Algorithmica (1997) 18: 405–416

Algorithmica

The Bit Complexity of Distributed Sorting

O. Gerstel[1] and S. Zaks[2]

Abstract. We study the bit complexity of the sorting problem for asynchronous distributed systems. We show that for every network with a tree topology T, every sorting algorithm must send at least $\Omega(\Delta_T \log(L/N))$ bits in the worst case, where $\{1, 2, \ldots, L\}$ is the set of possible initial values, and Δ_T is the sum of distances from all the vertices to a median of the tree. In addition, we present an algorithm that sends at most $O(\Delta_T \log((L \cdot N)/\Delta_T))$ bits for such trees. These bounds are tight if either $L = \Omega(N^{1+\varepsilon})$ or $\Delta_T = \Omega(N^2)$. We also present results regarding average distributions. These results suggest that sorting is an inherently nondistributive problem, since it requires an amount of information transfer that is equal to the concentration of all the data in a single processor, which then distributes the final results to the whole network. The importance of bit complexity—as opposed to message complexity—stems also from the fact that, in the lower bound discussion, no assumptions are made as to the nature of the algorithm.

Key Words. Distributed system, Distributed sorting, Bit complexity.

1. Introduction. In this paper we study the bit complexity of the sorting problem for asynchronous distributed systems. Such systems are composed of N processors, connected by communication lines, and communicate with one another by exchanging messages through these lines. The processors and the communication lines are modeled by an undirected graph. A protocol (or distributed algorithm) consists of operations of sending or receiving messages and doing local computations at each processor. The bit complexity of a given algorithm for a given network is the maximal number of bits sent during any possible execution of the algorithm on that network.

We study upper and lower bounds for the bit complexity of the sorting problem, in which each processor initially has an identifier and an initial value, and at the end the initial values have to be rearranged according to the initial identifiers. We show that, for every network with a tree topology T with N processors, there exists a distribution of initial values and identifiers for which every sorting algorithm must send at least $\Omega(\Delta_T \log(L/N))$ bits in the worst case, where $\{1, 2, \ldots, L\}$ is the set of possible initial values, and Δ_T is the sum of distances from a median of T to all the other processors in it. We then present an algorithm that sends at most $O(\Delta_T \log((L \cdot N)/\Delta_T))$ bits for such trees. These bounds are tight if either $L = \Omega(N^{1+\varepsilon})$ or $\Delta_T = \Omega(N^2)$, both of which are realistic assumptions (the first of which requires that the range of possible values is large enough: networks do not have 2^{32} processors, but integers do have this range; the second requires that the tree will not be too flat, e.g., trees with bounded degree). We also discuss the average distribution of initial values, and prove that, on the average, every sorting algorithm must send at least $\Omega(\Delta_T((L/N)^N/\binom{L}{N}) \log(L/N))$ bits.

[1] Optical Networking Group, Tellabs Operations, c/o IBM T.J. Watson Research Center, 30 Saw Mill River Road, Hawthorne, NY 10532, USA. ori@watson.ibm.com. Work done while at the Department of Computer Science, Technion, Haifa, Israel.

[2] Department of Computer Science, Technion, Haifa, Israel. zaks@cs.technion.ac.il.

Received May 2, 1994; revised December 22, 1995. Communicated by T. Lengauer.

These results imply that sorting is an inherently nondistributive problem, since the amount of information that has to be transferred is equal to a nondistributive version of the algorithm, in which one processor collects all the information from all other processors, calculates the final results, and redistributes the values appropriately.

In Section 2 we discuss related works, and in Section 3 we present a formal definition of the model and the problem. The worst-case and average-case results are discussed in Section 4. We mention a few open problems in Section 5.

2. Related Works. In [6] the message complexity of the sorting problem for ring networks was studied, where initial values have to be sorted clockwise, starting at any position; it was shown that every sorting algorithm requires $\Omega(N^2 \log(L/N))$ bits on a ring of size N, and an algorithm was presented that achieves this lower bound; similar results are shown for meshes. While [6] discusses only these two specific topologies, we study the class of all trees, and our results—especially the lower bound—apply for *every* tree in the class.

