THE OPTIMALITY OF DISTRIBUTIVE CONSTRUCTIONS
OF MINIMUM WEIGHT AND DEGREE RESTRICTED
SPANNING TREES IN A COMPLETE NETWORK
OF PROCESSORS

by
E. Korach*, S. Moran** and S. Zaks**

Technical Report #334
August 1984

* IBM-Israel Scientific Center, Technion City, Haifa, Israel 32000
** Computer Science Department, Technion - Israel Institute of
Technology, Haifa, Israel 32000
The Optimality of Distributive Constructions of Minimum Weight and Degree Restricted Spanning Trees in a Complete Network of Processors

E. Korach

IBM Israel Scientific Center
Technion City
Haifa, Israel 32000

S. Moran and S. Zaks

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

ABSTRACT

In a previous paper it was shown that the distributive construction of a spanning tree in a complete network of processors can be done in $O(n \log n)$ messages. We show in this work that if the spanning tree is required to satisfy certain properties, then the complexity of its construction increases: First we show that the construction of a minimum weight spanning tree requires, in the worst case, $O(n^2)$ messages, and then we show that the construction of a spanning tree where the maximum degree is at most $k$ may require $O(n^2/k)$ messages in the worst case. The validity of the results is independent of the length of the messages used. On the other hand, there are algorithms for the above tasks which achieve these lower bounds, up to a constant factor, and use messages of $O(\log n)$ length.
1. INTRODUCTION

The model under investigation is a network of \( n \) processors with distinct identities. Each processor has some communication lines, connecting him to some others (to all others in a complete network); he knows the (non-negative) cost associated with each such line, but not the identities of his neighbors. The communication is done by sending messages along the communication lines. The processors all 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 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 him.

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. An edge in this graph is considered unused during a certain execution of an algorithm as long as no message was sent along it (otherwise it is used).

Working within this model, when no processor knows the value of \( n \), a minimum weight spanning tree (MST) is found in [1] in \( O(n \log n + |E|) \) messages for a general graph. It is pointed out in [1] that for general graphs no algorithm to find a spanning tree (ST) which is more efficient than their algorithm to find a MST is known. In [3] it is shown that \( O(n \log n) \) messages are always sufficient (and - in the worst case - necessary) to construct a spanning tree in a complete network of processors. We show here, that for such a network, the complexities of the following problems are strictly larger then the above \( o(n \log n) \) upper bound:

a) Finding an MST;

b) Finding an ST with maximum \( k \) degree at most \( k = k(n) \), where \( k(n) = o(n / \log n) \).
More precisely, we show that any algorithm for finding an MST in a complete graph uses, in the worst case, \( O(n^2) \) edges, and that any algorithm that finds a spanning tree with maximum degree at most \( k \) uses, in the worst case, \( O(n^2/k) \) edges. In proving these results, no assumptions are made on the length of the messages used by the algorithms. This implies that these lower bounds are also lower bounds on the number of messages used by any algorithm for those problems. The first lower bound is achieved, up to a constant factor, by the algorithm in [1], and the second by a composition of the leader finding algorithm in [3] and a straightforward serial construction.

2. MINIMUM SPANNING TREE

The main result in this section is the following theorem:

**Theorem 1:** Any distributed algorithm to find an MST on a complete weighted graph \( G = (V, E) \) with \( n \) vertices uses, in the worst case, \( O(n^2) \) edges of the graph.

