Network Partition and Its Application to Distributed Selection Problem

Jyun-Jy Hu†, Shi-Nine Yang† and Maw-Sheng Chern‡

† Institute of Computer Science
National Tsing Hua University
Hsinchu, Taiwan 30043
Republic of China

‡ Institute of Industrial Engineering
National Tsing Hua University
Hsinchu, Taiwan 30043
Republic of China

Abstract: The distributed selection problem is to select the k-th smallest element of a set N of elements distributed among nodes of a point-to-point asynchronous communication network. Efficient algorithms have been found for networks with simple topologies such as stars, meshes and complete binary trees but not for general trees. In this paper, we present an improved selection algorithm for general tree networks. Base on the conventional reduction strategy, we introduce a tree partition technique to localize the message passing and therefore to reduce the total message complexity.

Key words: Distributed algorithms, selections, medians, tree partitions, message complexity.

1. Introduction

The classical problem of selecting the k-th smallest element of a set drawn from a totally ordered set has been studied by many researchers in serial and parallel models. But in a distributed model, it has different formulations and complexity measures and in this paper we will confine ourself to a point-to-point asynchronous communication network.

A point-to-point asynchronous communication network can be modeled by an undirected graph $G = (V, E)$, where the nodes of V represent processors of the network and the edges of E represent bidirectional non-interfering communication links. There is no central controller, neither common clock for network synchronization and there is no global memory for interprocessor communications. Instead, they are done through various messages. Therefore the distributed algorithms considered in this paper are primarily message driven. All the messages have a fixed length, and may carry only a limited amount of information. Furthermore, we assume the network is sufficiently reliable that the messages sent in a link is received error free by the receiver node in First-In-First-Out (FIFO) order with finite but totally unpredictable delays. It is usually assumed that the local processing time is negligible when compared to the communication delay, therefore the performance of the algorithm is customarily measured by the amount of communication activity, that is, the message complexity, rather than the amount of processing activity. This type of computation model has been used by several authors [1–3, 5, 7, 8].

The distributed selection problem is to find the k-th smallest element of a set N of elements which are stored distributively among nodes of a communication network. For a general point-to-point network, Shrira et al [8] assumed there is a rooted spanning tree on the network and proposed three algorithms to solve this problem. The first one is a probabilistic algorithm with expected $O(|V| \log |N|)$ messages while in the worst case it needs $O(|V| |N|)$ messages. The other two algorithms are deterministic algorithms with the same message complexity $O(|V| |N|^{0.9114})$ and different storage requirements. In this paper, we present an improved algorithm with message complexity $O(\sqrt{\log h}\, 4^{\sqrt{\log h}} |V| \log |N|)$ where h is the height of the tree. This result significantly improves the existing solution proposed by Shrira et al [8]. In [7], Santoro and Suen proposed an open problem to find an efficient selection algorithm for arbitrary tree networks. Since the message complexity of our algorithm is bounded by $O(h^{\epsilon} |V| \log |N|)$ for any small $\epsilon > 0$ and Frederickson [5] has shown $\Omega(|V| \log(2|N|/|V|))$ is the message lower bound for the selection problem of complete binary trees, we think our algorithm gives an asymptotically efficient solution to that open problem. Finally we note that if the topology of a network is not a tree, a spanning tree of the network can be constructed first [1–3] and then the k-th smallest element can be found by using the proposed algorithm.

There are other algorithms which solve the k-th smallest element problem but all are for networks with different models or special topologies. Santoro and Suen [7] proposed a reduction technique for shout-echo networks and point-to-point networks whose topology have a spanning star graph. Frederickson [5] presented algorithms for rings, meshes and complete binary trees.

The rest of the paper is arranged in the following manner. A basic distributed selection algorithm is given in Section 2. Section 3 presents a tree partition. The reduction technique based on tree partitions is discussed in Section 4. Section 5 gives an application of selection algorithm — distributed convex hull algorithm. Section 6 contains our conclusions.