Our algorithm is a modification of the ones in [7], [3], and [6], where the *message complexity* of the sorting problem is studied for tree networks. However, the lower bounds in [7] and [3] assume a special class of algorithms, whereas we do not impose any constraints on the design of the algorithm. Moreover, in contrast with [7], in this work we deal with *bit complexity*, and our lower bound applies for *every* tree, and not just for the worst-case tree. A problem closely related to the sorting problem is ranking, and it is discussed in [5] and [7].

While all previous works discussed only the worst-case distribution of values, this work presents a lower bound on the average distribution as well. The work is an extension of the one presented in [4].

3. Preliminaries. A distributed system is composed of a set of N processors, connected by communication links. A network is viewed as an undirected graph $G = (V, E)$, whose set of vertices is the set of processors $V = \{v_1, \ldots, v_N\}$, and $(v_i, v_j) \in E$ iff there is a communication line connecting v_i and v_j (refer to [1] for graph-theoretic terms).

Every processor executes an *algorithm* that specifies the actions to be taken upon receipt of a message; these actions are sending messages and doing local computations. It is assumed that a message arrives at its destination after an unknown but finite delay. We do not assume that messages on a link arrive at the same order they were sent; nor do we assume that the algorithm in each processor is the same. This is a somewhat relaxed version of the conventional message-passing asynchronous model for networks (see [2]).

An *execution* is a sequence of *events* that are either sending or receiving a message at a processor according to its algorithm (a local computation can be regarded as part of either of these two types of actions). It is assumed that any nonempty subset of the processors may start the algorithm, and that no two events happen simultaneously. An execution has terminated if all the local algorithms have terminated. Initially each processor has some information concerning the problem it is about to solve, and upon termination of the algorithm each processor has a final value.

The *initial distribution* (or *distribution*) of identifiers and initial values in a distributed system is a function

$$\delta\colon V \to ID \times INIT,$$

where ID is the set of initial identities and $INIT$ is the set of initial values. Given a distribution δ, $ID_\delta(v)$ and $INIT_\delta(v)$ denote the initial identity and the initial value of processor v. Two distributions δ_1 and δ_2 are said to *agree* on a set of processors $\mathcal{Q}$ if $\delta_1(v) = \delta_2(v)$ for every $v \in \mathcal{Q}$. The set of executions of a given algorithm $\mathcal{A}$ for a given distribution δ is denoted by $EX(\mathcal{A}, \delta)$.

The *bit complexity* $b(\mathcal{A}, G, \delta)$ of an algorithm $\mathcal{A}$ on a graph G with initial distribution δ, is the maximal total number of bits (in all messages, over all the communication links) sent during any possible execution of $\mathcal{A}$ on G with initial distribution δ. We write $b(\mathcal{A})$ when G and δ are clear from the text.

We study the *sorting problem*. For this we assume that, for every distribution δ, all initial identifiers $ID_\delta(v_1), \ldots, ID_\delta(v_N)$ are distinct, while the initial values $INIT_\delta(v_1), \ldots, INIT_\delta(v_N)$ are not necessarily distinct. An algorithm is solving the sorting problem if it rearranges the initial values according to the initial identifiers; in other words, in every execution every processor v will have a final value $FINAL(v)$, such that the following two conditions are satisfied:

1. The multiset $\{INIT_\delta(v_1), \ldots, INIT_\delta(v_N)\}$ is equal to the multiset $\{FINAL(v_1), \ldots, FINAL(v_N)\}$, and
2. $ID_\delta(v) < ID_\delta(w) \Rightarrow FINAL(v) \leq FINAL(w), \forall v, w \in V$.

Let $T = (V, E)$ be a tree, and for $x, y \in V$ let $d(x, y)$ denote the distance between x and y in the tree. For $v \in V$, define

$$\Delta_T(v) = \sum_{u \in V} d(u, v),$$
$$\Delta_T = \min_{v \in V} \Delta_T(v).$$

Define the *median* of the tree to be a vertex m satisfying $\Delta_T(m) = \Delta_T$.

Let $\mathcal{F}(n) = \{0, 1\} \times \{1, 2, \ldots, n\}^*$, i.e., all finite sequences of pairs $\langle i, j \rangle$ such that $i \in \{0, 1\}$ and $j \in \{1, 2, \ldots, n\}$. Each such pair is called a *component*.