**Proof:** We make use of the following basic property of distributed algorithms in the above model: A vertex cannot distinguish between unused edges of equal costs adjacent to himself (since he does not know the identities of the neighbors across these edges). The proof of this theorem, as well as the one in the next section, are described as a game between the network and an outside adversary performed on the graph \( G \). Whenever a vertex wants to send a message along an unused edge of a certain cost, the adversary chooses such an edge for him (if more than one exists). The goal of the adversary is to demonstrate that the algorithm is not correct if it stops before using \( \binom{m}{2} \) edges, where \( m = \left\lfloor \frac{|V|}{2} \right\rfloor \). The adversary may exchange costs of unused edges as long as the sets of costs of the edges adjacent to each vertex do not change. For example, if we have as part of the graph the subgraph shown in Figure 1a, where the shown edges are unused, then the adversary can change the costs of the edges to these shown in Figure 1b; no vertex should be able to tell that any change was made in the network.
within $Y$ have a cost of $0$, and all the edges connecting $X$ and $Y$ have a cost of $1$. It is important to note that this initial structure of the graph $G$ is known only to the adversary; as far as each single vertex is concerned, he only knows that there are $n$ edges of cost equal $1$ and $n - 1$ edges of cost equal $0$ adjacent to himself.

The adversary can exchange costs of edges as described above. In fact, the adversary will make at most one exchange of costs of the type depicted in Figure 1 (where $z_1, z_2 \in X$ and $y_1, y_2 \in Y$) at the end of the algorithm. Therefore any MST in the resulting graph will have a cost equal $0$, and will use the edge $(x_1, y_1)$ or the edge $(x_2, y_2)$ or both.

When the algorithm starts, some vertex eventually asks for an (unused) edge of cost $0$ or $1$, and the adversary chooses such an edge at random from the initial structure above and gives it to this vertex. This process is repeated each time a vertex asks for an unused edge. When the algorithm stops the set of unused edges does not contain a cycle of four edges like the one shown in Figure 1a, called $XY$-square (where $x_1, x_2 \in X$ and $y_1, y_2 \in Y$) from the following reasons: The MST produced by it uses at least one edge of cost $1$ (connecting $X$ and $Y$), and if such a square exists, then by making at this point an exchange as described above the adversary creates an ST of cost $0$, which proves the algorithm to be incorrect. Thus, the algorithm must use a set of edges that intersects all such cycles before it stops. As we show in the next Lemma, such a set contains at least $\left\lfloor \frac{n}{2} \right\rfloor$ edges. This will complete the proof of the theorem.

Let $\nu$ denote the maximum number of $XY$-squares which are pairwise edge disjoint, and $\tau$ denote the minimum number of edges that meet every $XY$-square.

**Lemma 1:** Let $G = (V,E)$ be a complete graph on $2n$ vertices where $V = X \cup Y$, $X = \{x_1, x_2, ..., x_n\}$ and $Y = \{y_1, y_2, ..., y_n\}$, $n \geq 2$. Then

(i). $\nu = \tau = \left\lfloor \frac{n}{2} \right\rfloor$

(ii). $E(X)$ and $E(Y)$ are the only sets of cardinality $\tau$ that meet all $XY$-squares.
Proof:

(i) Clearly \( E(X) \geq T \geq \nu \). It is easy to see that \( S \) is a collection of pairwise edge-disjoint XY-squares, where:

\[
S = \{(x_i, y_i), (y_i, x_i), (y_j, x_i), (x_i, y_j) \mid (x_i, y_i) \in E(X)\}
\]

The cardinality of \( S \) is \( |E(X)| \), which implies that \( \nu \geq |E(X)| \). We have:

\[
|E(X)| \geq \nu \geq |E(X)| \geq \nu \geq \left[ \frac{n^2}{2} \right].
\]

It follows that \( \nu = T = \left[ \frac{n^2}{2} \right] \).

