NOTE

OPTIMAL LOWER BOUNDS FOR SOME DISTRIBUTED ALGORITHMS FOR A COMPLETE NETWORK OF PROCESSORS

F. KORACH,* S. MORAN and S. ZAKS**

Department of Computer Science, Technion-Israel Institute of Technology, Haifa, Israel 32000

Communicated by M. Nivat
Received September 1987

Abstract. Lower bounds for distributed algorithms for complete networks of processors (i.e., networks where each pair of processors is connected by a communication line) are discussed. We first show an $\Omega(n \log n)$ lower bound for the number of messages required by any algorithm in a given class of distributed algorithms for such networks. This class includes algorithms for problems like finding a leader or constructing a spanning tree. We then show an $\Omega(n^2)$ lower bound for other problems, like constructing a maximal matching or a Hamiltonian circuit. In proving the lower bounds we are counting the edges which carry messages during the executions of the algorithms (ignoring the actual number of messages carried by each edge). Interestingly, this number is shown to be of the same order of magnitude as the total number of messages needed by these algorithms. The proofs of the lower bounds apply for synchronous networks and for arbitrarily long messages.

1. Introduction

Distributed computing deals with a network of processors, connected by some communication lines, that solves a certain problem. One of the basic problems studied in this area is the leader election problem, in which exactly one processor has to be chosen. This problem was originated in [10] as a token regeneration procedure. This was also the first out of many papers that studied many aspects of the election problem in circular networks. The problem was discussed for general graphs in [4]. Another special network, the complete network, also became a focus of extensive research. The subject of this paper is lower bounds on the message complexity of algorithms for this network.

The model under investigation is a network of $n$ processors with distinct identities. No processor knows any other processor’s identity. Each processor has some

* Part of this work was done while this author was at IBM Israel Scientific Center, Haifa, Israel.
** The work of this author was sponsored in part by the Foundation for Research in Electronics, Computers and Communication, administered by the Israel Academy of Sciences and Humanities, and by Technion Grant no. 121-0641.
communication lines, connecting it to some others. The processor knows the lines connected to itself, but not the identities of its neighbors. The communication is done by sending messages along the communication lines; all the processors perform the same algorithm, that includes operations of (1) sending a message to a neighbor, (2) receiving a message from a neighbor and (3) processing information in their (local) memory.

We assume that the messages on each edge arrive, with no error, in a finite time, and are kept in order in a queue until processed. We also assume that any non-empty set of processors may start the algorithm; a processor that is not a starter remains asleep until a message reaches it. Our lower bounds apply to the synchronous model, in which every message arrives after one time unit, and hence also to the asynchronous model, where there is an unbounded and unpredictable delay on the messages.

The communication network is viewed as an undirected graph $G = (V, E)$ with $|V| = n$, and we assume that the graph $G$ is connected. We refer to algorithms for a given network as algorithms acting on the underlying graph.

We address two classes of algorithms for complete graphs: the first must use edges of a spanning subgraph in every possible execution, and the second must use edges of a maximum matching in every possible execution. The problems of choosing a leader, finding the maximum identity and constructing a spanning tree clearly require algorithms that belong to the first class, while the problems of finding a complete matching or constructing a Hamiltonian cycle clearly require algorithms that belong to the second class.

In this paper the problem of proving lower bounds on the number of messages sent by distributed algorithms in the above classes is reduced to proving lower bounds on the lengths of certain sequences of edges of the corresponding network. We apply this technique to prove the following lower bounds:

1. $\Omega(n \log n)$ for the number of edges (hence messages) used by any algorithm in the first class, and
2. $\Omega(n^2)$ for the number of edges (hence messages) used by any algorithm in the second class.

For the first class, the assumption that nodes can be awakened spontaneously at any time is crucial. In fact, if all nodes that spontaneously start the algorithm are awakened at the same time, an $O(n)$ upper bound can be shown, by using a technique similar to the one in [3]. The lower bound for the second class remain valid also without this assumption. A preliminary version of the results in this paper appeared in [8], together with an $O(n \log n)$ election algorithm for complete networks. Results similar to those in Section 3 were obtained independently in [2], by using different techniques.