LEMMA 1 [6]. *Let S be a set of σ distinct sequences of $\mathcal{F}(n)$. The total number $\alpha(\sigma, n)$ of components in all the sequences of S satisfies*

$$\alpha(\sigma, n) \geq \frac{4\sigma \log(\sigma/10)}{5 \log(2n)}.$$

For a given network $G = (V, E)$ with distribution δ, let $\mathcal{Q}$ be a subset of V. $CUT(\mathcal{Q})$ contains all edges that have one end in $\mathcal{Q}$ and the other end not in $\mathcal{Q}$; namely,

$$CUT(\mathcal{Q}) = \{(v, v') \in E \mid v \in \mathcal{Q}, v' \notin \mathcal{Q}\}.$$

For an algorithm $\mathcal{A}$, the *signature* $SIG_\delta(e, c)$ of an execution e of $\mathcal{A}$ on the cut $c = CUT(\mathcal{Q})$ is a sequence in $\mathcal{F}(n)$, where $n = 2|c|$. Each component represents a

message arrival event, in the following way: assign the labels $1, 2, \ldots, n$ to the links of c (two labels per link), each label representing a direction on that link. During the execution e, messages are sent and received on links in c. The ith component of $SIG_\delta(e, c)$ is $\langle m, k\rangle$ if the ith message received is a message with contents m, and it was sent on a link and a direction implied by label k.

LEMMA 2 [6]. *Let $D = \{\delta_1, \ldots, \delta_t\}$ be a set of distributions that agree on a set of processors $\mathcal{Q}$, and let $\mathcal{A}$ be an algorithm. Let $E(D) = \{e_1, \ldots, e_t\}$ be a set of executions such that $e_i \in EX(\mathcal{A}, \delta_i)$ for every i. If the executions in $E(D)$ have less than $|D|$ different signatures on $CUT(\mathcal{Q})$, then there exist two executions $e_i, e_j \in E(D)$ for which the final values for processors in $\mathcal{Q}$ are the same.*

Define a *pairing* P to be a partition of the vertices of the tree into disjoint pairs. This partition leaves at most one vertex unmatched, in case $|V|$ is odd, and this vertex is marked by *free*(P).

LEMMA 3 [3]. *Let $T = (V, E)$ be a tree, and let m be a median of T. There exists a pairing P_T, such that all the paths connecting pairs of vertices in P_T pass through m. Furthermore, if $|V|$ is odd, then $m = free(P_T)$.*

4. Results

4.1. *Worst-Case Analysis.* Given a rooted tree T and a vertex v, define T_v to be the subtree rooted at v. Denote by $V(T)$ the vertex set of a tree T, and $V_w = V(T_w)$.

We first prove the following graph theoretic result:

LEMMA 4. *For every tree $T = (V, E)$ rooted at r:*

(1) $\sum_{w \in V\setminus\{r\}} |V_w| = \Delta_T(r)$.
(2) $\sum_{w \in V\setminus\{r\}} |V_w| \log |V_w| \geq \Delta_T(r) \log(\Delta_T(r)/|V|)$.

PROOF. The first part of the lemma is proven by induction on the structure of the tree T. The equation trivially holds for a tree with a single vertex.

In a tree rooted at r with children $u_1, u_2, \ldots, u_k$ we have $\Delta_T(r) = \sum_{i=1}^{k} (\Delta_{T_{u_i}}(u_i) + |V_{u_i}|)$, since every distance from a vertex w in T_{u_i} to u_i must be extended by one to reach r. By the induction hypothesis, $\Delta_{T_{u_i}}(u_i) = \sum_{w \in V_{u_i}\setminus\{u_i\}} |V_w|$. Combining these, we get

$$\begin{aligned}\Delta_T(r) &= \sum_{i=1}^{k} (\Delta_{T_{u_i}}(u_i) + |V_{u_i}|) = \sum_{i=1}^{k} \Delta_{T_{u_i}}(u_i) + \sum_{i=1}^{k} |V_{u_i}| \\ &= \sum_{i=1}^{k} \sum_{w \in V_{u_i}\setminus\{u_i\}} |V_w| + \sum_{i=1}^{k} |V_{u_i}| = \sum_{w \in V\setminus\{r\}} |V_w|.\end{aligned}$$

The proof of (2) is based on the fact that if the function $f(x)$ is concave, then, for

every x_is and k, we have

$$f\left(\frac{\sum_{i=1}^{k} x_i}{k}\right) \leq \frac{\sum_{i=1}^{k} f(x_i)}{k}.$$

