Acta Univ. Sapientiae, Mathematica, **5**, 1 (2013) 54–82

Leader election in synchronous networks

Antal Iványi
Faculty of Informatics
Eötvös Loránd University
Budapest, Hungary
email: tony@inf.elte.hu

Dedicated to the memory of my friend Professor Antal Bege

Abstract. Worst, best and average number of messages and running time of leader election algorithms of different distributed systems are analyzed. Among others the known characterizations of the expected number of messages for LCR algorithm and of the worst number of messages of Hirschberg-Sinclair algorithm are improved.

1 Introduction

We consider the problem of leader election in synchronous networks [11, 16, 30, 43, 59, 92]. The networks are modeled by directed graphs, the processors are called processes and are modeled as an automaton (see e.g. [11, 59]). In the case of the deterministic algorithms it is supposed that the processes have a *unique identifier* (UID).

The main topic of this paper is the presentation of leader election algorithms of different synchronous networks and their performance features.

It is known that if the processes are indistinguishable then there is no deterministic algorithm to solve the problem. For such *anonymous* or *symmetric* networks random algorithms are proposed by Itai and Rodeh [38, 39], by Ghaffni et al. [31], and by Kalpathi et al. [42].

Lower and upper bounds for the number of necessary messages or necessary bits are presented by Afek and Gafni, Attiya et al., Bodlaender, Frederickson and Lynch, Korach et al., and Loui et al. [1, 2, 8, 9, 26, 47, 58].

2010 Mathematics Subject Classification: 68Q25, 68W10, 68W40
Key words and phrases: leader election, synchronous networks, analysis of algorithms, LCR, HS

The structure of the paper is as follows. After the introductory Section 1 in Section 2 the enumeration of some distributed systems is presented, then in Section 3 simple (as complete, chain, mesh and star networks), ring (unidirectional and bidirectional), special (such as De Bruijn, hypercube, Cayley, tree and recursively scalable networks) and general networks are analyzed.

2 Enumeration of labeled directed networks

Leader election requires that any process can inform any other process on its own data (e.g. on its own uid). In order to guarantee the participation of all processes we suppose that the investigated networks are strongly connected. It is worth to remark that there are also algorithms not requiring the strong connectedness, but these algorithms have also such output that the leader election is not solvable.

In this section we deal at first with the influence of the requirement of strong connectedness on the number of the tested networks, then with some simple networks such as complete network, star and chain.

2.1 Enumeration of connected and strongly connected networks

Let $D(n)$, $C(n)$, and $S(n)$ denote the number of labeled simple, labeled simple weakly connected and labeled simple strongly connected digraphs, respectively.

The known simple formula

$$D(n) = 2^{n(n-1)} \tag{1}$$

gives $D(n)$. The values of $D(n)$, further $C(n)/D(n)$ and $S(n)/D(n)$ are shown in Table 1 for $n = 1, \ldots, 15$. Values of $D(n)$ for $n = 1, \ldots, 35$ can be found in [65].

In 2012 Critzer [17] proposed the following method to determine the number $C(n)$ of the simple labeled weakly connected digraphs:

$$C(n) = D(n) - \frac{1}{n}\sum_{i=1}^{n-1} k\binom{n}{k}C(k)D(n-k). \tag{2}$$

Using (1) one can compute the $D(n)$ values necessary to get the values of $C(n)$ from (2). E.g. (1) results $D(1) = 1$ and then (2) gives $C(1) = 1$. In a similar way $D(2) = 4$ and $C(2) = 3$, further $D(3) = 64$ and $C(3) = 54$.

n	$D(n)$	$C(n)/D(n)$	$S(n)/D(n)$
1	1	1.000000	1.00000
2	4	0.750000	0.25000
3	64	0.843750	0.28125
4	4096	0.936035	0.39209
5	1 048 576	0.979500	0.53890
6	1 073 741 824	0.994008	0.68431
7	4 398 046 511 104	0.998280	0.80106
8	72 057 594 037 927 936	0.999511	0.88506
9	$\sim 4.722\,366\,483 \cdot 10^{21}$	0.999863	0.93161
10	$\sim 1.237\,940\,039 \cdot 10^{27}$	0.999962	0.96132
11	$\sim 1.298\,074\,215 \cdot 10^{33}$	0.999990	0.97843
12	$\sim 5.444\,517\,871 \cdot 10^{39}$	0.999997	0.98835
13	$\sim 9.134\,385\,523 \cdot 10^{46}$	0.999999	0.99367
14	$\sim 6.129\,982\,164 \cdot 10^{54}$	0.9999998	0.99659
15	$\sim 1.645\,504\,557 \cdot 10^{60}$	0.99999994	0.99817

Table 1: Number $D(n)$ of simple labeled directed graphs and the ratios $C(n)/(D(n)$ and $S(n)/D(n)$.

Table 2 contains $C(n)$ for $n = 1, \ldots, 15$. In [66] the values for $n = 16, \ldots, 35$ can be found.

V. A. Liskovets in 1969 [52, 100] proposed the following recursive formulas to compute $S(n)$:

$$a(n) = n(n-1) - \sum_{i=1}^{n-1} \binom{n-1}{t-1} a(t), \tag{3}$$

$$\lambda_t(m) = 2^{m(m+t-1)} - \sum_{k=0}^{m-1} \binom{m}{k} \lambda_t(k) \tag{4}$$

and

$$S(n) = a(n) + \sum_{i=1} \binom{n-1}{t-1} 2^{(m-1)(m-k)} \lambda_t(n-t) S(t). \tag{5}$$

Using (3) and (4) one can compute the $a(n)$ and $\lambda(n)$ values necessary to get the values of $S(n)$ from (5).

n	$C(n)$
1	1
2	3
3	54
4	3 834
5	1 027 080
6	1 067 308 488
7	4 390 480 193 904
8	72 022 346 388 181 584
9	4 721 717 643 249 254 751 360
10	1 237 892 809 110 149 882 059 440 768
11	1 298 060 596 773 261 804 821 355 107 253 504
12	5 444 502 293 680 983 802 677 246 555 274 553 481 984
13	91 343 781 554 246 596 956 424 128 384 394 531 707 099 632 640
14	6 129 980 884 648 631 844 901 425 521 287 946 137 183 899 295 465 755 648
15	1 645 504 465 371 454 407 878 687 557 239 154 898 196 072 267 336 301 175 996 872 704

Table 2: Number $C(n)$ of simple labeled connected digraphs.

Simplifying Liskovets's method in 1971 Wright [100] proposed the following formulas. Let $n \geq 1$,

$$\eta(n) = D(n) - \sum_{i=1}^{n-1} 2^{(n-1)(n-i)}\eta(i) \tag{6}$$

and

$$S(n) = \eta_n + \sum_{i=1}^{n} \binom{n-1}{i-1} S(i)\eta_{n-i}. \tag{7}$$

According to (6) $\eta_1 = 1$, $\eta_2 = 0$, $\eta_3 = 16$, and $\eta_4 = 1536$. Using these η values from (7) we get $S(1) = 1$, $S(2) = 1$, $S(3) = 18$, and $S(4) = 1606$.

The values of $S(n)$ are in Table 3 for $n = 1, 2, \ldots, 15$. In [75] also the values for $n = 16, 17, 18$ can be found.

In 1969 Liskovets [52] proved the following theorem.