The lower bounds above are best possible: An asynchronous algorithm of $O(n^2)$ messages can easily be designed for the second class. An optimal asynchronous algorithm for the problem of choosing a leader in a complete graph was first presented in [8], with message complexity $5n \log n + O(n)$. This algorithm is quite involved,
and was later extended in [7] to efficient algorithms for choosing a leader in general networks and in dynamic networks. Following [8], simpler algorithms for choosing a leader in complete networks appeared in [1, 5, 11]. These algorithms had a message complexity of $2n \log n + O(n)$. A more general approach in [6] resulted in yet another $2n \log n + O(n)$ simple algorithm for this network. These algorithms, together with the lower bound of $\Omega(n^2)$ for finding a minimum-weight spanning tree presented in [9], show that in complete networks it is easier to find a spanning tree than to find a minimum-weight spanning tree. Note that in general graphs the situation is different: when no processor knows the value of $n$, a minimum-weight spanning tree can be found in $O(n \log n + |E|)$ messages [4], and this bound is best possible for finding a (not necessarily minimum) spanning tree (see e.g. [12]). Moreover, this bound holds even if $n$ is known and, in fact, the graph is known up to isomorphism, as observed in the sequel.

2. Definitions and axioms

Let $A$ be a distributed algorithm acting on a graph $G = (V, E)$. An execution of $A$ consists of events, each being either sending a message, receiving a message or doing some local computation. With each execution we can associate a sequence $SEND = \langle send_1, send_2, \ldots, send_n \rangle$ that includes all the events of the first type in their order of occurrence (if there are no such events then $SEND$ is the empty sequence). In the case that two or more messages are sent at the same time, order them randomly (thus, in such case many sequences $SEND$ may correspond to the same execution). We identify each event $send_i$ with the triple $(v(send_i), e(send_i), m_i)$, where $v(send_i)$ is the node sending the message $m_i$ on the edge $e(send_i)$. We assume that $send_i$ occurred at time $0$, and $send_j$ occurred at time $\tau_j$ where $\tau_j \geq \tau_{j-1}$ for $i > 1$.

Let $SEND(t)$ be the prefix of length $t$ of the sequence $SEND$, namely $SEND(t) = \langle send_1, \ldots, send_t \rangle$ ($SEND(0)$ is the empty sequence). If $t < t'$ then we say that $SEND(t')$ is an extension of $SEND(t)$, and we denote $SEND(t) < SEND(t')$. $SEND$ is called a completion of $SEND(t)$. Note that a completion of a sequence is not necessarily unique.

Let $NEW = NEW(SEND)$ be the subsequence $\langle new_1, new_2, \ldots, new_r \rangle$ of the sequence $SEND$ that consists of all the events in $SEND$ that use previously unused edges. (An edge is used if a message has already been sent along it from either side.) This means that the message $send_i = (v(send_i), e(send_i), m_i)$ belongs to $NEW$ if and only if $e(send_i) \neq e(send_j)$ for all $j < i$. $NEW(t)$ denotes the prefix of size $t$ of the sequence $NEW$.

Define the graph $G(NEW(t)) = (V, E(NEW(t)))$, where $E(NEW(t))$ is the set of edges used in $NEW(t)$, and call it the graph induced by the sequence $NEW(t)$. If for every execution of the algorithm $A$ on a graph $G$ the corresponding graph $G(NEW)$ is connected then we say that algorithm $A$ is global for $G$. Note that if
A is global for \( G \) then \( G \) must be connected. Algorithm \( A \) is \textit{global for a family of graphs} \( \Gamma \) if it is global for every graph in \( \Gamma \). In this paper we shall call algorithm \textit{global} if it is global for the family of complete graphs.