Since $f(x) = x \log x$ is concave, we get

$$\begin{aligned}\frac{\sum_{w\in V\setminus\{r\}} |V_w| \log |V_w|}{|V|} &\geq \left(\frac{\sum_{w\in V\setminus\{r\}} |V_w|}{|V|}\right) \log \left(\frac{\sum_{w\in V\setminus\{r\}} |V_w|}{|V|}\right) \\ &= \left(\frac{\Delta_T(r)}{|V|}\right) \log \left(\frac{\Delta_T(r)}{|V|}\right),\end{aligned}$$

$$\sum_{w\in V\setminus\{r\}} |V_w| \log |V_w| \geq \Delta_T(r) \log \frac{\Delta_T(r)}{|V|}. \qquad \square$$

The next theorem deals with the lower bound.

THEOREM 5. *Let $\mathcal{A}$ be any sorting algorithm on a tree network T with N nodes, and let $INIT = \{1, 2, \ldots, L\}$. Then there exists an initial distribution δ, for which the bit complexity $b(\mathcal{A})$ of $\mathcal{A}$ satisfies*

$$b(\mathcal{A}) = \Omega\left(\Delta_T \log \frac{L}{N}\right).$$

PROOF. Let $T = (V, E)$, and let $P_T = \{(v_1, v_2), (v_3, v_4), \ldots, (v_{N-1}, v_N)\}$ be the pairing satisfying Lemma 3, where $V = \{v_1, \ldots, v_N\}$ or $V = \{v_1, \ldots, v_{N+1}\}$ (in which case $free(P_T) = v_{N+1}$). Let D be a set of initial distributions such that $\delta \in D$ if:

— $ID_\delta(v_i) = i$ for every $v_i \in V$,
— if i is even, then $INIT_\delta(v_i) \in \{(i-2)(L/N) + 1, \ldots, (i-1)(L/N)\}$,
— if i is odd: $INIT_\delta(v_{N+1}) = L$, and $INIT_\delta(v_i) \in \{i(L/N) + 1, \ldots, (i+1)(L/N)\}$, for $i < N$.

Note that $|D| = (L/N)^N$, since every processor in $\{v_1, \ldots, v_N\}$ has L/N possible values in D. It is easy to argue that, for every distribution in D, every pair of vertices $(u, v) \in P_T$ have to exchange values in order to perform sorting.

Direct the tree to be rooted at a median m. Let $a \in V$, $a \neq m$, and let b be its parent in T. Consider the following partition of V into disjoint sets:

(i) $V_1 = V(T_a)$,
(ii) $V_2 = \{v \mid \exists w \in V_1 : (v, w) \in P_T\}$,
(iii) $V_3 = V - V_1 - V_2$.

Consider a subset $D_a \subset D$ of executions such that processors in V_2 have all possible values as in D, while processors in $V_1 \cup V_3$ have some *fixed* value out of the values allowed in D.

From the definition of sorting it follows that, for every two distributions in D_a, if the initial values in V_2 are different, then the final values in V_1 will also be different, and that all processors in V_1 agree on all distributions of D_a.

Consider the cut $C = CUT(V_1) = \{(a, b)\}$ that separates V_1 from the rest of the network. If there are less than $|D|$ different signatures on C of executions in $E(D)$, then there exist two distributions $\delta_k, \delta_m \in D$ such that there exist executions $e_k \in E(\delta_k)$, $e_m \in E(\delta_m)$ for which the final values in V_1 are identical (by Lemma 2), a contradiction.

Therefore there exist at least $|D_a|$ different signatures of executions in $E(D_a)$ on C. Every such signature is in $\mathcal{F}(2)$, as C consists of one link, therefore, by Lemma 1, in all these signatures there are at least $\alpha(|D_a|, 2)$ components, each corresponding to sending a one-bit message.

