GLOBAL JOURNAL OF ENGINEERING SCIENCE AND RESEARCHES

MINIMUM SPANNING TREE ALGORITHM: DESIGN AND ANALYSIS

Nassiri Khalid[*1], EL Hibaoui Abdelaaziz[2], and and Hajar Moha[3]

[*1]Faculty of Sciences and Technologies - My Ismail University, Errachidia

[2]Faculty of Sciences - Abdelmalek Essaâdi University, Tetuan

[3]Faculty of Sciences and Technologies - My Ismail University, Errachidia

ABSTRACT

In this paper, we introduce and analyze a probabilistic distributed algorithm for a construction of a minimum spanning tree on network. This algorithm is based on the handshake concept. Firstly, each network node is considered as a spanning sub-tree. And at each round of the execution of our algorithm, spanning sub-trees are merged. The execution continues until all spanning sub-trees are merged into one. We proof that spanning tree obtained is minimal and characterize the expected number of phases required to reduce the number of spanning tree form n to 1. We show that this expected number is bounded between O (log2 (n)) and O(n).

Keywords- *Spanning tree, Distributed Algorithm, Handshake Algorithm, Matching, and Probabilistic Analysis.*

I. INTRODUCTION

It goes without saying that information networks are pervasive in our daily lives. These networks provide several services (sharing data archives, videos, music,...) which require the exchange of information between autonomous units constituting the network calculations. Such exchange of information must be optimized to avoid overloading communication links leading to network paralysis. To send an information unit to all others, so we wish to minimize the number of messages sent and avoid redundancy. The structure of communication the most used that meet these criteria being the tree. In addition, the network units and/or communication links, at present, are characterized by their dynamicity over time.

A network is usually represented by a graph where the nodes represent the units of networks and the communication links are represented by the edges.

A dynamic network is modeled by a dynamic graph [1, 2], its vertices and its edges appear and disappear [3]. Also if the weight of the edges change, (a weighted graph) over time, the graph is dynamic [4]. This kind of graph is simply a sequence $(\mathbf{G_t})$ $t\geq 0$ of a static graphs on the same set of vertices, where $t\in N$ (to indicate that we consider discrete view even if the graph can evolve continuously). The dynamic graph is noted by g.

We will use, in the remainder of this work, the static graph $\mathbf{G_t}$ which is an image of dynamic graph g at time t. Let Δt being the lifetime of the static graph $\mathbf{G_t}$, our task is focused on the construction of a minimum spanning tree of the graph **G** by assuming that t_c the time of its construction is pretty less than Δt, in other words, there exists a real $\propto$ such that $\propto \ll \Delta t$ then $t_c \leq \propto$.

The paper is organized as follows: Section 2 presents some notations and definitions necessary for understanding the rest of the document. It exposes also our model and assumptions. Section 3 is devoted to describe our algorithm for the minimum spanning tree construction. The analysis of this algorithm will be given in Section 4 and Section 5. Finally, Section 6 concludes the paper and presents our further work.

II. DEFINITIONS AND MODEL

A) Definition and notions

Consider a graph **G= (V, E)** with n vertices (n=|**V**|) and a set of edges **E**.

A spanning tree is a sub graph which will maintain one path between any pair of vertices. Moreover, this covering structure is in terms of the minimum number of edges used, no scan cycles is formed. Those two properties make that a structure of trees is the most used structure in communication networks. This structure is very often the basic brick on whicsubh many protocols centralized or distributed are leaned. For more information see [5, 6, 7, 8].

A matching of **G** is a subset of edges **M⊂ E** such that each vertex is incident with at most one edge in **M**. In other words the edges in the matching **M** do not touch each other. Every edge of **M** is called a matched edge and an edge in the set **E\M** is called an unmatched edge. For more information see [9].

A maximal matching is a matching **M** of a graph **G** with the property that if any edge not in **M** is added to **M**, it is no longer a matching, that is, **M** is maximal if it is not a proper subset of any other matching in graph **G**.

The size of the matching, |**M**|, is the number of edges in **M**. We denote the size of maximum possible matching by $|\mathbf{M}^*|$. The trivial relation $|\mathbf{M}^*| \leqslant n/2$ follows from the definition. If a maximum matching covers all the vertices, $|\mathbf{M}^*|=n/2$. The perfect matching is noted $\mathbf{M}^*$.

B) Model and generality

We consider a network N with $n \geq 1$ entities. T is
a task to be performed on N and $A_p = (A_i)_{1\leq i\leq n}$ a probabilistic algorithm to achieve the task T. We define the random variable (r.v.) X as the number of n iterations for the algorithm $(A_i)_{\forall i\in\{1,2,\cdots,n\}}$ is complete the stage t. The average complexity of the algorithm A_p is the expectation value of X .
In the analysis of probabilistic algorithms, it is mainly concerned with expectancies X . However, a more refined analysis, and sometimes more complex, allows to calculate, at least asymptotically, the distribution of X. In many cases, we can show that the probability that X exceeds C(n) (a function of size n of the network) is bounded by $1/n^{\propto}$ with $\propto \geq 1$. We then say that $X\leq C(n)$ with high probability[10][11].

III. MINIMAL SPANNING TREE

A) Algorithm

Initially each vertex is itself a *cluster*. The virtual graph **G** is a graph of clusters. The latter consists of at least one vertex or set of vertices linked together, without cycle, by a links named *intra-cluster edge*. A connection between two clusters being the minimum weight edge between the set of vertices of these two clusters is called the *inter-cluster edge*, which becomes an intra-cluster edge after the merger of the two clusters into a single cluster by applying the handshake algorithm.