Theorem 1 (Liskovets, 1969 [52]) *If* $n \geq 1$, *then*

$$D(n) - 2(n+4)n^{(n+1)(n+1)} \leq S(n) \leq D(n) \tag{8}$$

and

$$S(n) = D(n)\left(1 - n2^{2-n} + n(2n-1)2^{2-2n}\right) + O(n^3 n^{n(n-4)}). \tag{9}$$

Proof. See (Liskovets, 1969 [52]). □

n	$S(n)$
1	1
2	1
3	18
4	1 606
5	565 080
6	734 774 776
7	3 523 091 615 568
8	63 519 209 389 664 176
9	4 400 410 978 376 102 609 280
10	1 190 433 705 317 814 685 295 399 296
11	1 270 463 864 957 828 799 318 424 676 767 488
12	5 381 067 966 826 255 132 459 611 681 511 359 329 536
13	90 765 788 839 403 090 457 244 128 951 307 413 371 883 494 400
14	6 109 064 462 821 545 704 046 426 032 465 737 763 224 760 635 732 888 576
15	1 642 494 209 200 959 152 585 925 675 993 911 516 594 334 047 201 121 102 632 675 328

Table 3: Number $S(n)$ of simple labeled strongly connected digraph.

2.2 Generation of all strongly connected graphs

Let m and n be positive integers, $V = \{V_1 \ldots, V_n\}$ be a finite set and $A = \{a_1, \ldots, a_m\}$ be a finite family of ordered pairs $(V_i, V_j) \in V \times V$ of the elements of V. Let $D = (V, A)$ be an arbitrary directed graph [78, Volume A, page 28] and $D^T = (V, A^T)$ be the *transpose* [15, page 530, Exercise 22.1-3] of D defined by

$$A^T = \{(V_i, V_j) \in V \times V \mid (V_j, V_i) \in V \times V\}. \tag{10}$$

A *directed spanning tree* T of a directed graph $D = (V, A)$ is a rooted tree that consists entirely of arcs in A, all arcs directed from parents to children in the tree, and that contains every vertex of D. A directed spanning tree of D with root vertex $V_i \in V$ is a *breadth-first spanning tree* provided that each vertex of D at distance d from V_i appears at depth d in the tree (that is, at distance d from V_i in the tree) [59].

We enumerated the strongly connected networks. The base of the enumeration is the fact that the strong components of a directed graph D and its transpose D^T contain the same strongly connected components [15]. Therefore we choose arbitrary vertex as a root and build a BST (breath-first spanning tree) of the given D and of its transpose D^T. D is strongly connected if and only if both deep search trees contain all vertices of D.

Lemma 1 (Cormen et al., 1969 [15]) *A directed graph* D *is strongly connected*

if and only if its arbitrary vertex (*e.g.* V_1) *is the root of a breadth-first spanning tree of* D *and also the root of the breadth-first search tree* D^T.

Proof. Let V_a and V_b be arbitrary vertices of a strongly connected graph D. Then D contains a directed path $V_a = V_{i_1}, \ldots, V_{i_p} = V_b$ and also a directed path $V_b = V_{j_1}, \ldots, V_{j_q} = V_a$. Therefore D^T contains the directed paths $V_a = V_{j_q}, \ldots, V_{j_1} = V_b$ and $V_b = V_{i_p}, \ldots, V_{i_1} = V_a$, therefore the given condition is necessary.

Again let V_a and V_b arbitrary vertices of D. If D contains a directed path $(V_1 = V_{i_1}, \ldots, V_{i_r} = V_a)$ and also a directed path $(V_1 = V_{j_1}, \ldots, V_{j_q} = V_b)$, further D^T contains directed paths $(V_1 = V_{k_1}, \ldots, V_{k_r} = V_a)$ and $(V_1 = V_{l_1}, \ldots, V_{l_s} = V_b)$, then D contains directed paths $(V_a = V_{k_r}, \ldots, V_{k_1} = V_1 = V_{i_1}, \ldots, V_{i_r} = V_b)$ and $(V_b = V_{l_s}, \ldots, V_{l_1} = V_1 = V_{i_1}, \ldots, V_{i_r} = V_a)$, therefore the given condition is sufficient. □

Algorithm STRONG is based on Lemma 1. It decides if a given directed graph D is strongly connected.

Input parameters are: $n > 1$: the number of processes; $B = (b_1, \ldots, b_{n^2})$: the adjacency matrix of the current graph as a vector.

Output parameter is L: if D is strongly connected then $L = 1$, otherwise $L = 0$.

Working parameters are i (current number of the vertices); j, k: cycle variables; m: the current number of vertices in the tree; $Q = (Q_1, \ldots, Q_n)$: a queue for the waiting vertices; $h(Q) = h$: the *head* index of the queue; $t(Q) = t$: the *tail* index of the queue; $p = (p_1, \ldots, p_n)$: the presence vector of the vertices ($p_i = 1$, if V_i is in the tree, and $p_i = 0$ otherwise).

```
STRONG(n, B)
01   p_1 = 1                                  // line 01–08: Initialization.
02   m = 1
03   h = 1
04   Q_1 = 1
05   t = 2
06   L = 1
07   for j = 2 to n
08       p_j = 0                              // V_j is not in the tree.
09   while t > h                              // line 09–30: Test of D.
10       u = Q_h
11       for j = 1 to n − 1
12           for k = 1 to j − 1               // line 12–20: Before the main diagonal.
```

```
13                  if b_{(u−1)n+k} == 1 and p_{(u−1)n+k} == 0
                                                  // line 13: V_j not in tree
14                    p_k = 1 // line 14–15: A new vertex of the tree is found.
15                    m = m + 1
16                  if m == n
17                    return L
18                  Q_t = j
19                  t = t + 1
20                  h = h + 1
21                for k = j + 1 to n − 1 // line 21–30: After the main diagonal.
22                  if b_{(u−1)n+k} == 1 and p_{(u−1)n+k} == 0
                                                  // line 22: V_j not in tree.
23                    p_k = 1
24                    m = m + 1
25                    return L
26                  Q_t = j
27                  t = t + 1
28                  h = h + 1
29 L = 0                  // line 30–31: The graph is not strongly connected.
30 return L
```

We remark that STRONG tests *only* the existence of a breadth-first spanning tree of D. The test of the existence of a breadth-first spanning tree of D^T requires similar instructions (the only difference that in lines 13 and 22 $b_{(u-1)n+k} == 1$ must be replaced by $b_{(u-1)n+k} == 0$.

The next assertion characterizes the resource requirements of STRONG.

Theorem 2 *If* $b \geq 2$, *then* STRONG *requires* $\Theta(n^2)$ *memory locations in all cases and* $O(2^{b(b-1)}n^2)$ *time units in worst case.*

Proof. The memory requirement is determined by the size of the input neighborhood matrix B, therefore the maximal memory requirement is $\Theta(n^2)$ memory locations. The time requirement of STRONG is determined by the facts that the algorithm investigates at most $2^{n(n-1)}$ graphs and constructs an $n \times n$ sized matrix for all investigated graphs. □

Algorithm ALL-STRONG enumerates the strongly connected networks for $a,\ a+1,\ \ldots,\ b$ vertices. It is also based on Lemma 1.

The *input parameters* of ALL-STRONG are $a \geq 2$ and $b \geq a$: lower and upper bound for the current size of the investigated network.

Output parameter is $S = (S(a), \ldots, S(b))$, where $S(a)$ is the number of the strongly connected networks consisting of a processes, ..., $S(b)$ is the number of the strongly connected networks consisting of b processes.

Working parameters are i (current number of the vertices) and j (both are cycle variables); $B = (b_1, \ldots, b_n)$: the adjacency matrix of the current network as a vector; b_0: help variable to stop the increasing of the adjacency vector; $Q = (Q_1, \ldots, Q_n)$: a queue for the waiting vertices; $h(Q)$: the *head* index of the queue; $t(Q) = t$: the *tail* index of the queue; L: logical variable (if the current graph is strong, then $L = 1$, otherwise $L = 0$.