Let \( \text{NEW}_i(t_i) \) be a prefix of a sequence \( \text{NEW}_i \) in which the last event occurred at time \( \tau_i \) for \( i = 1, 2 \), where \( \tau_1 \leq \tau_2 \). The \textit{synchronous merge} \( \text{NEW}_1(t_1) \circ \text{NEW}_2(t_2) \) of \( \text{NEW}_1(t_1) \) and \( \text{NEW}_2(t_2) \) is a sequence \( \langle s_1, \ldots, s_{n_1+n_2} \rangle \) obtained by associating with each event in \( \text{NEW}_1(t_1) \) that occurred in time \( \tau \) the time \( \tau + (\tau_2 - \tau_1) \), and then merging them in any order that is consistent with the new timing of the events. This merge modifies the timing of the messages so as to terminate their execution at the same time. Note that a synchronous merge of two sequences that correspond to legal executions of the algorithm does not necessarily correspond to a legal execution, as the two original executions may conflict with each other; however, in certain cases, as in Axiom 2 below, the synchronous merge yields a legal sequence.

For each algorithm \( A \) and graph \( G \) we define the \textit{exhaustive set of \( A \) with respect to \( G \)}, denoted by \( \text{EX}(A, G) \) (or \( \text{EX}(A) \) when \( G \) is clear from the context), as the set of all the sequences \( \sigma = \text{NEW}(t) \) corresponding to all possible executions of \( A \) and every possible \( t \geq 0 \).

From the model used in this paper the following facts—defined below as axioms—hold for every algorithm \( A \) and every graph \( G \).\textsuperscript{1}

**Axiom 1.** The empty sequence is in \( \text{EX}(A, G) \).

**Axiom 2.** If two sequences \( \sigma_1 \) and \( \sigma_2 \), which do not interfere with each other, are in \( \text{EX}(A, G) \), then so is also their synchronous merge \( \sigma_1 \circ \sigma_2 \). (\( \sigma_1 \) and \( \sigma_2 \) do not interfere with each other if no two edges \( e_1 \) and \( e_2 \) that occur in \( \sigma_1 \) and \( \sigma_2 \), respectively, have a common end point; this means that the corresponding partial executions of \( A \) do not affect each other and hence any of their synchronous merges corresponds to a legal execution of \( A \).)

**Axiom 3.** If \( \sigma \) is a sequence in \( \text{EX}(A, G) \) with a last element \( (v, e, m) \), and if \( e' \) is an unused edge adjacent to \( v \), then the sequence obtained from \( \sigma \) by replacing \( e \) by \( e' \) is also in \( \text{EX}(A, G) \). (This reflects the fact that a node cannot distinguish between its unused edges.)

Note that these three axioms do not imply that \( \text{EX}(A, G) \) contains any non-empty sequence. However, if the algorithm \( A \) is global then the following axiom holds as well:

**Axiom 4g.** If \( \sigma \) is in \( \text{EX}(A, G) \) and \( C \) is a proper subset of \( V \) containing the set of all the non-isolated nodes in \( G(\sigma) \), then there is an extension \( \sigma' \) of \( \sigma \) in which the first message \( (v, e, m) \) in \( \sigma' \) but not in \( \sigma \) satisfies \( v \in C \). (This reflects the facts that some unused edge will eventually carry a message and that isolated nodes in \( G(\sigma) \) may remain asleep until messages from already awakened nodes will reach them.)

\textsuperscript{1} These axioms reflect only some properties of distributed algorithms which are needed here.
The edge complexity $e(A)$ of an algorithm $A$ acting on a graph $G$ is the maximal length of a sequence $NEW$ over all executions of $A$.

The message complexity $m(A)$ of an algorithm $A$ acting on a graph $G$ is the maximal length of a sequence $SEND$ over all executions of $A$. Clearly $m(A) \geq e(A)$.

3. Lower bound for global algorithms in complete networks

The following lemma is needed in the sequel.

**Lemma 1.** Let $A$ be a global algorithm acting on a complete graph $G = (V, E)$, and let $U \subseteq V$. Then there exists a sequence of messages $\sigma$ in $EX(A, G)$ such that $G(\sigma)$ has one connected component whose set of vertices is $U$ and the vertices in $V - U$ are isolated.