2. Basic distributed selection algorithm

For sequential selection problem, Blum et al [4] introduced the idea of median of medians and obtained an optimal sequential selection algorithm with $O(n)$ time complexity. For a set of elements, a median of medians drawn from the set has the property that there are at least 1/4 of elements which are greater than and at least 1/4 elements which are smaller than the median of medians. Therefore Blum et al pick a median of medians as the pivoting element, recursively prune at least 1/4 elements which can not be the k-th smallest element each time. Shrira et al [8] transferred the idea into distributed environment and derived a distributed selection algorithm with message complexity $O(|V|\,|N|^{0.9114})$. In Shrira et al's algorithm, major amount of messages are due to finding a median of medians. On the other hand, a similar idea of median of medians, the weighted median, is introduced by Johnson and Mizoguchi [6]:

Definition 1 [6]. Let $w: A \rightarrow \mathbf{R}^+$ be a weight function defined on A. An element a_m satisfying $\sum_{i=1}^{m} w(a_i) \geq \sum_{i=m+1}^{n} w(a_i)$ and $\sum_{i=1}^{m-1} w(a_i) \leq \sum_{i=m}^{n} w(a_i)$ is called a *weighted median of A with respect to w.*

If the element a_i is a median of a set of elements A_i and the weight function $w(a_i) = |A|$, then it can be shown that for the set of elements $A = \bigcup_{i=1} A_i$, there are at least 1/4 of elements in A which are greater than and at least 1/4 elements in A which are smaller than the weighted median a_m. By using weighted median instead of median of medians as pivoting element, Marberg and Gafni [9], Santoro and Suen [7] proposed several optimal distributed selection algorithms in shout-echo networks.

It is not necessary to find a pivoting element which can prune at least 1/4 elements still under consideration. In fact, we may find a pivoting element which just can prune at least 1/8, 1/16, or fewer elements. It also costs fewer messages to find such a pivoting element and therefore reduces the total message complexity of the distributed selection algorithm. In the following, we will formalize this idea.

First we generalize the notion of median of medians and weighted median to the p-th order median for $p \geq 1$. A constructive method for finding a p-th order median distributively will be given in Theorem 1.

Let $A = \{a_i \mid 1 \leq i \leq n\}$ be a set of elements with the total order "$\leq$". Without loss of generality we may assume $a_i \leq a_{i+1}$ for $1 \leq i < n$.

Definition 2. A subset of A is said to be the *p-th order median set* for $p \geq 1$ if the subset contains all elements which are greater than or equal to at least $\lceil |A|/2^p \rceil$ elements in A and are less than or equal to at least $\lceil |A|/2^p \rceil$ elements in A. Any element in a p-th order median set is called a *p-th order median.*

Let A_i, $1 \leq i \leq n$, be sets of elements and let $A = \bigcup_{i=1}^{n} A_i$. The following theorem provides a method for finding a $(p+1)$-th order median of A if a p-th order median of each A_i is available.

Theorem 1. Let A_i, $1 \leq i \leq n$, be sets of elements and m_i be a p-th order median of A_i. If m is a weighted median of the set $\{m_i \mid 1 \leq i \leq n\}$ with respect to the weight function $w(m_i) = |A_i|$, then m is a $(p+1)$-th order median of $A = \bigcup_{i=1}^{n} A_i$.

Proof. Without loss of generality, we may assume $m_i \leq m_{i+1}$ for $1 \leq i < n$ and $m = m_j$ for some j, $1 \leq j \leq n$. We can prove m_j is greater than or equal to at least

$\lceil |A|/2^{p+1} \rceil$ elements in A. Similarly, we can prove that m is less than or equal to at least $\lceil |A|/2^{p+1} \rceil$ elements in A. Therefore, m is a $(p+1)$-th order median of A. □

Now we will introduce a basic algorithm to solve the distributed selection problem based on the conventional reduction strategy, which iteratively selects a pivoting element to reduce the problem size until it reaches a comfortable size or the k-th smallest element is found.

Intuitively, this algorithm starts from the root to ask for a first order median and the number of elements of each node. After collecting all these information, the root finds a second order median m according to Theorem 1 and sends m back to each node as the pivoting element for the first reduction. By comparing the rank of m in N with k, either we find m is the k-th smallest element or we can remove at least 1/4 of the elements. For the latter case, since some elements have been removed from consideration, we may adjust k and N and call the same procedure recursively until the k-th smallest element is found. We will discuss the details in the following.