ALL-STRONG(a, b)

```
01 for i = a to b                    // line 01–04: Generation of the first graph.
02     S(i) = 0                      // line 02: Initialization of the enumeration.
03     for j = 0 to i(i − 1)
04         b_j = 0
05     STRONG(i, B)                  // line 05–07: Test of D.
06     if L == 0                     // line 06–07: D is not strong.
07       go to 14
08     for j = 1 to i(i − 1)         line 08–12: Test of D^T.
09         t_j = 1 − b_j
10     STRONG(i, T)
11     if L == 0                     // line 11–12: D^T is not strong.
12       go to 14
13     S(i) = S(i) + 1               // line 14: D is strong
14     for j = i(i − 1) downto 1     // line 14–18: Generation
           of the next graph.
15         if b_j == 0
16           b_j = 1
17           for k = j + 1 to i(i − −1)
18               b_k = 0
19         go to 05                  // line 19: Continue with the next graph.
20     print i, S(i)                 // line 20: Print result for the current size.
```

The next assertion characterizes the resource requirements of ALL-STRONG.

Theorem 3 *If* $b \geq 2$, *then* ALL-STRONG *requires* $\Theta(b(b-1))$ *memory locations in all cases and* $O(2^{b(b-1)}n^2)$ *time units in worst case.*

Proof. The memory requirement is determined by the size of the neighborhood matrices $\mathcal{B}$ and $\mathcal{T}$ defined in lines 03–04 and 08–09. The maximal size of

these matrices appears in the case when the graphs contain b vertices, therefore the maximal memory requirement is $\Theta(b(b-1))$ memory locations. The time requirement of ALL-STRONG is determined by the facts that the algorithm investigates $2^{b(b-1)}$ graphs and constructs an $n \times n$ sized matrix what according to Theorem 1 requires $O(n^2)$ time for one matrix. Multiplying these expression we get the bound $O(2^{b(b-1)}n^2)$. □

Another possible approach to generate all labeled strongly connected digraphs is to use the minimal digraphs investigated by García-López and Marijun [27].

3 Leader election

In the following sections the problem of leader election is considered. The mathematical models described in [59] are used: networks are modeled by directed (or sometimes undirected) graphs, processes by vertices. We suppose that the processors communicate and compute in synchronous rounds. The *leader election problem* is *to elect a unique leader*. Usually it is supposed that the processes are identical except for *unique identifiers* (UIDs). The size of the network is usually unknown.

In Subsection 3.1 some simple networks, then in Subsection 3.2 ring networks, in Subsection 3.3 further unidirectional networks, and finally in Subsection 3.4 further special and general networks are considered.

3.1 Leader election in simple networks

In this subsection the problem of leader election in simple networks as complete, chain, mesh and star networks is considered.

Peterson [71] in 1985, Afek and Gafni [1, 2] in 1981 and in 1985, Singh [81] in 1992 derived time and complexity bounds for mesh and complete networks.

In 1984 Korach et al. [47] proved optimal lower bounds for the number of messages in complete networks.

In 1985 Loui et al. [58] investigated the influence of the direction of the connections on the leader election algorithms.

There are known algorithms for chain [19] and star [80] networks too.

3.2 Leader election in ring networks

In this subsection comparison-based algorithms of different ring networks (in details unidirectional and bidirectional ones) are described and analyzed.

3.2.1 LCR algorithm in unidirectional ring

Figure 1 shows an unidirectional ring consisting of the processes $P_1, \ldots, P_n$.

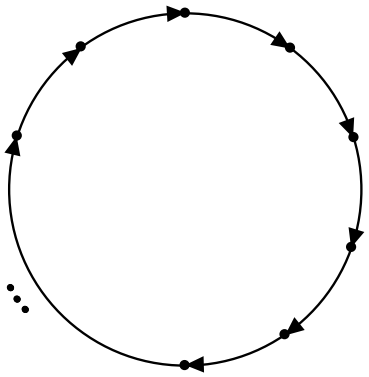

Figure 1: A ring of processes $P_1, \ldots, P_n$.

The first known leader election algorithm was proposed by Le Lann [51] in 1977 for unidirectional rings. It is a very simple algorithm. In the first step each process sends its UID to its clockwise neighbor. In the further steps each process compares the received UID with its own UID, and if they are equal, then the process declares itself the leader, otherwise sends the larger UID to the clockwise neighbor. The algorithm terminates when the process having the largest UID gets back its own UID.

This algorithm requires n steps and n^2 messages.

Chang and Roberts in 1979 [13] proposed an improved version of the previous algorithm: after the comparison of the received and own UID the processes send a message only if the received UID is the larger one. We give a formal description [59] of this algorithm called usually LCR (after Le **L**ann, **C**hang and **R**oberts) algorithm. It is supposed that the UID's are the natural numbers $1, 2, \ldots, n$.

Input parameter is n: the number of processes and $p = p_1, \ldots, p_n$: a permutation of the UID's.

Output parameter is $M_n = M$: the number of messages.

The message alphabet is $\{1, 2, \ldots, n\}$. For each i ($1 \leq i \leq n$) the state $state_i$ consists of three components:

- u, a UID, initially the UID of P_i;
- $send_i$, a UID or `null`, initially the UID of P_i;

- $\texttt{status}_i$, having possible values {unknown,leader}.

The state of P_i consists of the single state defined by the given initial values. The message generation function $\texttt{msgs}_i$ is defined by

- send the current value of send to P_i.

We remark that indices are interpreted everywhere mod n.

The transition function $\texttt{trans}_i$ is defined by the following pseudocode used in [59]:

```
send := null
if the incoming message is v, then
    case
        v > u: send := v
        v = u: status_i := leader
        v < u: do nothing
    endcase
```

Since LCR is a basic algorithm of leader election and since we execute the simulation of LCR on a sequential processor, the algorithm is described also using the pseudocode of [15, 40].

Input parameters are $n > 1$: the number of processes; $p = p_1, \ldots, p_n$: a permutation of the UID's.

Output parameters are L: the index of the elected leader; M: the number of messages.

Working parameters are $m = (m_1, \ldots, m_n)$, where m_i is the current message of P_i; i cycle variable.

```
LCR(n, p)
01 P_i in parallel for i = 1 to n                  // line 01–05: Initialization.
02     read p_i
03     m_i = i
04     s_i = 0
05 M = n
06 while all states s_i == 0                       // line 06–13: Election.
07         P_i in parallel for i = 1 to n
08             if m_{i-1} > p_i
09                 m_i = m_{i-1}
10                 M = M + 1
11             if m_{i-1} == p_i
```

```
12              s_i = m_{i-1}
13              L = i
14 return L, M                    // line 14: Return of the result.
```

Let X_n be a random variable characterizing the number of messages of LCR and let M_n be the expected value of X_n at the uniform distribution of the permutations of the UID's.

Chang and Roberts in [13] not only improved the algorithm of Le Lann, but also determined M_n.

Theorem 4 (Chang, Roberts, 1979 [13]) *If the permutations of the UID's have uniform distribution, then*

$$M_n = n + \sum_{i=1}^{n-1} \sum_{k=1}^{n-1} kP(n;i,k) = n + \sum_{k=1}^{n-1} \frac{n}{k+1} = O(n \log n), \tag{11}$$

and

$$M_n = nH_n = O(n \log n), \tag{12}$$

where H_n *is the* n*th harmonic number and* $P(n;i,k)$ *is the probability that the message* i *is passed* k *times.*

Proof. See [13]. □

$P(i,k,n)$ is the probability that the $k-1$ clockwise neighbors of i are less than i and the kth clockwise neighbor of i is larger than i. There are $i-1$ processes less than i and $n-i$ processes larger than i.