**Proof.** If $|U| = 1$ then $\sigma = \emptyset$ (Axiom 1). Else, a desired sequence $\sigma$ can be constructed in the following way. Start with the empty sequence (using Axiom 1). Then add a message along a new edge that starts in a vertex in $U$ (Axiom 4) and that does not leave $U$ (Axiom 3 and the completeness of $G$). This is repeated until a graph having the desired properties is constructed. $\square$

**Theorem 1.** Let $A$ be a global algorithm acting on a complete graph $G$ with $n$ nodes. Then the edge complexity $e(A)$ of $A$ is $\Omega(n \log n)$.

**Proof.** For a subset $U$ of $V$ we define $e(U)$ to be the maximal length of a sequence $\sigma$ in $EX(A, G)$ which induces a graph that has a connected component whose set of vertices is $U$ and isolated vertices otherwise (such a sequence exists by Lemma 1). Define $e(k)$, $1 \leq k \leq n$, by

$$e(k) = \min \{ e(U) | U \subseteq V, |U| = k \}.$$  

Note that $e(n)$ is a lower bound on the edge complexity of the algorithm $A$.

The theorem will follow from the inequality

$$e(2k+1) \geq 2e(k) + k + 1 \quad (k < \frac{1}{2} n).$$

Let $U$ be a disjoint union of $U_1$, $U_2$ and $\{v\}$, such that $|U_1| - |U_i| = k$, and $e(U) = e(2k+1)$. We denote $C = U_1 \cup U_2$.

Let $\sigma_1$ and $\sigma_2$ be sequences in $EX(A, G)$ of lengths $e(U_1)$, $e(U_2)$ inducing subgraphs $G_1$, $G_2$ that have one connected component with vertex set $U_1$, $U_2$ (and all other vertices are isolated), respectively. These two sequences do not interfere with each other, and therefore—by Axiom 2—their synchronous merge $\sigma = \sigma_1 \cdot \sigma_2$.
is also in $EX(A, G)$. The proper subgraph $C$ of $G(\sigma)$ satisfies the assumptions of Axiom 4g. Note that each node in $C$ has at least $K$ adjacent unused edges within $C$. By Axiom 4g there is an extension of $\sigma$ by a message $(v, e, m)$, where $v \in C$. By Axiom 3 we may choose the edge $e$ to connect two vertices in $C$. This process can be repeated until at least one vertex in $C$ saturates all its edges to other vertices in $C$. This requires at least $k$ messages along previously unused edges. One more application of Axiom 4g and Axiom 3 results in a message from some node in $C$ to the vertex $v$. The resulting sequence $\sigma'$ induces a graph that contains one connected component on the set of vertices $U$ and isolated vertices otherwise. Thus we have

$$e(2k+1) = e(U) \geq e(U_1) + e(U_2) + k + 1 \geq 2e(k) + k + 1.$$  

The above inequality implies that for $n = 2^i - 1$ and the initial condition $e(1) = 0$ we have

$$e(n) \geq \frac{1}{2}(n+1) \log\left(\frac{1}{2}(n+1)\right).$$

It is obvious that $e(m) \geq e(n)$ for $m > n$, and the theorem is thus proved. \(\Box\)

From Theorem 1 we obtain the following theorem.

**Theorem 2.** Let $A$ be a global algorithm acting on a complete graph $G$ with $n$ nodes. Then the message complexity $m(A)$ of $A$ is $\Omega(n \log n)$. \(\Box\)

**Note 1.** The lower bounds in Theorems 1 and 2 hold even in the case where every node knows the identities of all other nodes (but cannot tell which edge leads to which node).

**Note 2.** In the execution constructed in the proof of Theorem 1 the number of processors which initialize the algorithm is $O(n)$ (it equals $\frac{1}{2}(n+1)$ for $n = 2^i - 1$). In fact, $\Omega(n)$ initiators are essential for any such example, since global algorithms of message complexity $O(n \log k)$, where $k$ is the number of the initiators of the algorithm, do exist (see [8]). Also, the timing of the initiations in this execution was not arbitrary. In fact, if all processors start within any known bound, then a synchronous algorithm that is using at most $O(n)$ messages can be constructed, using the ideas in [3].