Let the given communication tree network be $T = (V, E)$ with height h. We denote the set of elements stored on the node v as $N(v)$ and $L(v; m) = \{a \mid a \in N(v) \text{ and } a < m\}$. Furthermore, we assume all elements in N are distinct for simplicity.

The following two procedures are called by the selection algorithms. The first procedure is the Procedure Median which is executed by the root node to find a second order median of N. The second procedure is the Procedure Reduce-and-Find which is also executed by the root node to check whether an element m is the k-th smallest element. If it is, then the root node will terminate the selection algorithm by broadcasting a stopping message to each node. If it is not the case, then the root node will use the element m as a pivoting element to reduce the problem size.

Procedure Median.

Step 1. The root node broadcasts an initial message to each node.

Step 2. After receiving the initial message sent from the root node, every node $v_i \in V$ sends the median m_i of $N(v_i)$ and $|N(v_i)|$ to the root node.

Step 3. The root node finds a weighted median m of $\{m_i \mid 1 \le i \le |V|\}$ with respect to the weight function $w(m_i) = |N(v_i)|$. Now the weighted median m is a second order median of N.

Procedure Reduce-and-Find.

Input: An element m.

Step 1. The root node finds the overall rank r of m in N. This can be done by broadcasting m to each node and then accumulate $|L(v; m)|$ from all leaf nodes up to the root node for all $v \in V$.

Step 2. The root node determines whether or not m is the k-th smallest element of N. If $r = k$ then m is the k-th smallest element of N and the root node terminates the algorithm by broadcasting a stopping message to each node. If $r > k$ then the root node broadcasts a message *Too-Large* to each node. If $r < k$ then the root node sets $k = k - r$ and broadcasts a message *Too-Small* to each node.

Step 3. If node v receives the message *Too-Large*, v sets $N(v) = L(v; m)$. If v receives the message *Too-Small*, v sets $N(v) = N(v) - L(v; m) - \{m\}$.

Now we give the basic distributed selection algorithm as follows.

Algorithm Selection-I.

Step 1. The root node finds a second order median m of N by executing the Procedure Median.

Step 2. Execute the Procedure Reduce-and-Find with input m. If m is the k-th smallest element of N then every node will receive the stopping message. If not, the problem size is reduced.

Step 3. If a node has received the stopping message, then it stops. When the root node has stopped, the algorithm is terminated and the k-th smallest element of N is m. If not, go to Step 1.

It can be easily seen that the message complexity of the Procedure Median and the Procedure Reduction is $O(h|V|)$ and $O(|V|)$ respectively. Since in each iteration of the Algorithm Selection-I we use a second order median of N to reduce at least 1/4 of the elements in N and there are at most $O(\log |N|)$ iterations, the message complexity of Algorithm Selection-I is therefore $O(h|V| \log |N|)$.

3. Tree Partitions

In the Algorithm Selection-I, we notice that to send every median all the way up to the root is the major cost. In order to avoid this process, we can first partition the tree properly into subtrees within a given height. Then each node only has to send its median to the root node of the subtree it belongs to. Now each

root of the subtrees can compute a second order median of elements in its own subtree. Then the root of the tree collects these second order medians to find a third order median as the pivoting element. We can show that with the help of tree partitions, the improved algorithm runs with message complexity $O(h^{1/2} \mid V \mid \log \mid N \mid)$. In fact we may partition the tree recursively several times, and obtain a selection algorithm with more better message complexity. In the following, we give some notations of tree partitions and a distributed tree partition algorithm.

Definition 3. Let $s \geq 1$ be an integer. A tree is said to be a *height and weight constrained tree* (*HW tree* for short) *with respect to* s if the number of nodes of the tree is at least s and the height of the tree is at most s. A tree is said to be an *underweight tree with respect to* s if the number of nodes of the tree is less than s.

Definition 4. Let T be a tree and $s \geq 1$ be an integer. A *height and weight constrained partition* (*HW partition* for short) *of* T *respect to* s, denoted by $P(s)$, is a partition of T such that T is partitioned into *HW* subtrees except at most one underweight subtree with respect to s. Therefore an *HW* partition is an almost uniform partition as far as the tree height is concerned.