Since the place of the UID i can be fixed, the remaining identifiers can be permuted in $(n-1)!$ manner. The small UID's can be choosen in $(i-1)\cdots(i-k+1)$ manner, the kth large UID in $n-i$ manner, and the remaining UID's $(n-k)\cdots 1$ manner. So we get

$$P(n;i,k) = \frac{[(i-1)\cdots(i-k+1)](n-i)[(n-k)\cdots 1]}{(n-1)(n-2)\cdots 1}. \tag{13}$$

Using the well-known bounds

$$\frac{1}{2}\lfloor \log n \rfloor < H_n < \lceil \log n \rceil \tag{14}$$

it is easy to get the stronger assertion

$$M_n = \Theta(n \log n). \tag{15}$$

Using Leonhard Euler's following lemma we prove Lemma 3 in which (18) and (19) are stronger than (12) in Theorem 4.

Lemma 2 (Euler [22]) *If* $n \geq 1$ *then*

$$H_n = \sum_{i=1}^{n} \frac{1}{i} = \ln n + \gamma + \beta_n, \tag{16}$$

where γ *is the Euler-Mascheroni constant* ($\gamma \sim 0.577\ 215\ 665$) [22, 63, 96] *and*

$$\lim_{n\to\infty} \beta_n = 0. \tag{17}$$

Proof. See Fichtengolz [24, Volume II, page 270]. □

Lemma 3 *If* $n \geq 1$, *then*

$$M_n = n \ln n + n\gamma + n\beta_n \tag{18}$$

and

$$M_n = \Theta(n \log n). \tag{19}$$

Proof. Substitution of (16) into (12) results (18) which implies (19). □

Table 4 illustrates the accuracy of the approximation of (18).

Chen [14] in 2006 published a detailed probabilistic cost analysis of LCR algorithm. Using generating functions he proved

$$M_n = \frac{2}{n-1} \sum_{i=1}^{n-1} M_i + \frac{n}{2} \quad \text{for} \quad n \geq 2 \tag{20}$$

and remarked that $M_1 = 1$.

Using (20) Chen reproved (11) and gave a more exact characterization

$$M_n = n \log n + \gamma n + O(1) \tag{21}$$

of the mean of X_n, further determined the variance of X_n as

$$V(X_n) = \left(2 - \frac{\pi^2}{6} n^2\right) + O(n \log n). \tag{22}$$

Using Euler-Maclaurin summation [21, 60, 64, 95] D. E. Knuth [46] derived the following improved version of Lemma 2.

n	$E(M_{LCR}(n))$	$n \ln n$	$n\gamma$	$n\beta_n$
1	1.00000000000	0.000000000000	0.5772156649015	0.4227843350985
2	3.00000000000	1.386294361120	1.154431329803	0.4592743090770
3	5.50000000000	3.295836866004	1.731646994705	0.4725161392911
4	8.33333333333	5.545177444480	2.308862659606	0.4792932292476
5	11.41666666667	8.047189562171	2.886078324508	0.4833987799885
6	14.70000000000	10.75055681537	3.463293989409	0.4861491952225
7	18.15000000000	13.62137104339	4.040509654311	0.4881193023021
8	21.74285714286	16.63553233344	4.617725319212	0.4895994902062
9	25.46071428571	19.77502119603	5.194940984114	0.4907521055745
10	29.28968253968	23.02585092994	5.772156649015	0.4916749607267
11	33.21865079365	26.37684800078	6.349372313917	0.4924304789518
12	37.23852813853	29.81887979746	6.926587978818	0.4930603622537
13	41.34173881674	33.34434164700	7.503803643720	0.4935935260189
14	45.52187257187	36.94680261461	8.081019308621	0.4940506486375
15	49.77343489843	40.62075301653	8.658234973523	0.4944469083788
16	54.09166389166	44.36141955584	9.235450638425	0.4947936974029
17	58.47239288489	48.16462684896	9.812666303326	0.4950997326112
18	62.91194540753	52.02669164213	10.38988196823	0.4953717971751
19	67.40705348573	55.94434060416	10.96709763313	0.4956152484385
20	71.95479314287	59.91464547108	11.54431329803	0.4958343737632

Table 4: Concrete values of the expressions in (18).

Lemma 4 (Knuth [46]) *If* $n \geq 1$ *then*

$$H_n = \sum_{i=1}^{n} \frac{1}{i} = \ln n + \gamma + \frac{1}{2n} + \frac{1}{12n^2} + \frac{1}{120n^4} - \frac{\Theta_{2,n}}{252n^6}\,, \tag{23}$$

where $0 < \Theta_{2,n} < 1$.

Proof. See [46, Page 474]. □

It is remarkable that in the *Online Encyclopedia of Integer Sequences* [83, 88] one can find further members of the series in (23). Using the ideas of the proof of Lemma 4 we get the following characterization of M_n.

Theorem 5 *If* $n \geq 1$ *then*

$$M_n = n \ln n + \gamma n + \frac{1}{2} + \frac{1}{12n^2} + \frac{1}{120n^4} + \Theta\left(\frac{1}{n^5}\right)\,. \tag{24}$$

Proof. Using different methods in 1979 Chang and Roberts, in 2006 Chen proved (12). Substitution of the right side of (23) into (11) results

$$M_n = E(n) = n \ln n + \gamma n + \frac{1}{2} - \frac{1}{12n} + \frac{1}{120n^3} - \frac{\Theta_{2,n}}{252n^5}\,, \tag{25}$$

implying (24). □

Table 5 illustrates the accuracy of the approximation of (25).

n	$E(n)$	$n \ln n$	$n\gamma + \frac{1}{2}$	$-\frac{1}{12n}$	$\frac{1}{120n^3}$	$-\frac{\Theta_{2,n}}{252n^5}$
1	1.00000	0.00000	1.07722	−0.0833333	0.0083333	−0.0022157
2	3.00000	1.38629	1.65443	−0.0416667	0.0010417	−0.0001007
3	5.50000	3.29584	2.23165	−0.0277778	0.0003086	−0.0000147
4	8.33333	5.54518	2.80886	−0.0208333	0.0001302	−0.0000036
5	11.41667	8.04719	3.38608	−0.0166667	0.0000667	−0.0000012
6	14.70000	10.75056	3.96329	−0.0138889	0.0000386	−0.0000005
7	18.15000	13.62137	4.54051	−0.0119048	0.0000243	−0.0000002
8	21.74286	16.63553	5.11773	−0.0104167	0.0000163	−0.0000001
9	25.46071	19.77502	5.69494	−0.0092593	0.0000114	−0.0000001
10	29.28968	23.02585	6.27216	−0.0083333	0.0000083	−0.0000000
11	33.21865	26.37685	6.84937	−0.0075758	0.0000063	−0.0000000
12	37.23853	29.81888	7.42659	−0.0069444	0.0000048	−0.0000000
13	41.34174	33.34434	8.00380	−0.0064103	0.0000038	−0.0000000
14	45.52187	36.94680	8.58102	−0.0059524	0.0000030	−0.0000000
15	49.77343	40.62075	9.15824	−0.0055556	0.0000025	−0.0000000
16	54.09166	44.36142	9.73545	−0.0052083	0.0000020	−0.0000000
17	58.47239	48.16463	10.31267	−0.0049020	0.0000017	−0.0000000
18	62.91195	52.02669	10.88988	−0.0046296	0.0000014	−0.0000000
19	67.40705	55.94434	11.46710	−0.0043860	0.0000012	−0.0000000
20	71.95479	59.91465	12.04431	−0.0041667	0.0000010	−0.0000000

Table 5: Concrete values of the expressions in (25).