(ii) Let \( T \) be a minimum set of edges that meets all XY-squares (i.e. \( |T| = \tau \)). If \( T \) contains an edge \( e \) from \( X \) to \( Y \), relabel \( X \) and \( Y \) (if necessary) so that \( e = (x_i, y_i) \). Clearly, \( e \) does not meet any XY-square in \( S \), hence \( |T| > \left[ \frac{n^2}{2} \right] \), a contradiction. If \( T \) meets both \( E(X) \) and \( E(Y) \), then (since \( |E(X)| = |E(Y)| = |T| \)) we have an edge in \( E(X) \) and an edge in \( E(Y) \), both not in \( T \). It follows that these two edges are in an XY-square that does not meet \( T \). This implies (ii).

\[ \square \]

Remark 1: In the case where all the edge-weights are distinct, and arbitrarily long messages are allowed, an MST can be constructed in the following way:

1. Choose a leader using the algorithm in [3] \((O(n \log n) \) messages).

2. Each node transfers to the leader the weights of the edges adjacent to him \((O(n) \) "long" messages).

3. The leader reconstructs the weighted graph by assigning to the edge \((i, j)\) the unique weight that was transferred to him by both \( i \) and \( j \).

4. The leader finds the MST of \( G \), and transfers to each node the weights of the MST edges adjacent to it \((O(\eta) \) messages).

The above remark can be generalized to show that if we have in a given graph \( G \) \( k \) edges of non-distinct weights, then an MST of \( G \) can be constructed in \( O(\max(k, n \log n)) \) "long" messages.
3. DEGREE RESTRICTED SPANNING TREES

Spanning trees in distributed networks are generally used to simplify the communication in the network. If in such a tree the degree of a node is large, it might cause an undesirable communication load in that node. In such cases, the construction of spanning trees in which the degree of a node cannot exceed some given value \( k \) (which might depend on the size of the network \( n \)), is needed. The known \( O(n \log n) \) algorithms for constructing a spanning tree in a complete network of processors may not meet such a requirement. In fact, the following theorem shows that for \( k = o(n/\log n) \) any \( O(n \log n) \) algorithm must fail in certain cases. It should also be noted that, unlike the MST problem, the corresponding problem for general graphs is NP Hard ([2]). This makes the focus on complete networks more interesting.

**Theorem 2:** For all \( n \) and \( k \), \( 1 < k < n \), any distributed algorithm that constructs a spanning tree with maximum degree bounded by \( k \) in a complete graph on \( n \) vertices, uses, in the worst case, at least \( O(\max \{n \log n, n^2/k\}) \) edges.

**Proof:** The \( O(n \log n) \) lower bound follows from the lower bound on the number of messages required for the construction of any spanning tree [3]. It remains to prove the \( O(n^2/k) \) lower bound for the case where \( k = n/\log n \). In the rest of the proof we assume this case, and we again use the adversary approach. We also assume that any algorithm for this problem uses all the edges of the constructed tree (this might increase the number of edges used by it by at most \( n - 1 \)). Let \( A \) be such an algorithm, and assume that the vertices of the graph are \( \{1, \ldots, n\} \). Whenever a vertex \( j \) wishes to send a message along an unused edge, the adversary supplies him with the edge \((j, i)\) with the smallest possible \( i \).

Assume that the required tree \( T \) was constructed by the algorithm. We take vertex \( 1 \) to be the root of \( T \) and direct the edges to get an arborescence rooted at \( 1 \) (i.e., there is a directed path from \( 1 \) to every vertex in \( T \)). Let \( D \) be the set of edges of \( T \) on which the first message was sent in the direction of the edge, and let \( B \) be the set of the remaining tree edges. Let \( d = |D| \) and \( b = |B| = n - 1 - d \).
Assume first that \( d \geq b \) (and hence \( d \geq (n-1)/2 \)), and let \( S = \{ s_1, \ldots, s_{|S|} \} \) be the subset of vertices which are a tail of an edge in \( D \). For each \( s_i \in S \) let \( m_i \) be the maximal vertex \( h(s_i) \) to which an edge of \( D \) is directed from \( s_i \). Without loss of generality assume that \( m_1 \leq m_2 \leq \cdots \leq m_{|S|} \).