Let us consider $\{T_i = (V_i, E_i) \mid 1 \leq i \leq n\}$, the collection of all second level subtrees obtained by an *HW* partition $P(s_1)$ on a tree T. If we identify each T_i as a node, then the original tree can be regarded as a tree of subtrees, denoted by $T[s_1]$. If we construct an *HW* partition $P(s_2)$ on the tree $T[s_1]$, then we obtain a set of *HW* subtrees with respect to s_2 and each of them contains at least s_2 subtrees, and therefore $s_1 \times s_2$ nodes of T. This is a multi-level partition of T and a recursive definition based on top-down principle is given in the following.

Definition 5. Let T be a tree and $s = (s_1, s_2, \ldots, s_p)$ be a p-tuple of integers where $s_i \geq 1$ for $1 \leq i \leq p$. A *p-th level height and weight constrained partition* (*p-th level HW partition* for short) of T with respect to s, denoted by $P(s_1, s_2, \ldots, s_p)$, is a partition of T such that if $p = 1$ then $P(s_1)$ is a *HW* partition with respect to s_1, otherwise T is partitioned into *HW* subtrees except at most one underweight subtree with respect to $\prod_{i=1}^{p} s_i$, and each subtree has a $(p-1)$-th level *HW* partition with respect to the $(p-1)$-tuple $(s_1, s_2, \ldots, s_{p-1})$.

In the following, we introduce an integer representation based on given $s = (s_1, s_2, \ldots, s_p)$ to represent the height of multi-level subtrees.

Definition 6. Let n be a positive integer and $s = (s_1, s_2, \ldots, s_p)$ be a p-tuple of integers where $s_i \geq 1$ for $1 \leq i \leq p$. The *partition representation of n with respect to s* is $(n_p n_{p-1} \cdots n_2 n_1 n_0)$ if $n = n_p \prod_{i=1}^{p} s_i + n_{p-1} \prod_{i=1}^{p-1} s_i + \cdots + n_2 \prod_{i=1}^{2} s_i + n_1 \prod_{i=1}^{1} s_i + n_0$ where $n_p \geq 0$ and $0 \leq n_i < s_{i+1}$ for $0 \leq i < p$.

For example, let $s = (3, 2)$ and $n = 5$. The partition height of n with respect to s is (012) since $5 = 0 \times 6 + 1 \times 3 + 2$.

Definition 7. Let T be a tree with a p-th level *HW* partition $P(s_1, s_2, \ldots, s_p)$. The *partition height* of a node v is 1 if the node is a leaf node. Otherwise the partition height of v is $(n_p n_{p-1} \cdots n_2 n_1 n_0) + 1$ where $n_j = \max_{1 \leq i \leq m} \{n_{i,j}\}$ and $(n_{i,p} n_{i,p-1} \cdots n_{i,2} n_{i,1} n_{i,0})$ is the partition height of v_i with respect to $s_1, s_2, \ldots, s_p$ and nodes $v_1, v_2, \ldots, v_m$ are sons of v.

The following algorithm describes a labeling method which will correspond to a p-th level *HW* partition discussed above.

Algorithm Label.

Input: p integers $s_1, s_2, \ldots, s_p$.

Step 1. The root node broadcasts a message P and p integers $s_1, s_2, \ldots, s_p$ to all nodes.

Step 2. After receiving the message P and integers $s_1, s_2, \ldots, s_p$, if node v is a leaf node, then v sets its

partition height θ to be 1, otherwise it sets $\theta = (n_p n_{p-1} \cdots n_2 n_1 n_0)$ with respect to $s_1, s_2, \ldots, s_p$.

Step 3. For each node $v \in V$, v is labeled $\lambda + 1$ if the n_λ is the first digit not equal to zero, that is, $n_\lambda > 0$ and $n_i = 0$ for $0 \leq i < \lambda$.

Step 4. The root node is labeled $p + 2$.

Since Step 1 broadcasts p integers to all nodes, the message complexity of Algorithm Label is $O(p \mid V \mid)$. If all inputs s_i are equal to the same integer s for $1 \leq i \leq p$, we can just send two integers: p and s. So the message sent in the first step is reduced to $O(\mid V \mid)$ and the message complexity of the algorithm is also reduced to $O(\mid V \mid)$.