A third possibility for the proof of (12) is the application of Pascal's next formula [68] allowing the recursive computation of the sum of the kth powers of the first n positive integers.

Theorem 6 (Kovcs [49], Pascal [68], Pólya [72], Wolfram [98]) *If* $n \geq 1$ *and*

$p \geq 1$, *then*

$$S(n,p) = \sum_{i=1}^{n} i^p = \frac{1}{p+1}\left((n+1)^{p+1} - 1 - \sum_{k=1}^{p-1}\binom{p+1}{k}S(n,k)\right). \quad (26)$$

The following Faulhaber formula [23] also allows the computation of $S(n,p)$.

Theorem 7 (Faulhaber [23], Weisstein [97]) *If* $n \geq 1$ *and* $p \geq 1$, *then*

$$S(n,p) = \frac{1}{p+1}\sum_{i=1}^{p+1}(-1)^{\delta_{i,p}}\binom{p+1}{i}B_{p+1-i}n^i, \quad (27)$$

where $\delta_{i,p}$ *is the Kronecker-delta* [87] *and* B_i *is the Bernoulli number* [84, 85].

The following double sum gives $S(n,p)$ without recursion.

Theorem 8 (Weisstein [94]) *If* $n \geq 1$ *and* $p \geq 1$, *then*

$$S(n,p) = \sum_{i=1}^{p}\sum_{j=0}^{i-1}(-1)^j(i-j)\binom{n+p-i+1}{n-i}\binom{p+1}{j}. \quad (28)$$

3.2.2 Hirschberg-Sinclair algorithm in bidirectional ring

Hirschberg and Sinclair [35] in 1980 proposed an algorithm (HS) for bidirectional rings which elects as leader also the process having the largest UID. HS requires in worst case only $\Theta(n \log n)$ messages instead of the $\Theta(n^2)$ requirement of LCR. Figure 2 shows a bidirectional ring.

Input parameters are $n > 1$: the number of processes; $p = p_1, \ldots, p_n$: a permutation of the UID's $1, \ldots, n$.

Output parameters: i the index of the elected leader process; $N = (N_1, \ldots, N_n)$, where N_i is the number of messages, sent by process P_i; Q: the total number of sent messages.

Working parameters are $\mathcal{M}$: the message alphabet $ml = (ml_1, \ldots, ml_n)$, where ml_i is the current message of P_i to P_{i-1}; $mr = (mr_1, \ldots, mr_n)$, where mr_i is the current message of P_i to P_{i+1}; $s = (s_1, \ldots, s_n)$: status of P_i; i is a cycle variable; *null* the empty message.

The messages are triples, consisting a UID, a *flag value(in* or *out,* and a positive integer counter (*hop-count*) h. The possible values of the status of the processes are *unknown* or *leader*.

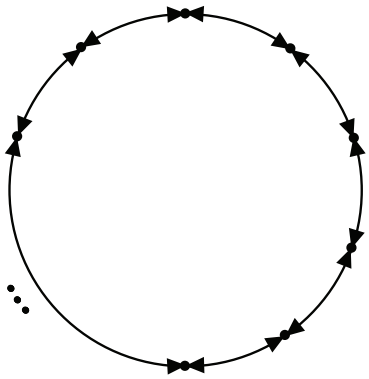

Figure 2: A bidirectional ring of n processes.

```
HS(n, p)
 01 P_i in parallel for i = 1 to n                    // line 01–05: Initialization.
 02     read p_i
 03     ml_i = (i, out, 1)                  // line 03: First message of P_i to P_{i−1}.
 04     mr_i = (i, out, 1)                  // line 04: First message of P_i to P_{i+1}.
 05     s_i = unknown              // line 05: Initialization of the first state of P_i.
 06 N = 2n                                          // line 06:Iinitialization of M.
 07 while all states are unknown                  // line 07–12: Computation of M.
 07         P_i in parallel for i = 1 to n
 08             mr_i = null
 09             ml_i = null
 10             if mr_{i−1} == (j, out, h)
 11                 if j > i and h > 1
 12                     mr_i = (j, out, h − 1)
 13                     N_i = N_i + 1
 14                 if j > i and h == 1
 15                     ml_i = (j, in, 1)
 16                     N_i = N_i + 1
 17                 if j = i
 18                     s_i = leader
 19                     Q = 0          // line 17–19: Summing numbers of messages.
 20                     for i = 1 to n
 21                         Q = Q + N_i
 22                     return i, N, Q                // line 22: Return of the results.
```

```
23          if ml_{i+1} == (j, out, h)
24            if j > i and h > 1
25            ml_i = (j, out, h − 1)
26            N_i = N_i + 1
27            if j > i and h == 1
28            mr_i = (j, in, 1)
29            N_i = N_i + 1
30            if j = i
31            s_i = leader
32            Q = 0 // line 17–19: Summing the numbers of the messages.
33            for i = 1 to n
34                Q = Q + N_i
35            return i, N, Q                    // line 17: Return of the results.
36          if ml_{i+1} == (j, in, 1) and i ≠ j
37            mr_i = (j, in, 1)
38            N_i = N_i + 1
39          if ml_{i+1} == (j, in, 1) and i ≠ j
40            ml_i = (j, in, 1)
41            N_i = N_i + 1
42          if mr_{i−1} == (i, in, 1) and ml_{i+1} == (i, in1)
43            phase = phase + 1
44            mr_i = (i, out, 2^phase
45            ml_i = (i, out, 2^phase
```

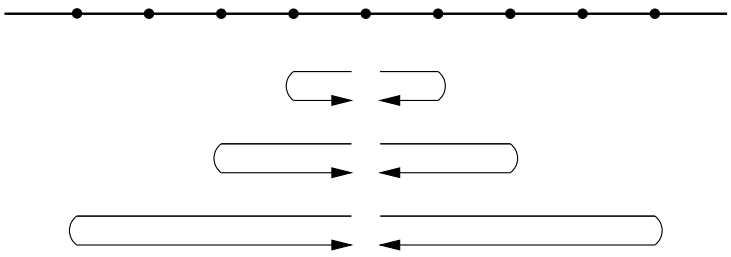

Figure 3: Paths of messages of process P_i in algorithm HS.

Hirschberg and Sinclair [35] proved the following property of their algorithm. Let W_n denote the maximal number of messages required by HS in a bidirectional synchronous ring.

Theorem 9 (Hirschberg, Sinclair [35]) *If* $n \geq 1$, *then*

$$W_n \leq 8n(\lceil \log n \rceil + 1) = \Theta(n \log n) \tag{29}$$

and

$$W_n = O(n \log n). \tag{30}$$

We proved the following, stronger assertion.

Theorem 10 *If* $n \geq 2$, *then*

$$2n\lfloor \log n \rfloor \leq W_n \leq 8n\lceil \log n \rceil \tag{31}$$

and

$$W(n) = \Theta(n \log n). \tag{32}$$

Proof. The proof follows the ideas of application of *bit reversing rings* (see [59, Example 3.6.3] and [59, Figure 3.3]. Let $n = 2^k$, for example with $k = 3$. If we choose $p_{2^0} = p_1 = n = 8$ and $p_{2^0+2^{k-1}} = p_5 = 7$, then $p_{2^0+2^{k-2}} = p_3 = 5$, $p_{2^0+2^{k-1}+2^{k-2}} = p_7 = 6$, and finally the remaining processes get the UIDS 1, 2, 3, 4, and use similar construction for larger k's then we need at least $8 \cdot 2 + 4 \cdot 2 + 2 \cdot 2 = 28$ (in general: $3{,}5n$) messages. If $2^{k-1} \leq n < 2^k$ then we suppose $n = 2^k$ processes and need at least n messages instead of $2n$. If $n = 2$ then we need only $2 \cdot 2$ (in general: $2n$) messages, therefore appears in the theorem only $2n$ as lower bound. □

Burns [12] published in 1980 a bidirectional algorithm which has a bit better worst case bound for the number of necessary messages.

