International Journal of Foundations of Computer Science

THE FIRST APPROXIMATED DISTRIBUTED ALGORITHM FOR THE MINIMUM DEGREE SPANNING TREE PROBLEM ON GENERAL GRAPHS

LÉLIA BLIN
LIPN, CNRS-UMR-7030, Université Paris-Nord,
Avenue Jean-Baptiste Clément
93430 Villetaneuse, France.

and

FRANCK BUTELLE
LIPN, CNRS-UMR-7030, Université Paris-Nord,
Avenue Jean-Baptiste Clément
93430 Villetaneuse, France.

Received
Revised
Communicated by

ABSTRACT

In this paper we present the first distributed algorithm on general graphs for the Minimum Degree Spanning Tree problem. The problem is NP-hard in sequential. Our algorithm give a Spanning Tree of a degree at most 1 from the optimal.
The resulting distributed algorithm is asynchronous, it works for named asynchronous arbitrary networks and achieves $O(|V|)$ time complexity and $O(|V|\,|E|)$ message complexity.

Keywords: Distributed algorithms; Spanning trees; Minimum degree spanning trees; Asynchronous algorithms

1. Introduction

In this paper we focus on the problem of finding a distributed approximated algorithm for a Minimum Degree Spanning Tree (MDegST). The spanning tree construction is a fundamental problem in the field of distributed network algorithms. A tree is an essential structure in various communication protocols, e.g Network Synchronization, Bread-First-Search and Deadlock Resolution. 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 case, the construction of spanning trees in which the degree of a node is the lowest as possible (or cannot exceed a given value k which might depend on the size of the network n), is needed.

Solutions to the broadcast problem have mainly concentrated on how to complete the broadcasting as quickly as possible. But, sites may want to reduce the amount of work done by their site. Broadcasting information on a MDegST is one such solution.

In this paper, we consider the problem of finding the Minimum Degree Spanning Tree. Let $G = (V, E)$ be a graph, with $n = |V|$ vertices and $m = |E|$ edges. A spanning tree $T = (V, E_T)$ of G is a tree such that $E_T \subset E$. Among all spanning trees of G, we search for those of minimal degree (the degree of a graph is equal to the maximum degree of its nodes). This problem has been proved to be NP-hard, since it is a generalization of the hamiltonian path problem, see [1].

In a distributed algorithm, each node in V is associated with its own processor, and processors are able to communicate with each other via the edges in E, each edge is associated with a bidirectional link. A common goal for distributed algorithms is to have the nodes cooperate in order to construct a structure, for instance a tree.

The MDegST problem has been studied before, but mainly in a sequential point of view. To our knowledge, there is only one article addressing the problem in a distributed system: it is the work of Korach, Moran and Zacks on complete graphs (see [2]). They show that any algorithm that constructs a spanning tree with maximum degree at most k uses at least, in the worst case, $O(\frac{n^2}{k})$ messages. In sequential, Furer and Raghavachari [3] propose an heuristic for the MDegST whose degree is from one to the optimal degree. Their work starts with an arbitrary spanning tree and improves locally by edges' exchange the maximum degree of that tree. They show that the best bound achievable in polynomial time is the optimal degree plus one. Their heuristic is simple, it uses a local optimality property: no edge (not belonging to the Spanning Tree) should be able to reduce the maximal degree of vertices on the basic cycle induced by that edge.

In this paper, we present a distributed algorithm based on the main ideas of [3]. As a startup process, we need a Spanning Tree. For constructing such a tree, many different distributed algorithms exist, see for example Minimum-weight Spanning Tree algorithms [4, 5, 6, 7, 8, 9], or DFS trees [10], etc.

Our results in term of complexity are function of the degree k of the initial spanning tree and of the degree k^* of its corresponding Locally Optimal Tree. We obtain an algorithm in $O((k-k^*)m)$ messages and $O((k-k^*)n)$ units of time. These results show that at least in term of messages we are not far from the optimal of [2].

2. The Model

Now, we consider the standard model of static asynchronous network. This is a point-to-point communication network, described by an undirected communication graph (V, E) where the set of nodes V represents network of processors and the set of edges E represents bidirectional non-interfering communication channels operating between neighboring nodes. No common memory is shared by the nodes' processors.

All the processors have distinct identities. However each node is ignorant of the global network topology except for its own edges, and every node does know identity of its neighbors. This assumption is only used to simplify algorithm presentation, knowing neighbor's identities is not an essential requirement, since every node can send its identity to its neighbors in the first message.

We confine ourselves only to event-driven algorithms, which do not use time-outs, i.e. nodes cannot access a global clock in order to decide what to do. This is a common model for static communication networks [4].