Let $K(C, D)$ be the number of bits sent on C among all distributions in D. We have

$$K(C, D_a) \geq \alpha(|D_a|, 2).$$

Now all the above discussion is true for a specific $D_a \subset D$, and may be applied for every other D_a (there are $(L/N)^{N-|V_a|}$ such $D_a \subset D$). Therefore we have

$$K(C, D) = \sum_{D_a \subset D} K(C, D_a) \geq \left(\frac{L}{N}\right)^{N-|V_a|} \alpha(|D_a|, 2).$$

Note that all the above discussion may be applied for every $a \neq m$, and not just the fixed a we have chosen. Thus $K(D)$, the total number of bits sent on all the cuts in all distributions of D, satisfies

$$\begin{aligned}
K(D) &= \sum_C K(C, D) \geq \sum_{a \neq m} \left[\left(\frac{L}{N}\right)^{N-|V_a|} \alpha(|D_a|, 2) \right] \\
&\geq \sum_{a \neq m} \left[\left(\frac{L}{N}\right)^{N-|V_a|} \frac{4}{5 \log 4} \left(\frac{L}{N}\right)^{|V_a|} \log \frac{1}{10} \left(\frac{L}{N}\right)^{|V_a|} \right] \\
&= \sum_{a \neq m} \left[\left(\frac{L}{N}\right)^{N} \frac{4}{5 \log 4} \left(|V_a| \log \frac{L}{N} - \log 10 \right) \right] \\
&= \frac{4}{5 \log 4} \left(\frac{L}{N}\right)^{N} \left(\log \frac{L}{N} \sum_{a \neq m} |V_a| - (N-1) \log 10 \right) \\
&= \frac{4}{5 \log 4} \left(\frac{L}{N}\right)^{N} \left(\Delta_T \log \frac{L}{N} - (N-1) \log 10 \right),
\end{aligned}$$

where the last equality follows from Lemma 4.

By the pigeon-hole principle there exists a distribution $\delta_0 \in D$ for which at least $K(D)/|D|$ bits are sent on all links (cuts), and therefore the worst-case bit complexity satisfies

$$b(\mathcal{A}) \geq \frac{K(D)}{|D|} = \frac{4}{5 \log 4} \left(\Delta_T \log \frac{L}{N} - (N-1) \log 10 \right) = \Omega\left(\Delta_T \log \frac{L}{N} \right). \quad \square$$

THEOREM 6. *There exists a sorting algorithm $\mathcal{A}$, such that, for every tree network T with N nodes and every initial distribution taken from the set $INIT = \{1, 2, \ldots, L\}$, the bit complexity $b(\mathcal{A})$ satisfies*

$$b(\mathcal{A}) = O\left(\Delta_T \log \frac{L \cdot N}{\Delta_T}\right).$$

PROOF. We use a simple encoding of a set of numbers $\{n_1, n_2, \ldots, n_k\}$ in the range $\{1, 2, \ldots, L\}$. This encoding is done by sorting the numbers into ascending order $n_{i_1} \leq n_{i_2} \leq \cdots \leq n_{i_k}$, and sending the sequence $n_{i_1}, n_{i_2} - n_{i_1}, n_{i_3} - n_{i_2}, \ldots, n_{i_k} - n_{i_{k-1}}$ (it is easy to see how the original sequence can be derived from this sequence). This method requires $O(k \log(L/k))$ bits (rather than $O(k \log L)$ bits; see [6]).

Given the tree T, consider it as rooted at one of its median vertices m. The algorithm starts at the leaf processors, which send their identifiers and their initial values to their parents. Every internal processor waits to receive an encoding of the lists of identities and initial values accumulated at the children, decodes them, inserts its own ID and initial value to the corresponding lists and sends an encoding of these new lists to its parent. Eventually the root processor receives all the IDs and initial values of the processors in the tree, sorts them and sends to each child u the list of the final values that correspond to u and the processors in its subtree. Each internal processor can now determine its own final value and routes to each of its children the list with the corresponding final values. This process terminates at the leaf nodes. (This algorithm is a modification of the one in [7], to cope with bit complexity.)

Despite the inherently large amount of information transfer, this algorithm does have good local properties:

LOW LOCAL PROCESSING TIME. Although the algorithm is centralized, each processor does not actually need to sort the values it received from its children, but only to merge them, adding its own value.

LOW LOCAL MEMORY REQUIREMENTS. The amount of memory in each processor can be proportional to its degree in the tree, since it can take the first (and smallest) value from each child, and decide which value is sent upward to the parent, and only then take the next value from that child. This modification requires a careful design of the algorithm (in particular, special steps must be taken so that the root of the tree does not use too much memory, and so that the queues between the processors remain bounded).