3.3 Leader election is further unidirectional networks

Dolev et al. [20] and Peterson in 1982 [70] independently published an unidirectional algorithm whose worst message number is $O(n \log)n$, but their algorithm allows that the processes have arbitrary long response time that is they algorithm works only in asynchronous networks.

Rotem et al. [76] in 1987, Santoro et al. [77] in 1988 proposed an unidirectional asynchronous algorithm having $O(n \log n)$ messages in the worst case. Their algorithm elected not only the process having the largest UID, but also the processes having the k largest UID's.

Higham and Przytycka [33, 34] used a trick of Smith [89] and proposed an asynchronous algorithm what sends no more then $1.271n \log n + O(n)$ messages in worst case.

The mentioned algorithms suppose that the processes start in the same round (otherwise they can not terminate). Recently Arrieta et al. [4] elaborated an algorithm allowing different starting rounds of the processes. The price of this property is that the guarantee for the worst message number is only $O(n^2)$.

In 1996 Alimonti et al. [3] considered the problem of choosing the minimum and maximum of the UID's when equal UID's are allowed. If the size of the ring is unknown then the problem is unsolvable. The authors describe an algorith for the unidirectional ring network containing n processes, where the processes know n. The worst bit complexity (that is the number of sent bits) of their algorithm is $O((c + \log n)n)$ with arbitrary $c > 0$ and the time bound is $O(c \cdot n \cdot x^{1/c})$, where $x = \max(|u_{min}|, |u_{max}|)$.

Attiya et al. [5] in 1989, Kalamboukis et al. [41] in 1991, and Pan [67] in 1994 studied the leader election in chordal rings.

Vitányi [93] in 1984 analyzed the leader election algorithms of Archimedean rings, Kranakis and Krizane [50] in 1997 of *anonymous* (in which the processes are undistinguishable) hypercube, and Mans [61] also in 1997 of unlabeled tori

Attiya et al. [6] proved lower bounds for the necessary number of messages for anonymous ring networks.

Ingram et al. [37] proposed a leader election algorithm for dynamic asynchronous network. Ingram et al. [36] described algorithms for dynamic networks with clausal clocks. Augustin et al. [7] published a robust leader election algorithm for the fast-changing world.

3.4 Leader election in further special and general networks

Peterson [71] in 1985 described efficient algorithms for mesh networks.

In 1995 Masapati and Ural [62] proposed a linear time leader election algorithms for recursively scalable networks.

Yamashita and Kameda [101], further Kranakis and Krizanc [50] investigated algorithms in anonymous hypercube networks.

Tel in 1995 [91], Flocchini and Mans [25] in 1996 analyzed the leader election algorithms of hypercube networks.

King et al. [45] in 1989, Kim and Belford [44] in 1996 proposed algorithms for unreliable networks.

In 1997 Mans [61] described an optimal distributed algorithm for unlabeled tori.

In 2001 Gavoille [29] analyzed the leader election problem of De Bruijn networks.

In 2005 Shi and Srimani [80] described an algorithm for hierarchical star

networks.

In 2007 Srimani and Lafiti [90] proposed an algorithm for Cayley networks.

In 2008 Sepehri and Godarzi [79] described an algorithm for tree networks and using heap structure they proved that their algorithm in worst case requires only $O(n)$ messages.

Peterson [70] in 1952 described efficient algorithms for general networks.

In 1985 Afek and Gafni [2] proved that leader election in general networks requires $\Omega(n \log n)$ messages and $\Omega(\log n)$ time.

Peleg in 1990 [69] proposed a time optimal leader election algorithm for gereral networks which can be applied also for some special networks.

The basic algorithms of general networks are FloodMax and OptFloodMax (see e.g. [59]).

Das et al. [18] proposed effective algorithms which either elect a leader or signalize that the election is impossible.

Acknowledgement. The author thanks Zoltán Kása (Sapientia Hungarian University of Transylvania) for his useful critical remarks, Valery Liskovets (Mathematical Institute of Belorussian Academy of Sciences) for his help connected with the enumeration problems, István Csörgő, Attila Kovács, Sándor Kovács, and László Szili (all from Faculty of Informatics of Eötvös Loránd University) for their help connected with the sums of powers of natural numbers, PhD student Balázs Pinczel for the figures and computer experiments and PhD students Gergő Gombos and Kristóf Szabados (from the same faculty) for their technical help. The author also thanks the unknown referee for the useful remarks.

References

[1] Y. Afek, E. Gafni, Time and message bounds for election in synchronous and asynchronous complete networks, *SIAM J. Comp.,* **20** (1981), 376–394.

[2] Y. Afek, E. Gafni, Time and message bounds for election in synchronous and asynchronous complete networks, in: *Principles of Dist. Comp.,* ACM, 1985, 186–195.

[3] P. Alimonti, P. Flocchini, N. Santoro, Finding the extrema of a distributed multiset, *J. Parallel Dist. Comp.,* **37** (1996), 23–33.

[4] I. Arrieta, F. Fariña, J. R. G. de Mendl, M. Raynal, Leader election: From Higham-Przytyckas algorithm to a gracefully degrading algorithm, *Publications Internes de l'IRISA,* inria-00605799, version 1, July 2011, 9 pages.

[5] H. Attiya, J. van Leeuwen, N. Santoro, S. Zaks, Efficient elections in chordal ring networks, *Algorithmica,* **4** (1989), 437–446.

[6] H. Attiya, M. Snir, M. K. Warmuth, Computing on an anonymous ring, *J. ACM*, **35** (1988), 845–875.

[7] J. Augustine, T. Kulkarni, P. Nakhe, P. Robinson, Robust leader election in a fast-changing world, *arXive* arXiv:1310.4908v1 [cs.DC], 2013, 12 pages.

[8] H. L. Bodlaender, Some lower bound results for decentralized extrema-finding in rings of processors, *J. Comp. System Sci.,* **42** (1991), 97–118.

[9] H. L. Bodlaender, New lower bound techniques for distributed leader finding and other problems on rings of processors, *Theor. Comp. Sci.,* **81** (1991), 237–256.

[10] B. Bollobás, *Random Graphs* (Cambridge Studies in Advanced Mathematics **73**). Cambridge University Press, Cambridge, United Kingdom, 2001, XVIII + 498 pages.

[11] E. D. Burkhard, D. Kowalski, G. Malewicz, A. A. Shwartsman, Distributed algorithms, in: (ed. A. Iványi) *Algorithms of Informatics, Vol. 2*, mondAt Kiadó, Budapest, 2007, 591–642. Electronic version: AnTonCom, Budapest, 2011, http://progmat.hu/tananyagok/.

[12] J. E. Burns, A formal model for message passing systems. Tech. Rep. 91, Computer Science Dep., Indiana Univ., Bloomington, IN, May 1980, 21 pages.

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

[14] W.-M.Chen, Cost distribution of the ChangRoberts leader election algorithm, *Theoret. Comp. Sci.,* **369** (2006), 442–447.

[15] T. H. Cormen, Ch. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to Algorithms* (3rd edition), The MIT Press Hill, Cambridge/New York, 2009, 1312 pages.

[16] G. Coulouris, J. Dollimore, T. Kindberg, G. Blair, *Distributed Systems: Concepts and Design* (5th edition), Addison-Wesley, 2011, 1008 pages.