4. Selection Algorithm

Now we introduce the main idea of our improved selection algorithm. Let U be a subset of V and $N(U) = \bigcup_{v \in U} N(v)$ be the set of elements in U. Let $\{T_i = (V_i, E_i) \mid 1 \leq i \leq n\}$ be the collection of all subtrees after an HW partition. Suppose the root node of subtree T_i finds a second order median m_i of $N(V_i)$ first, and then sends it to the root node of the tree T to compete for the pivoting element — a third order median of N. Then we can avoid every node sending its candidate to the root. We note that our labeling method implicitly implies this idea. For example, a node labeled 2 can be regard as the root of two subtrees, namely, the subtree containing itself and a subtree T_i, therefore it has to compute its own median first and then a second order median of $N(V_i)$. In general, the node with label λ will compute a p-th order median for every $1 \leq p \leq \lambda$.

The notion of multi-level subtrees induced by our labeling method is formally described in the following definition.

Definition 8. Let $T = (V, E)$ be a tree and $\lambda(v)$ be the label of v according to the Algorithm Label. Then for each $1 \leq p \leq \lambda(v)$, the *p-th level subtree rooted at node v* is a subtree T' of T such that a node $u \in T'$ if and only if $u = v$ or $\lambda(u) < p$ and the father of u is in T'.

In the following, we will partition the tree and use a p-th order median instead of a second order median to remove a certain amount of elements. We will investigate how much further we can keep on partitioning as far as the communication efficiency is concerned. First we will analyze the message complexity for the selection algorithm based on a reduction strategy by partitioning the tree $p-1$ times recursively, and then derive an optimal value p such that the message complexity is minimized.

A $(p+2)$-th order median of a tree can be found when a p-th level HW partition has been constructed. Let v be the root of a λ-th level subtree $T_v = (V_v, E_v)$. Then v finds a λ-th order median m and sends m and $\mid N(V_v) \mid$ to the nearest ancestor node with label $\lambda + 1$ for $1 \leq \lambda < p + 2$. Then the root node find a $(p+2)$-th order median.

Let $M(s_1, s_2, \ldots, s_p)$ be the number of messages sent during the computation of a $(p+2)$-th order median of N on a tree with a p-th level HW partition $P(s_1, s_2, \ldots, s_p)$. The following theorem gives an estimation of $M(s_1, s_2, \ldots, s_p)$.

Theorem 2. $M(s_1, s_2, \ldots, s_p) = O(\mid V \mid \sum_{i=1}^{p} s_i + h \mid V \mid / \prod_{i=1}^{p} s_i)$.

Since $\sum_{i=1}^{p} s_i + h / \prod_{i=1}^{p} s_i \geq (p+1)\, h^{1/(p+1)}$, we have $M(s_1, s_2, \ldots, s_p) = O((p+1) h^{1/(p+1)} \mid V \mid)$. By choosing $s_i = \lceil h^{1/(p+1)} \rceil$ for $1 \leq i \leq p$, we can reach the bound $M(s_1, s_2, \ldots, s_p) = O((p+1)\, h^{1/(p+1)} \mid V \mid)$.

Since a p-th order median of N will remove at least $1/2^p$ of the elements in N, we will obtain the k-th smallest element of N after some finite number of iterations. Let n be the number of iterations. It is clear that there always exists at least one element. Thus $(1 - 1/2^p)^n \mid N \mid \geq 1$. This implies $1 \leq n \leq \log \mid N \mid / \log(2^p/(2^p - 1)) = \log \mid N \mid / (\log 2^p - \log(2^p - 1))$. By the Mean Value Theorem, we obtain $\log \mid N \mid / (\log 2^p - \log(2^p - 1)) = (2^p - \epsilon) \log \mid N \mid$ for some $0 \leq \epsilon \leq 1$.

Combine the upper bound of the number of iteration and the number of messages required in each iteration, we can construct a $(p-1)$-th level HW partition on the tree and using a $(p+1)$-th order median to obtain a

distributed selection algorithm with message complexity $O(p\,2^{p+1}\,h^{1/p}\,|V|\log|N|)$ or $O(p\;2^{p}\;h^{1/p}\,|V|\log|N|)$. By computing the minimum of the convex function $f(p)=p\,2^{p}\,h^{1/p}$, we choose $p=\sqrt{\log h}$ as a good approximation and obtain the following theorem.