During the execution of the algorithm two messages are sent on each edge from child to parent (containing identities and initial values), and one message is sent from parent to child (containing final values). Each message from/to a subtree T_v rooted at v contains at most $3|V_v| \log(L/|V_v|)$ bits, so the total number of bits sent during an execution is

$$b(\mathcal{A}) \leq 3\left(\sum_{w \in V \setminus \{m\}} |V_w| \log \frac{L}{|V_w|}\right) = 3\left(\sum_{w \in V \setminus \{m\}} |V_w| \log L - \sum_{w \in V \setminus \{m\}} |V_w| \log |V_w|\right)$$

and this implies, by Lemma 4(2),

$$b(\mathcal{A}) \leq 3\Delta_T \left(\log L - \log \frac{\Delta_T}{N} \right) = 3\Delta_T \log \frac{L \cdot N}{\Delta_T}.$$

This completes the proof of the theorem. □

The above two theorems give upper and lower bounds on the bit complexity of sorting on a tree network. These bounds are tight if either the range of initial values in L is significantly larger than the size N of the network (which is usually the case in communication networks), or if the tree is deep enough (e.g., a tree with bounded degrees—paths, binary trees, etc.; this assumption of bounded degree is common in practice). Formally, we get:

COROLLARY 7. *For every tree network T and set of initial values $INIT = \{1, 2, \ldots, L\}$, if either of the following conditions holds, then the bounds of Theorem* 5 *and Theorem* 6 *are tight*:

1. $L = \Omega(N^{1+\varepsilon})$ *for some fixed* $\varepsilon > 0$.
2. $\Delta_T = \Omega(N^2)$.

These results can be easily extended to the case where each processor holds several values (and not just one), namely, each processor p has a capacity $C(p)$ of values, which is the number of initial and final values it holds. This extension implies that each vertex of the underlying graph has a weight. To this end, the graph-theoretic terms must be redefined, namely, $\Delta_T(v)$ is modified to $\Delta_{T,W}(v)$, the median as the vertex achieving minimum $\Delta_{T,W}$, etc. In this case, we achieve similar bounds to the ones mentioned above:

$$b(\mathcal{A}) = \Omega\left(\Delta_{T,W} \log \frac{L}{N} \right),$$

$$b(\mathcal{A}) = O\left(\Delta_{T,W} \log \frac{LN}{\Delta_{T,W}} \right).$$

4.2. *Average-Case Analysis*. In this subsection we deal with the expected bit complexity on an average distribution. The proof technique is similar to the one of Theorem 5, but more definitions and lemmata are needed.

We define a *permutation* π as $\pi = \{\langle u_1, v_1\rangle, \ldots, \langle u_N, v_N\rangle\}$ where $\{u_i\}_{i=1}^N = \{v_i\}_{i=1}^N = V$. The permutation defines the source and destination (u and v, respectively) of the values in the given distribution.

The *distance* $\Gamma_T(\pi)$ of a given permutation is defined by

$$\Gamma_T(\pi) = \sum_{\langle u,v\rangle \in \pi} d(u, v).$$

Every edge e splits the tree into two subtrees T_0^e and T_1^e. Define the *split set*, $\Upsilon_\pi(e)$, as the set of vertices whose source in π is in T_1^e and its destination in T_0^e (see Figure 1); namely,

$$\Upsilon_\pi(e) = \{u \mid \langle u, v\rangle \in \pi, u \in T_1^e, v \in T_0^e\}.$$

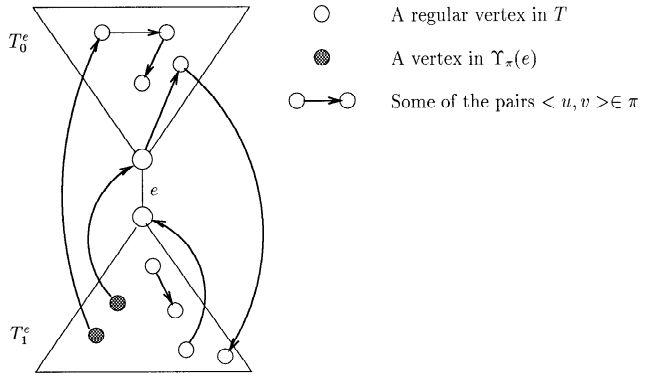

Fig. 1. The split set of edge $e - \Upsilon_\pi(e)$.