The following complexity measures are used to evaluate performance of distributed algorithms. The Message Complexity is the worst case total number of messages exchanged. The time complexity is the maximum possible number of time units from start to the completion of the algorithm, assuming that the inter-message delay and the propagation delay of an edge is at most one time unit of some global clock. This assumption is used only for algorithm analysis, but can not be used to prove its correctness, since the algorithm is event-driven.

In a distributed algorithm for the Minimum Degree Spanning Tree problem, each node has a copy of a node algorithm determining its response to every kind of message received. Namely, the algorithm specifies which computation should be done and/or which message should be sent. The algorithm is started independently by all nodes, perhaps at different times. At the start time, each node is ignorant of the global network topology except for its own adjacent edges. Upon the termination of the algorithm, every node knows its neighbors in an approximated Minimum Degree Spanning Tree.

3. Our Algorithm

3.1. General Method

We now describe the general method we use to construct a MDegST. We suppose a Spanning Tree already construct, and almost all spanning tree construction algorithm gives a root of the tree.

Our algorithm is divided into rounds, each round consists in the following steps :

- **SearchDegree:** Search for the maximum degree k of the nodes in the ST.
- **MoveRoot:** Move the root to the node of maximum degree and minimum identity. Let p be that node. In the following we will suppose node p to be the only one of degree k in order to simplify the description of the method.

- **Cut:** Virtually cut the children x of the root in the ST. Each x is now the root of a subtree called a fragment of the ST.
- **BFS** Each x starts a Breadth First Search (also called BFS wave) for outgoing edges of its fragment. When completed, x forwards its best outgoing edge to p.
- **Choose** p chooses one of the outgoing edges to make an exchange of edges. Doing such, it decreases its degree k (see Figure 1).

The algorithm performs these previous steps until no improvement is found or $k = 2$ (the tree is a chain).

When there is more than one node of degree k, these nodes behave like roots except that they have to send back a message toward the real root of the Spanning Tree.

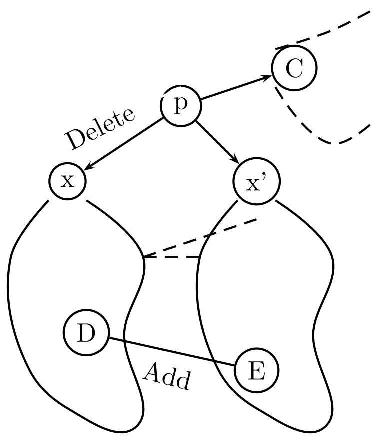

Fig. 1. The tree's maximum degree is improved

3.2. Precise Description

We suppose a spanning tree already constructed. We need the algorithm that constructs that tree to terminate by process (that is to say that every node knows that the ST algorithm is terminate).

We will now describe precisely each step of an algorithm round for any node x.

3.2.1. SearchDegree

Each node cooperates in order to find the maximum degree in the ST. At the end of this step the root knows the maximum degree of the initial spanning tree. This step proceeds in the following way:

- Every leaf x of the ST sends a message with its degree in the tree to its parent.
- Each node x receiving the degrees of all its children keeps the maximum degree k of its children and itself and by which node it received that degree. If two

or more nodes have the degree k then x chooses the one of minimum identity. Thus x sends the maximum degree to its parent.
- This process is continued until the messages are send to the root of ST. The root chooses between the maximum degrees received has before.

3.2.2. MoveRoot

The root moves down the tree toward the maximum degree node p of minimum identity. During the moving, the path is reversed (see "Path Reversal" techniques in [11]). At the end of this step the new root is p.

- When a node receives a message "Move root" it starts this subroutine.
- There exist two cases:
  - The degree of the node x is k and its identity is the one we look for. This step is done.
  - else the root knows the identity of its neighbor y who sent the maximum degree to it (in the search step, each node keeps, in a variable named "via", by which processor arrived the maximum degree with minimum identity).
    - Neighbor "via" becomes the parent of node x.
    - x sends a message "Move Root" to "via".

3.2.3. Cut

In the following step the root p virtually cuts the link with its children with an aim of doing to decrease its degree. The name of this step is "Cut".

- The root p (node of degree k), send to its children a message $< cut, k, p >$.

3.2.4. BFS

Every node x receiving $< cut, k, p >$ starts a BFS wave on its children by sending a $< BFS, k, p, x >$ message. In figure 2, the dashed arrows represent the "cousin" message that is to say the BFS wave has discovered an outgoing edge of the tree that can improve the maximum degree.