Theorem 3. The distributed selection problem can be solved with message complexity $O(\sqrt{\log h}\;2^{\sqrt{\log h}}\;h^{1/\sqrt{\log h}}\;|V|\log|N|)$ or $O(\sqrt{\log h}\;4^{\sqrt{\log h}}\;|V|\log|N|)$.

Since the functions $\sqrt{\log h}$ and $4^{\sqrt{\log h}}$ are both bounded by $O(h^{\epsilon})$ for any small $\epsilon>0$, so the message complexity $O(\sqrt{\log h}\,4^{\sqrt{\log h}}\,|V|\log|N|)$ is bounded by $O(h^{\epsilon}\,|V|\log|N|)$ for any small $\epsilon>0$. The distributed selection algorithm using a $(p+1)$-th order median is listed as follows.

Algorithm Selection-II.

Step 1. Execute the Algorithm Label with input $p=\lceil\sqrt{\log h}-1\rceil$ and $s_i=\lceil h^{1/(p+1)}\rceil$, $1\le i\le p$.

Step 2. Every root node of λ-th level subtree $T_i=(V_i, E_i)$ finds a λ-th order median m_i and send m_i and $|V_i|$ to the root of the $(\lambda+1)$-th level subtrees recursively for $2\le\lambda\le p$.

Step 3. The root node of the tree finds a $(p+1)$-th order median m of N.

Step 4. Execute the Procedure Reduce-and-Find with input m. If m is the k-th smallest element of N then every node will receive the stopping message. If not, the problem size is reduced.

Step 5. If a node has received the stopping message, then it stops. When the root node has stopped, the algorithm is terminated and the k-th smallest element of N is m. If the root node has not stopped, go to Step 2.

Since Step 1 takes only $O(|V|)$ messages, so the message complexity of Algorithm Selection-II is $O(\sqrt{\log h}\,4^{\sqrt{\log h}}\,|V|\log|N|)$.

5. Distributed convex hull problem

This section will give an application of the selection algorithm — the distributed convex hull problem. The convex hull of a finite point set S in the plane, denoted by $CH(S)$, is the smallest convex polygon containing the set. The vertices of this polygon must be points in the set. The convex hull problem is to construct the ordered sequence of vertices of $CH(S)$. The problem has been studied extensively in sequential and parallel environments but not for distributed environments. In this section we show the distributed selection algorithm can be applied to solve the convex hull problem distributively. The distributed convex hull problem is to identify the vertices of the hull where the points are stored among nodes of a network. For a tree network, by transforming the sequential convex hull algorithm [10] to a distributed algorithm and applying the distributed selection algorithm, we get a distributed convex hull algorithm. The message complexity of the algorithm is $O(\sqrt{\log h}\,4^{\sqrt{\log h}}\,|V|\;H\log(2|N|/H))$, where h is the height of the tree, V is the set of nodes in the tree, H is the number of vertices found to be on the hull and N is the set of planar points. If all points in N are allowed to be sent to a particular node, then the convex hull can be trivially constructed in the node. Therefore, there is a trivial distributed convex hull algorithm with message complexity $O(h\,|N|)$. But in general, the nodes' storage and channels' capacities are limited. So all points in N can not be sent to and stored in one node when $|N|$ is large enough. In that case, our algorithm is suitable since only $O(|V|/(h/2^{\sqrt{\log h}}))$ extra storage are need for each node during the execution of the algorithm. When $H<|N|$ and $|V|\ll|N|$, our algorithm is better than the trivial algorithm. For example, if points are chosen independently from a 2-dimensional normal distribution, then the expected message complexity of our algorithm is $O(\sqrt{\log h}\,4^{\sqrt{\log h}}\,|V|\,(\log|N|)^{3/2})$ [11, p. 145].

To solve the distributed convex hull problem, we translate the sequential convex hull algorithm [10] into a distributed algorithm step by step. There are two steps in the sequential algorithm which require finding the median. We replace the sequential selection algorithm by the proposed distributed selection algorithm. If N is the set of planar points stored distributively in the tree network, then these two selection steps will cost $O(\sqrt{\log h}\,4^{\sqrt{\log h}}\,|V|\log|N|)$ messages.

Theorem 4. The distributed convex hull problem can be solved with $O(\sqrt{\log h}\, 4^{\sqrt{\log h}} \mid V \mid H \log(2 \mid N \mid / H))$ message complexity.