**Note 3.** In [4] it was noted that global algorithms in general graphs require $|E|$ messages when the number of vertices is unknown. We note here that even when the numbers of nodes and edges are known—and, in fact, the graph is almost complete and is known up to isomorphism—then at least $|E| - 1$ messages may be required in the worst case. To see this, consider a complete graph of $n$ nodes to which a new vertex $v$ is added on some unknown edge (the resulting graph has $n + 1$ vertices and $(\frac{n}{2} + 1)$ edges). Apply the algorithm on such a graph with $v$ asleep, and as long as there are unused edges, assume that $v$ is on one of them. Thus $|E| - 1$ edges must be used in order to wake the vertex $v$. 

4. Lower bounds for matching-type algorithms in complete networks

The above theorems imply that algorithms for tasks like constructing a spanning tree, finding the maximum identity and finding a leader have a lower bound of $\Omega(n \log n)$ edges (and messages). These theorems are also applicable for tasks like constructing a Hamiltonian path or constructing a maximum matching. However, for these last two cases we show even a stronger result. Let a matching-type algorithm be an algorithm that is guaranteed to cover a maximum matching (that is, to induce a graph which contains a matching of size $\lfloor \frac{1}{2}n \rfloor$, where $\lfloor x \rfloor$ is the largest integer not larger than $x$). Clearly, a matching type algorithm must satisfy Axiom 4m below.

**Axiom 4m.** If $\sigma$ is in $EX(A, G)$ and $G(\sigma)$ does not contain a maximum matching of $G$, then there is a proper extension $\sigma'$ of $\sigma$.

**Theorem 3.** Let $A$ be a matching-type algorithm acting on a complete graph $G$ with $n$ nodes. Then the edge complexity $e(A)$ of $A$ is $\Omega(n^2)$.

**Proof.** Let $A$ be a matching-type algorithm. We construct a sequence in $EX(A, G)$ of length $\Omega(n^2)$. Arbitrarily number the vertices from 1 to $n$. We construct the sequence $\sigma$ in the following manner: Let $\sigma_0$ be the empty sequence (Axiom 1). For $i \geq 0$ if $G(\sigma_i)$ does not contain a maximum matching, let $\sigma_{i+1}$ be an extension of $\sigma_i$ by a message $(v, e, m)$ where $e = (v, i)$ is chosen with smallest possible $j$ (such an extension exists according to Axiom 3 and Axiom 4m).

By the assumption that $A$ is a matching-type algorithm, a sequence $\sigma$ in $EX(A, G)$ that does contain a maximum matching is eventually constructed. Let this matching be $\{(u_i, v_i)|1 \leq u_i < v_i \leq n \text{ and } u_i < u_{i+1}\}$.

Let $n_i$ be the number of messages in $\sigma$ which use an edge that connects $u_i$ or $v_i$ to some $j < u_i$. By the construction of $\sigma$ we have that $n_i \geq u_i - 1 \geq i - 1$. Thus the length of $\sigma$ is greater than

$$0 + 1 + \ldots + (\lfloor \frac{1}{2}n \rfloor - 1) = \frac{1}{2}n^2 + g(n),$$

where $|g(n)| = O(n)$. (Note that we did not count the edges $(u_i, v_i)$ of the matching.) This completes the proof of the theorem. Q.E.D.

From this theorem we obtain the following one.

**Theorem 4.** Let $A$ be a matching-type algorithm acting on a complete graph $G$ with $n$ nodes. Then the message complexity $m(A)$ of $A$ is $\Omega(n^2)$.

Note that Theorems 3 and 4 are independent of the number of initiators and of the timing of the initiations, which is not the case for Theorems 1 and 2. In other

---

3 It is not hard to see that an algorithm that is guaranteed to construct a maximum matching must be global for complete graphs of $n$ vertices for even $n$, and to induce connected graphs of at least $n - 1$ vertices for odd $n$. 
words, in complete networks, in which the number of initiators and their timing is not arbitrary, the lower bounds for global algorithms do not necessarily hold, while the lower bounds for matching-type algorithms still hold.

References