- **BFS**
  - A node x receiving a $< BFS, k, p, p' >$ message from its parent, note its fragment identity (p, p') and next broadcast this BFS to all its neighbors except its parent. If there were responses waiting to be sent, the responses are sent following the rules of the following item
  - When a node x receives a $< BFS, k, r, r' >$ message from a neighbor y (it cannot be from a children) three cases arise. First case, x does not have already been touched by a BFS wave from its parent then the answer has to be delayed until x learns its fragment identity. Second case, x has already received a $< BFS, k, p, p' >$ message from its parent

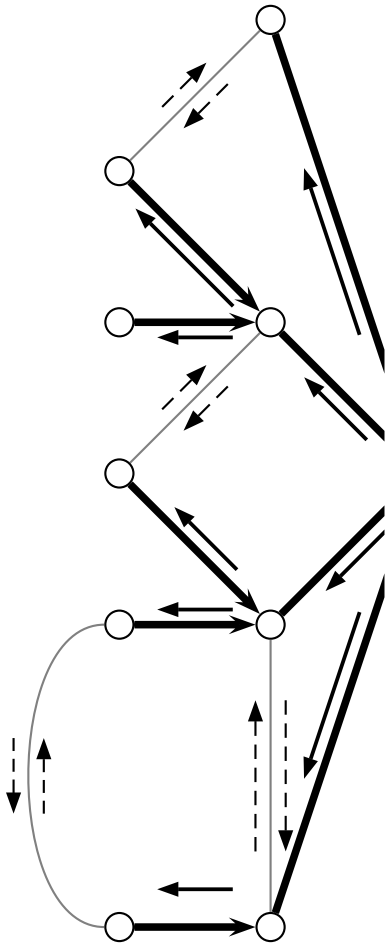

Fig. 2. BFS wave

and $(r, r') < (p, p')$ (i.e. $r < p$ or ($p = r$ and $r' < p'$)). In that case x reply to y by a messsage $< BFSBack, r, r', deg(x), () >$ (the last empty parenthesis are used to carry an interesting outgoing edge from parent to parent). Third case is almost the same as the previous one except that $(r, r') > (p, p')$. In that case x anyway will send or has already sent a BFS message to y and so it can ignore this message.

- **BFS-Back**
  - When a node x received an answer from all its neighbors:
  - The node x chooses an outgoing edge between the edges received by the "BFSBack" messages. These edges are the edges which can decreases the degree of p. Please note that nodes of degree $k-1$ cannot be considered because they would not improve the maximum degree of the tree (we obtains a maximum degree by the addition of an edge).
  - The node x sends a message "BFSBack" with the chosen edge (may be none) to its parent.
- The BSF-Back subroutine is repeated until the node which started the BFS.
- The node which started the BFS sends a message with the chosen edge to the node p.

3.2.5. Choose

If there is no more outgoing edge, that means the maximum degree cannot be (locally) improved then the algorithm stops.

Else the root p choose an edge between the outgoing edges received from its children. The child which sends the best outgoing edge will be suppressed from the children set and the best outgoing edge will be the one which will connect the two subtrees.

- The root p chooses the outgoing edge e whose maximal degree of its extremities is minimal. Let y be the identity of the node which sent the chosen edge.
- y is suppressed from the children set
- The root p send a message $< update, e >$ to y
- When a node x received this update message from its parent:
  - If e is an outgoing edge of x:
    . Let y be the node at the next extremity of e. It becomes the parent of x.
    . The node x sends the message "child" to y.
    . Upon receipt of the "child" message from x, the node y adds x to its children set.
  - Else: the identity found in its "via" variable becomes its parent and the same identity is suppressed from the set of its children.
- A round is terminated when a node received a "child" message.

3.2.6. More Than One Node of Maximum Degree

The initial spanning tree can include more than one node of maximum degree. In this case when a BFS wave meets a maximum degree node, this node follows the root behavior. At the end of its algorithm, this node had two possible behaviors:

- If there is no outgoing edge, that means that the maximum degree of this spanning tree cannot be improved, this node send a message "stop" to its parents. The "stop" message is forwarded to the root, because the maximum degree of the spanning tree cannot be improved.
- Else there exist two edges, one edge improves this sub tree, and the other edge can improve the tree. In this case an "update" message is sent to the child which improves the subtree and a "BFS-Back, edge" is sent to its parent. (If there exist no second edge, the "BFS-Back" is sent empty.)

4. Analysis

4.1. Correctness Proof

In [3], the problem was addressed in sequential and their key result is in the following theorem :

Theorem 1 *Let T be a spanning tree of degree k of a graph G. Let Δ^* be the degree of a minimum degree spanning tree. Let S be the set of vertices of degree k. Let B be an arbitrary subset of vertices of degree $k-1$ in T. Let $S \cup B$ be removed from the graph, breaking the tree T into a forest F. Suppose G satisfies the condition that, there are no edges between trees in F. Then $k \leq \Delta^* + 1$.*

When there are no edges between trees in F the tree T found at that point of the algorithm is called a Locally Optimal Tree (LOT).