Finally, we define $\bar{b}(\mathcal{A})$ to be the sum of $b(\mathcal{A})$ for all distributions, divided by the total number of distributions.

We need the following two lemmata on the functions we have just defined.

LEMMA 8. $\sum_{e \in E} |\Upsilon_\pi(e)| = \frac{1}{2}\Gamma_T(\pi)$ *for every permutation* π *and tree* T.

PROOF. Given a pair $\langle u, v\rangle \in \pi$, denote by $\mathcal{R}_{u,v}$ the (single shortest) route between u and v in T. It is easy to see that, on every edge e in T, there is an equal number of routes $\mathcal{R}_{u,v}$ that traverse e in each direction, and that these routes are of pairs $\langle u, v\rangle$ for which $e \in \mathcal{R}_{u,v}$. Therefore, each edge e is counted in $\Upsilon_\pi(e)$ for exactly half of the pairs $\langle u, v\rangle$ for which $e \in \mathcal{R}_{u,v}$ (i.e., for those whose source is in T_1^e). The proof is completed by the observation that $\Gamma_T(\pi) = \sum_{\langle u,v\rangle \in \pi} |\mathcal{R}_{u,v}|$. □

LEMMA 9. $\sum_\pi \Gamma_T(\pi) \geq \Delta_T N!$ *for every tree* T *with* N *vertices.*

PROOF. By a symmetry argument, each pair $\langle u, v\rangle \in \pi$ is included in an equal number of permutations. There are $N!$ permutations, each including N pairs, hence, each pair is included $(N-1)!$ times in $\sum_\pi \Gamma_T(\pi)$. On the other hand,

$$\sum_{\langle u,v\rangle \in \pi} d(u, v) = \sum_{u \in V} \Delta_T(u) \geq N\Delta_T.$$

Combining these, we get

$$\frac{\sum_\pi \Gamma_T(\pi)}{(N-1)!} \geq N\Delta_T$$

or

$$\sum_\pi \Gamma_T(\pi) \geq N!\,\Delta_T.$$ □

THEOREM 10. *Let $\mathcal{A}$ be any sorting algorithm on a tree network T, with a set of initial values $INIT = \{1, 2, \ldots, L\}$. The expected bit complexity $\bar{b}(\mathcal{A})$ of $\mathcal{A}$ satisfies*

$$\bar{b}(\mathcal{A}) = \Omega\left(\Delta_T \frac{(L/N)^N}{\binom{L}{N}} \log \frac{L}{N}\right).$$

PROOF. We prove the bound for a fixed distribution of IDs δ_{ID}, and since it does not depend on that distribution, it will also hold for all ID distributions.

Let $\mathcal{D}$ be the set of all distributions of distinct initial values. Define a *simple subset of distributions* $SD \subset \mathcal{D}$ by assigning each processor with ID $i \in \{1, 2, \ldots, L\}$, the values $\{(i-1)(L/N)+1, \ldots, i(L/N)\}$. For a given permutation $\pi = \{\langle u_1, v_1\rangle, \ldots, \langle u_N, v_N\rangle\}$, and a distribution δ, define δ_π by

(i) $ID_{\delta_\pi}(u_i) = ID_\delta(u_i)$ for every i, and
(ii) $INIT_{\delta_\pi}(u_i) = INIT_\delta(v_i)$ for every i,

and let $SD_\pi = \{\delta_\pi | \delta \in SD\}$.

We note the following:

1. Each of the distributions in SD is sorted,
2. $|D| = L!/(L-N)!$,
3. $|SD| = (L/N)^N$, and
4. for every two permutations $\pi \neq \pi'$, $SD_\pi \cap SD_{\pi'} = \emptyset$.