Proof. The total message complexity of the distributed convex hull algorithm can be derived by the following equation:

$$f(\mid N \mid, H) = \begin{cases} \text{if } H = 2,\ C \log \mid N \mid \\ \text{if } H > 2,\ C \log \mid N \mid + \max\limits_{H_l + H_r = H} \{f(\mid N \mid /2, H_l) + f(\mid N \mid /2, H_r)\}, \end{cases}$$

where $C = c\sqrt{\log h}\, 4^{\sqrt{\log h}} \mid V \mid$ for some constant c. It follows that

$$f(\mid N \mid, H) = C(H \log \mid N \mid - H \log H + 2H - \log \mid N \mid - 2)$$

$$= O(\sqrt{\log h}\, 4^{\sqrt{\log h}} \mid V \mid H \log(2 \mid N \mid / H)). \square$$

6. Conclusions

In this paper we have discussed the network partition and the distributed selection problem for a general tree network $T = (V, E)$. We propose a tree partition algorithm with message complexity $O(\mid V \mid)$ and an improved selection algorithm with message complexity $O(\sqrt{\log h}\, 4^{\sqrt{\log h}} \mid V \mid \log \mid N \mid)$, which is bounded by $O(h^{\epsilon} \mid V \mid \log \mid N \mid)$ for any small $\epsilon > 0$, where V is the set of nodes in the tree, h is the height of the tree and N is the set of elements distributed among nodes of V. Santoro and Suen [7] has proposed an open problem to find an efficient selection algorithm for general tree networks. Since the message lower bound for the selection problem of complete binary trees is $\Omega(\mid V \mid \log(2 \mid N \mid / \mid V \mid))$ [5], the proposed algorithm gives an asymptotically efficient solution to that open problem. We also propose a distributed convex hull algorithm which is based on the tree partition and selection algorithms. Finally, we remark that the Set-Up Procedure introduced in [7] can be incorporated into our algorithm to remove a certain amount of elements in advance.

References

[1] Baruch Awerbuch, "Reducing complexities of the distributed max-flow and breadth-first-search algorithms by means of network synchronization", *Networks*, vol. 15, 1985, pp. 425–437.

[2] Baruch Awerbuch, "Optimal distributed algorithms for minimum weight spanning tree, counting, leader election and related problems", *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, 1987, pp. 230–240.

[3] Baruch Awerbuch and Robert G. Gallager, "A new distributed algorithm to find breadth first search trees", *IEEE Transactions on Information Theory*, vol. IT-33, no. 3, May 1987, pp. 315–322.

[4] Manuel Blum, Robert W. Floyd, Vaughan Pratt, Ronald L. Rivest and Robert E. Tarjan, "Time bounds for selection", *Journal of Computer and System Sciences* 7, 448–461 (1973).

[5] G. N. Frederickson, "Tradeoffs for selection in distributed networks", *Proc. 2nd ACM Symp. Princ. Distrib. Comput.*, Aug. 1983, pp. 154–160.

[6] Donald B. Johnson and Tetsuo Mizoguchi, "Selecting the Kth element in $X+Y$ and $X_1 + X_2 + \cdots + X_m$", *SIAM J. Comput.*, vol. 7, no. 2, May 1978, pp. 147–153.

[7] Nicola Santoro and Ed Suen, "Reduction techniques for selection in distributed files", *IEEE Transactions on Computers*, vol. 38, no. 6, June 1989, pp. 891–896.

[8] L. Shrira, N. Francez, and M. Rodeh, "Distributed k-selection: From a sequential to a distributed algorithm", *Proc. 2nd ACM Princ. Distrib. Comput.*, Aug. 1983, pp. 143–153.

[9] J. M. Marberg and E. Gafni, "An optimal shout-echo algorithm for selection in distributed sets", *Proc. 23rd Allerton Conf. on Communication, Control and Computing*, Monticello, Oct. 1985.

[10] David G. Kirkpatrick and Raimund Seidel, "The ultimate planar convex hull algorithm?", *SIAM J. Comput.*, vol. 15, no. 1, February 1986, pp. 287–299.

[11] Franco P. Preparata and Michael Ian Shamos, *Computational Geometry: An Introduction*, Springer-Verlag New York Inc., 1985.