[17] G. Critzer, A recursive formula for the number of labeled simple digraphs. OEIS (ed. N. J. A. Sloane), 2012, Sequence A003027.

[18] S. Das, P. Flocchini, A. Nayak, N. Santoro, Effective elections for anonymous mobile agents, in: *Algorithms and Computation,* LNCS **4288,** 2006, 732–743.

[19] R. Dinitz, S. Moran, S. Rajsbaum, Bit complexity of breaking and acheaving symmetry in chains and rings, *J. ACM,* **55** (2008), 1–28.

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

[21] L. Euler, Methodus generalis summandi progressiones. *Commentarii Academiae Scientarum Imperialis Petropolitanae,* **6** (1738), 68–97, Euler Archiv **E025**, and *Opera Omnia,* **1** (1911), 42–72.

[22] L. Euler, De progressionibus harmonicus observationes, *Commentarii Academiae Scientarum Imperialis Petropolitanae,* **7** 1740, Euler Archiv **E043**, 150–161 and *Opera Omnia,* **1** (1911), 87–100.

[23] J. Faulhaber, *Academia Algebrae,* Johann Remelins Verlag, Ulm, 1631, 52 pages.

[24] G. M. Fichtengolz, *The Lecture on Differential and Integral Calculations, Vol. 1, 2, 3* (Russian), Nauka, Moskow, 1969, 607, 807, and 656 pages.

[25] P. Flocchini, B. Mans, Optimal elections in hypercube, *J. Parallel Dist. Comp.,* **33** 1996, 76–83.

[26] G. N. Frederickson, N. A. Lynch, Electing a leader in a synchronous ring, *J. ACM*, **34** 1987, 98–115.

[27] J. García-López, C. Marijuán, Minimal strong digraphs, *Discrete Math.* **312** (2012), 737–744. Also arXiv, arXiv:1004.4827v1 [math.CO] 27 Apr 2010.

[28] H. Garcia-Molina, Election in a distributed computing system, *IEEE Trans. Comp.,* **C-31** 1982, 48–59.

[29] C. Gavoille, Routing in distributed networks: Overview and open problems, *ACM SIGACT News,* **32** (2001), 36–52.

[30] C. Georgiou, A. A. Shvartsman, N. A. Lynch, *Cooperative Task-Oriented Computing: Algorithms and Complexity* (Synthesis Lectures on Distributed Computing Theory), Morgan & Claypool Publishers, 2011, 168 pages.

[31] M. Ghaffni, N. A. Lynch, S. Sastry, Leader election using loneliness detection, in: *Distributed Computing,* LNCS **6950**, 2011, Springer, Heidelberg, 2011, 268–282.

[32] F. Harary, Unsolved problems in the enumeration of graphs, *Magyar Tud. Akad. Mat. Kutató Int. Közl.* **5** (1960), 63–95.

[33] L. Higham, T. Przytycka, A simple, efficient algorithm for finding in rings, in: (ed. A. Schiper) *Distributed Algorithms* (7th Int. Workshp, WDAG'93, Lausanne, 1993), LNCS **725,** Springer-Verlag, Berlin, 1993, 249–263.

[34] L. Higham, T. Przytycka, A simple, efficient algorithm for maximum finding on rings, *Inf. Proc. Letters,* **58** (1996), 319–324.

[35] D. S. Hirschberg, J. B. Sinclair, Decentralized extrema-finding in circular configuration of processes, *Comm. ACM,* **23** (1980), 627–628.

[36] R. Ingram, T. Radeva, P. Shields, S. Viqar, J. E. Walter, J. L. Welch, A leader election algorithm for dynamic networks with clausal clocks, *Distrib. Computing*, **26** (2013), 75–97.

[37] R. Ingram, P. Shields, J. E. Walter, J. L. Welch, An asynchronous leader election algorithm for dynamic networks, *IEEE International Symposium on Parallel & Distributed Processing* (IPDPS 2009, Rome, 23–29 May 2009), 1–12.

[38] A. Itai, On the computational power needed to elect a leader, in: *LNCS,* **486**, Springer-Verlag, Berlin, 1991, 29–40.

[39] A. Itai, M. Rodeh, Symmetry breaking in distributed networks, *Inf. Comp.* **88** (1990), 60–87.

[40] A. Iványi, *Parallel Algorithms* (Hungarian), ELTE Eötvös Kiadó, Budapest, 2005.

[41] T. Z. Kalamboukis, S. L. Mantzaris, Towards optimal distributed election on chordal rings, *Information Proc. Letters*, **38** (1991), 265–270.

[42] M. Kalpathi, H. M. Mahmoud, M. D. Ward, Asymptotic properties of a leader election algorithm, *J. Appl. Probab.*, **48** (2011), 569–575.

[43] A. D. Kshemkalyani, M. Singhal, *Distributed Computing: Principles, Algorithms, and Systems,* Cambridge University Press, Cambridge, 2011, 756 pages.

[44] G. Kim, G. Belford, A distributed election protocol for unreliable networks, *J. Parallel Distr. Comp.*, **35** (1996), 35–42.

[45] C.-T. King, T. B. Gendreau, L. M. Ni, Reliable election in broadcast networks, *J. Parallel Distr. Computing*, **7** (1989), 521–540.

[46] D. E. Knuth, *Concrete Mathematics* (2nd edition), Addison-Wesley Publishing Co., 1994, 672 pages (1st edition: 1988).

[47] E. S. Korach, S. Moran, S. Zaks, Optimal lower bounds for some distributed algorithms for a complete network of processors, *Theoretical Comp. Sci.*, **64** (1989), 125–132.

[48] A. Kovács, Sums of the kth powers for the first twenty positive integers. Manuscript, Budapest, 2012.

[49] S. Kovács, Zur Berechnung der Potenzsummen. Manuscript. Budapest, 2013, 11 pages.

[50] E. Kranakis, D. Krizanc, Distributed computing on anonymous hypercube networks, *J. Alg.*, **23** (1997), 32–50.

[51] G. Le Lann, Distributed systems—towards a formal approach, in: (ed. B. Gilricst) *Information Processing 77* (Toronto, 1977).Vol. 7. of *Proc. of IFIP Congress,* North Holland, Amsterdam, 1977, 155–160.

[52] V. A. Liskovets, On a recurrent method for enumeration of graphs with labeled vertices (Russian), *Dokl. AN SSSR*, **184** (1969), 1284–1287.

[53] V. A. Liskovets, The number of strongly connected oriented graphs (Russian), *Mat. Zametki*, **8** (1970), 721–732.

[54] V. A. Liskovets, A contribution to the enumeration of strongly connected digraphs (Russian), *Dokl. AN BSSR,* **17** (1973), 1077–1080, 1163.

[55] V. A. Liskovets, On a general enumerative scheme for labeled graphs (Russian), *Dokl. AN BSSR*, **21** (1977), 496–499.

[56] V. A. Liskovets, Some easily derivable integer sequences. *J. Int. Sequences*, **3** (2000), Article 00.2.2.

[57] V. A. Liskovets, Exact enumeration of acyclic deterministic automata. *Discrete Appl. Math.*, **154**, (2006), 537–551.

[58] M. C. Loui, T. A. Matsushita, D. B. West, Election in a complete network with a sense of direction, *Inform. Proc. Letters*, **22** (1985), 185–187.

[59] N. A. Lynch, *Distributed Algorithms* (5th edition, The Morgan Kaufmann Series in Data Management Systems), Morgan Kaufmann Publishers, 2003, XIII + 873 pages (1st edition: 1996).

[60] C. MacLaurin, *A Treatise of Fluxions,* Vol. 1. and 2, T. W. Ruddimans and T. Ruddimans, Edinburgh, 1742, 763 pages.

