processors. CACM, vol. 23, Number 11, 1980, pp. 627-628.

[5] G. LeLann, Distributed Systems - Towards a Formal Approach. Information Processing 77, North Holland Pub. Co., Amsterdam, pp. 155-160.

[6] D.E. Knuth, The Art of Computer Programming, Vol. 1, Addison-Wesley 1968, pp. 175-176.

[7] M. Rodeh, Private communication. May 1981.