Define $D(\delta, S)$ as the set of distributions in SD_π that have fixed values for the processors outside a given set S, and varying values in S, for a given $\delta \in SD_\pi$; namely,

$$D(\delta, S) = \{\delta' \in SD_\pi | \forall v \in V : v \notin S \Rightarrow \delta'(v) = \delta(v)\}.$$

Given an edge e, Lemma 1 may be applied for the distributions in $D(\delta, \Upsilon_\pi(e))$, since they all agree on the values in T_0^e, so we have (for some constants C_1, C_2)

$$K(\{e\}, D(\delta, \Upsilon_\pi(e))) \geq \alpha(|D(\delta, \Upsilon_\pi(e))|, 2).$$

It can be shown that, for any given distribution $\delta \in SD_\pi$,

$$K(\{e\}, SD_\pi) = \sum_{\delta' \in D(\delta_\pi, V \setminus \Upsilon_\pi(e))} K(\{e\}, D(\delta', \Upsilon_\pi(e))),$$

and using Lemma 1 we get

$$K(\{e\}, SD_\pi) \geq C_1 \left(\frac{L}{N}\right)^{N-|\Upsilon_\pi(e)|} \left(\frac{L}{N}\right)^{|\Upsilon_\pi(e)|} \left(|\Upsilon_\pi(e)| \log \frac{L}{N} - \log 10\right).$$

By

$$K(SD_\pi) = \sum_{e \in E} K(\{e\}, SD_\pi)$$

and using Lemma 8, it follows that

$$\begin{aligned} K(SD_\pi) &\geq C_1 \left(\frac{L}{N}\right)^N \left(\sum_{e\in E} \Upsilon_\pi(e) \log \frac{L}{N} - (N-1)\log 10\right) \\ &= C_2 \left(\frac{L}{N}\right)^N \left(\Gamma_T(\pi) \log \frac{L}{N} - o(N)\right). \end{aligned}$$

Following our discussion, we now have

$$\begin{aligned} \bar{b}(\mathcal{A}) &= \frac{\sum_{d\in\mathcal{D}} K(d)}{|\mathcal{D}|} \\ &\geq \frac{\sum_\pi K(SD_\pi)}{L!/(L-N)!} \\ &\geq C_2 \frac{(L/N)^N}{L!/(L-N)!} \left(\sum_\pi \Gamma_T(\pi) \log \frac{L}{N} - o(N)\cdot N!\right). \end{aligned}$$

Finally, using Lemma 9, we get

$$\bar{b}(\mathcal{A}) \geq C_2 N! \frac{(L/N)^N}{L!/(L-N)!} \left(\Delta_T \log \frac{L}{N} - o(N)\right) = \Omega\left(\Delta_T \frac{(L/N)^N}{\binom{L}{N}} \log \frac{L}{N}\right). \quad \square$$

Note that if we restrict our discussion to the distributions in $\bigcup_\pi SD_\pi$, then we get a tight bound of $\bar{b}(\mathcal{A}) = \Theta(\Delta_T \log(L/N))$.

5. Open Problems. We mention the following open problems:

— Our bounds—for the worst case—are tight under the conditions of Corollary 1. Determine all cases where these bounds are tight. For example, the upper bound of Theorem 6 seems to be tight (it is tight for star trees and for simple paths).
— Tighten the gap between the lower and upper bound in the case of average complexity.
— Extend the results to other classes of graphs.

References

[1] J. A. Bondy and U. S. R. Murty, *Graph Theory with Applications*, North-Holland, Amsterdam, 1976.
[2] R. G. Gallager, P. A. Humblet, and P. M. Spira, A distributed algorithm for minimum spanning tree, *ACM Transactions on Programming Languages and Systems*, 5(1) (1983), 66–77.
[3] O. Gerstel and S. Zaks, A new characterization of tree medians with applications to distributed algorithms, *Networks*, 24(1) (1994), 23–29; also: *Proceedings of the* 18*th International Workshop on Graph-Theoretic Concepts in Computer Science* (*WG* '92), Frankfurt, June 1992, LNCS 657, Springer-Verlag, Berlin, pp. 135–144.
[4] O. Gerstel and S. Zaks, The bit complexity of distributed sorting, *Proceedings of the* 1*st Annual European Symposium on Algorithms* (*ESA* '93), Bad Honnef, September 1993, LNCS 726, Springer-Verlag, Berlin, pp. 181–191.

[5] E. Korach, D. Rotem, and N. Santoro, Distributed algorithms for ranking the vertices of a network, *Proceedings of the* 13*th Southeastern Conference on Combinatorics, Graph Theory, and Computing*, 1982, pp. 235–246.

[6] M. C. Loui, The complexity of sorting in distributed systems, *Information and Control*, 60(1-3) (1984), 70–85.

[7] S. Zaks, Optimal distributed algorithms for sorting and ranking, *IEEE Transactions on Computers*, 34(4) (1985), 376–379.