[61] B. Mans, Optimal distributed algorithms in unlabeled tori and chordal rings, *J. Parallel Distributed Comp.,* **46** (1997), 80–90.

[62] G. H. Masapati, H. Ural, Electing a leader in a synchronous recursively scalable network, in *ICCI90*, LNCS **468**, Springer-Verlag, Berlin, 1990, 463–472.

[63] L. Mascheroni, Ad notationes ad calculum integralem Euleri, Vol. 1 and 2. Ticino, Italy, 1790 and 1792. Reprinted in Euler, *L. Leonhardi Euleri Opera Omnia,* Ser. 1, Vol. 12, Teubner, Leipzig, Germany, 415–542.

[64] Yu. V. Matiyasevich, Alternatives to the Euler-Maclaurin formula for calculating infinite sums, *Math. Notes*, **88** (2010), 524–529.

[65] T. D. Noe, Number of labeled weakly connected digraphs with $\mathfrak{n}$ nodes for $\mathfrak{n} = 1, \ldots, 35$. In *OEIS* (ed. N. J. A. Sloane), May 11, 2007. http://oeis.org/A053763/b053763.txt.

[66] T. D. Noe, Number of labeled simple connected digraphs with $\mathfrak{n}$ nodes for $\mathfrak{n} = 1, \ldots, 30$. In *OEIS* (ed. N. J. A. Sloane), January 9, 2009. http://oeis.org/A003027/b003027.txt.

[67] Y. Pan, A near-optimal multistage distributed algorithm for finding leaders in clustered chordal rings, *Information Sci.*, **76** (1994), 131–140.

[68] B. Pascal, *Ouvres de Blaise Pascal, Vol. 3* (ed L. Brunschvicg, P. Bourroux), Nabu Press, Charleston, SC, 2010, 341–367. 1st edition: Blaise Pascal, *Ouvres,* 1640.

[69] D. Peleg, Time-optimal leader election in general networks, *J. Parallel Distr. Comp.*, **8** (1990), 96–99.

[70] G. L. Peterson, An $O(n \log n)$ unidirectional distributed algorithm for the circular extremal problem, *ACM Trans. Lang. Systems,* **4** (1982), 758–762.

[71] G. L. Peterson, Efficient algorithms for elections in meshes and complete networks, TR 140, Dept. of Computer Science, Univ. of Rochester, 1985, 5 pages.

[72] Gy. Pólya, *Mathematical Discovery on Understanding, Learning and Teaching Problem Solving.* John Wiley & Sons, Inc, New York, NY, 1962, 216 pages.

[73] R. W. Robinson, Counting labeled acyclic digraphs, in: *New Directions in the Theory of Graphs* (F. Harary, ed.), Academic Press, New York, 1973, 239–273.

[74] R. W. Robinson, Counting unlabeled acyclic digraphs, *Combinatorial Mathematics V.* Lecture Notes in Math., **622** (1977), 28–43.

[75] R. W. Robinson, Table of n, $a(n)$ for $n = 1, \ldots, 18$, in *OEIS* (ed. N. J. A. Sloane), 2012. Sequence A003030.

[76] D. Rotem, E. Korach, N. Santoro, Analysis of a distributed algorithm for extrema finding in a ring, *J. Parallel Distr. Comp.,* **4** (1987), 575–591.

[77] N. Santoro, M. Scheutzow, J. B. Sidney, On the expected complexity of distributed selection, *J. Parallel Distr. Comp.,* **5** (1988), 194–203.

[78] A. Schrijver, *Combinatorial Optimization. Vol. A, B, C* (Algorithms and Combinatorics, Vol. **24**, Springer-Verlag, Berlin, 2003, 1800 pages.

[79] M. Seperhi, M. Godarzi, Leader election algorithm using heap structure, in: *12th WSEAS Int.l Conf. on Computers* (Heraklion, Greece, July 23-25, 2008), 2008, 668–672.

[80] W. Shi, K. Srimani, Leader election in hierarchical star network, *J. Parallel Distr. Comp.*, **65** (2005), 1435–1442.

[81] G. Singh, Leader election in complete networks, in: *Proc. of the Eleventh Annual ACM Symp. on Principles of Distributed Computing*, ACM Press, 1992, 179–190.

[82] N. J. A. Sloane, *Number of directed graphs (or digraphs) with n nodes.* OEIS (ed. N. J. A. Sloane), 2013, Sequence A000273.

[83] N. J. A. Sloane, *Number of strongly connected digraphs with* $\mathfrak{n}$ *labeled nodes,* OEIS (ed. N. J. A. Sloane), 2013, Sequence A003030.

[84] N. J. A. Sloane, *Numerator of Bernoulli number* $\mathtt{B}_\mathfrak{n}$. OEIS (ed. N. J. A. Sloane), 2013, Sequence A027641.

[85] N. J. A. Sloane, *Denominator of Bernoulli number* $\mathtt{B}_\mathfrak{n}$*.,* OEIS (ed. N. J. A. Sloane), 2013, Sequence A027642.

[86] N. J. A. Sloane, *The number of directed graphs on* $\mathfrak{n}$ *vertices,* OEIS (ed. N. J. A. Sloane), 2013, Sequence A003085.

[87] N. J. A. Sloane, *Jacobi* (*or Knonecker*) *symbol,* OEIS (ed. N. J. A. Sloane), 2013, Sequence A034947.

[88] N. J. A. Sloane, *Bernoulli Numbers* $\mathtt{B}_{2\mathfrak{n}}/2\mathfrak{n}$, in OEIS (ed. N. J. A. Sloane), 2013, Sequence A006953.

[89] A. R. Smith, Cellular automata complexity trade-offs, *Inf. Control.,* **18** (1971), 466–482.

[90] P. Srimani, S. Lafiti, Some bounded degree communication networks and optimal leader election, in: *Combinatorial Optimization in Communication Networks* (Combinatorial Optimization), **18** (2006), 467–501.

[91] G. Tel, Linear election in hypercubes, *Parallel Proc. Letters,* **5** (1995), 357–366.

[92] G. Tel, *Introduction to Distributed Systems* (Second edition), Cambridge University Press, Cambridge, 2000, 612 pages (1st edition appeared in 1984).

[93] P. Vitányi, Distributed elections in archimedean ring of processors, in *Proc. 16th Ann. ACM Symp. on Theory of Computing,* 1984, 542–547.

[94] E. W. Weisstein, *Double Sum,* From Mathworld–A Wolfram Web Resource, 2013, http://mathworld.wolfram.com/PowerSum.html.

[95] E. W. Weisstein, *Euler-Maclaurin Integration Formulas,* 2013, http://mathworld.wolfram.com/Euler-MaclaurinIntegrationFormulas.html.

[96] E. W. Weisstein, *Euler-Mascheroni Constant,* From Mathworld–A Wolfram Web Resource, 2013, http://mathworld.wolfram.com/Euler-MascheroniConstant.html.

[97] E. W. Weisstein, *Power Sum,* From Mathworld–A Wolfram Web Resource, 2013, http://mathworld.wolfram.com/PowerSum.html.

[98] S. Wolfram, *Wolframalpha,* 2013. http://www.wolframalpha.com.

[99] E. M. Wright, Asymptotic enumeration of connected graphs. *Proc. Roy. Soc. Edinburgh Sect. A*, **68** (1968/1970), 298–308.

[100] E. M. Wright, The number of strong digraphs, *Bull. London Math. Soc.,* **3** (1971), 348–350.

[101] M. Yamashita, T. Kameda, Computing on anonymous networks: Parts I and II, *IEEE Trans. Par. Dist. Syst.,* **7** (1996), 69–96.

Received: 22 March 2013