Our algorithm correctness follows from this previous theorem: the BFS wave, started from the neighbours of each node of degree k, found every edges between trees in F. Outgoing edges from nodes of degree $k-1$ are simply ignored.

4.2. Complexity Analysis

We will describe the complexity of the algorithm using the commonly defined message and time complexity. The message complexity is the number of exchanged messages. It can sometimes be more precise to use the "bit-complexity" that is to say the total number of bits exchanged during the algorithm. In this algorithm, all messages are of size $O(\log n)$ where n is the number of nodes (at most four numbers or identities by message).

One of the common definition of the time complexity is the length of the longest causal dependency chain. That definition means that every message transmission delays are almost equal to one unit of time. Please note that for the sake of computation of the time complexity it is easier to consider a kind of synchronization of the network but our algorithm is definitely asynchronous.

We first express the two complexities during one round for each step described in the previous paragraphs.

- **SearchDegree** Search for the maximum degree k of the nodes in the ST.
  Since we already have a ST, we use it for the search, so only $n-1$ messages are needed.
- **MoveRoot** Move the root to one node of maximum degree.
  At most $n-1$ messages are used during this step (if the ST is a chain).
- **Cut** Virtually cut the children x of the root in the ST. Each x is now the root of a subtree called a fragment of the ST.
  We will count this step with the following one.
- **BFS** Each x starts a Breadth First Search (also called BFS wave) for outgoing edges of its fragment. After that, x forwards its best outgoing edge to p.
  A BFS wave has to cover the whole graph. Each edge of the graph will be seen at most twice : one for the BFS (or cut) and one for the BFS-back. So in this step (added to the previous one) at most $2m$ messages are send.
- **Choose** p chooses one of the outgoing edges to make an exchange of edges.
  p has to send a message toward one of the extremity of the outgoing edge. At most $n-1$ messages are used in this step for all nodes of maximum degree.

All in all, $O(m+n) = O(m)$ messages are send during one round in time complexity in $O(n)$.

The number of rounds depends on the degree of the ST constructed as a first step before our algorithm. Let k be the degree of the initial spanning tree and let k^* be the degree of its corresponding Locally Optimal Tree (LOT). There is $k-k^*+1$ rounds. The maximum value for k is equal to $n-1$ (the ST is a star) and the minimum value of k^* is 2 (the LOT is a chain). This gives a worst case in $O(mn)$ messages exchanged. Of course we can hope to change a bit the algorithm of ST construction in order to obtain a not so bad k.

5. Conclusion

We have presented in this article the first approximated distributed, minimum degree spanning tree algorithm that works on general graphs. It uses a Spanning Tree construction as a first step, and exchange a number of messages that is quite reasonable since any algorithm cannot exchange less than $O(\frac{n^2}{k})$ [2].

References

1. M. R. Garey and D. S. Johnson, *Computers and intractability: A guide to the theory of NP-completeness* (W. H. Freeman, 1979).
2. E. Korach, S. Moran, and S. Zaks, "The optimality of distributed constructions of minimum weight and degree restricted spanning trees in complete network of processors," *SIAM J. COMPT.* **16** (1987) 231–236.
3. M. Fürer and B. Raghavachari, "Approximating the minimum degree spanning tree to within one from the optimal degree," *ACM-SIAM Symposium on Discrete Algorithms* (1982) 317–324.

4. R. G. Gallager, P. A. Humblet, and P. M. Spira, "A distributed algorithm for minimum weight spanning trees," *ACM Trans. Prog. Lang.* **5** (1983) 66–77.
5. B. Awerbuch, "Optimal distributed algorithms for minimum weight spanning tree, counting, leader election and related problems," *ACM Symp. Theory Comp.* (1987) 230–240.
6. F. Y. Chin and H. F. Ting, "An almost linear time and $o(nlogn + e)$ messages distributed algorithm for minimum-weight spanning trees," *IEEE Symp. Found. Comput. Sci.* (1985) 257–266.
7. M. Faloutsos and M. Molle, "Optimal distributed algorithm for minimum spanning trees revisited," *Symposium on Principles of Distributed Computing* (1995) 231–237.
8. J. A. Garay, S. Kutten, and D. Peleg, "A sublinear time complexity distributed algorithm for minimum-weight spanning trees," *SIAM J. Comput.* **27** (1998) 302–316.
9. L. Blin and F. Butelle, "A very fast (linear time) distributed algorithm, in general graphs, for the minimum spanning tree," *Studia Informatica Universalis, Hors Srie OPODIS'2001* (ISBN:2-912590-14-0, 2002) 113–124.
10. G. Tel, *Introduction to Distributed Algorithms* (Cambridge University Press, 1994).
11. C. Lavault, *Evaluation des algorithmes distribués* (Hermes, 1995).