**Claim 1:** \( m_{|S|} \geq d+1 \).

**Proof:** This follows from the fact that in \( T \) the in-degree of each vertex, other than the root, is one, and hence \( m_{|S|} \) is the maximal integer in a set of \( d \) positive integers where the minimal is at least 2 (since 1 is the root and has in-degree zero).

**Claim 2:** For \( i = 1, \ldots, |S|-1 \), \( m_{|S|-i} \geq d-i \cdot k+1 \).

**Proof:** At most \( k \) edges in \( D \) leave each vertex in \( S \). Hence \( m_{|S|-i} \) is the maximal vertex in a set of at least \( d-i \cdot k \) vertices, which, again, does not contain the root.

By the adversary rule and the definition of \( m_i \), at least \( m_i - 1 \) edges adjacent to \( s_i \) were used (by a message to or from \( s_i \)). Thus, the total number of used edges adjacent to vertices in \( S \) is at least

\[
\frac{k}{2} \sum_{i=1}^{|S|} \max\{d-2k \cdot (i-1)+1-1, 0\}
\]

\[
\geq \frac{k}{2} [d+(d-k)+(d-2k)+\cdots+d \mod (k)]
\]

\[
= O(d^2/k).
\]

\[
= O(n^2/k) \quad \text{(since } d \geq (n-1)/2\).
\]

This completes the proof in the case where \( d \geq b \).

Assume now that \( b > d \), and let \( U = \{ u_1, \ldots, u_{|U|} \} \) be the set of vertices which are a head of an edge in \( B \). Let \( n_i \) be the maximal vertex from which an edge of \( B \) is directed to \( u_i \), and assume that \( n_1 \leq n_2 \leq \cdots \leq n_{|U|} \). Since the degree of each vertex is at most \( k \), we have that:

\[ i \leq n_1 \leq n_2 \leq \cdots \leq n_k. \]
\[2 \leq n_{i+1} \leq n_{i+2} \leq \cdots \leq n_{2k},\]
\[i+1 \leq n_{i+1} \leq n_{i+2} \leq \cdots \leq n_{(i+1)k},\]
\[t+1 \leq n_{t+1} \leq n_{t+2} \leq \cdots \leq n_b,\]

where \(t\) is such that \(b = tk + r, 1 \leq r \leq k\).

Using the adversary rule again, we get that the total number of edges adjacent to nodes in \(U\) that were used by the algorithm is at least

\[
\frac{1}{2k}(1+t+2t+\cdots+t) = O(k t^2) = O(k (b/k)^2)
\]

\[= O(n^2/k) \text{ (since } b > (n-1)/2).\]

This completes the proof of the theorem.

\[\square\]

Remark 2: A spanning tree with maximum degree at most \(k\) can be constructed in the following way:

1) Choose a leader using the algorithm in [3] \((O(n \log n)\) messages).

2) The leader chooses \(k\) descendants and asks one of them, say \(v_1\), to choose \(k-1\) yet unchosen descendants.

3) \(v_1\) chooses its descendants by serially asking its neighbors to be its descendants, and waiting for their positive or negative response. Then \(v_1\) asks one of its chosen descendants, say \(v_2\), to choose \(k-1\) yet unchosen descendants, and so on.

Clearly, the leader sends \(k\) messages, \(v_1\) sends less than \(2k\) messages (and receives less than \(2k\) responses), \(v_2\) sends less than \(3k\) messages (and receives less than \(3k\) responses), etc. The total number of messages sent at phases 2) and 3) is less than \(k + 4k + \cdots + \left\lfloor \frac{n}{k} \right\rfloor k = \frac{n^2}{k} + O(n)\) (and the total number of edges used is at most \(\frac{n^2}{2k} + O(n)\)).
Remark 3: A more detailed analysis in the proof above shows that the number of edges that must be used by the algorithm is at least $n^2/8k - n/2 + k/2$.

Acknowledgement: We would like to thank Reuven Bar-Yehuda for some helpful comments.
REFERENCES