The matching process operates on the clusters. So, at each round, clusters are coupled. Two coupled clusters are merged into a single cluster by applying the handshake algorithm. Therefore, only the minimum connections with the other clusters will be retained.

The algorithm terminates when it remains only one cluster.

Everything that proceeds proves to us that the intra-cluster edges on which handshakes are occurred constitute the edges of the minimum spanning tree reached.

The following algorithm describes the construction of a minimum spanning tree. It is started by each vertex of the graph. We recall that each vertex is considered in itself as a cluster.

Algorithm 1: Minimum Spanning Tree Algorithm (MST-Algorithm).
In parallel, each cluster *C* executes the following instructions:
Begin
While my degree is greater than 0 **do** call Handshake procedure;
if handshake is occurred **then**
•merge me with the other matching cluster;
• restore the minimum edges with other clusters;

Algorithm 2: Handshake Procedure
Each cluster *C* executes forever the following steps during each unit time (round):
C waits until one of the following three events occurs:
• t = *time* (e_i) and *C* has not received any signal from any other cluster neighbor;
C sends 1 on e_i and 0 on all other incidents edges ($\neq e_i$);
• *C* receives 1 from a neighbor *C'*;
C sends 0 to all its other neighbors;

(* There is a handshake between *C* and *C*'.*)
return *C* ;
• t = 1;
(* The round terminates without C taking part in a handshake.*)
Since time is continuous, with probability 1 one and only one of these events will occur.

IV. Handshake procedure analysis

A) Handshake number

Throughout this paper **G = (V, E)** is a simple undirected graph. We refer to n = |**V** | ≥ 2 as the size (or order) of **G**. To avoid the triviality, we suppose that m = |**E**| ≥ 1.
The algorithm starts with the generation of 2|**E**| independent uniform real random variables in the interval [0, 1]: two r.v. for each edge. We can assume that all (2|**E**|)! Orderings on the set of these real numbers have the same probability. This is the main hypothesis in the sequel.
In fact, for this assumption to be valid, one only has to postulate that, for each edge e={u, v} in **G**, the algorithm generates two continuous r.v. $r_e(u)$ and $r_e(v)$, corresponding to $d_u(v)$ and $d_v(u)$ in the algorithm, supposing that these r.v. associated with edges are all independent and identically distributed. The first matching takes place on the edge e ={u, v}, if one of two associated r.v., $r_e(u)$ or $r_e(v)$, is minimal in the whole graph. Thus, for the first handshake on **G**, all edges have the same chance 1/|**E**| to be chosen.
The assignment of the first meeting to u and v on {u, v}, involves that these vertices are removed with their incident edges and, then, the process continues on the new graph (preserving the random generations for the remaining edges) until no edges remain in the set of edges.
The handshake number in a round is simply the total number of edges to which a handshake is assigned. Let *M*(**G**) be this number whenever our algorithm is applied to the graph **G**. This is an integer valued r.v. It takes the value 0 with probability 1 if **E**=∅, the value 1 with probability 1 if |**E**|=1. In general it takes a value ranging over the set of all cardinalities of maximal (in the inclusion sense) matching in **G**, with some probability.
It seems useful to note the following fact which is easy to prove:

Fact1. Let the first matching be assigned to e={u,v}. Let $\mathbf{G}_e$= (**V**, **E**) be the graph obtained by dropping vertices u and v and all incident edges from **G**. Then, all (2|**E**|)! orderings of generated r.v. (two per edge) in $\mathbf{G}_e$, conditioned by the minimality $r_e(u)$ or $r_e(v)$, have the same probability 1/(2|**E**|)!. This means that keeping the 2|E| r.v. for the edges of $\mathbf{G}_e$ is equivalent to starting the algorithm on $\mathbf{G}_e$ with new random generation.
This fact allows us to compute the probability distribution of of *M*(**G**) as follows. Let **G = (V,E)**. If **E**= ∅, we have *M*(G) = 0. We can easily prove the following:

Proposition1. Let $M = \{e_1, e_2, \ldots, e_k\}$ a maximal matching, the probability of having a handshakes on *M* is expressed as follows:

$$Pr(M) = \prod_{j=1}^{k} \frac{1}{m - \sum_{j=1}^{i-1} |Ejej|}$$

Where E_{ei} the set of the egde $e_i=(u_i, v_i)$ and its incident edges, then $|E_{ei}| = d(u_i)+d(v_i)-1$.

Proposition2. For an integer k, a k-maximal matching on **G** is a maximal matching of size k. Let M_k the set of k-maximal matching. The probability to have k handshakes on M_k is given by the following formula:

$$Pr(Mk(G) = k) = \sum_{M \in Mk} Pr(M)$$

Proposition3. For **G = (V, E)**, let the r.v. $M_k(e)$ which takes 1 if a matching is occurred over the edge and e is 0 if not. The probability that the handshake takes place on the fixed edge e is given by the following formula:

$$Pr(Mk(e) = 1) = \sum_{M \in Mk} \frac{1}{m} + \sum_{i=2}^{k} \prod_{j=1}^{i-1} \left(1 - \frac{1}{mj}\right) \frac{1}{mi}$$

Where $mj = m - \sum_{l=1}^{j-1} |Eel|$

and $mi = m - \sum_{j=1}^{i-1} |Ejej|$

V. ANALYSIS OF OUR ALGORITHM

A) Expected number of phases

In a graph $\mathbf{G_c}=(\mathbf{V_c},\mathbf{E_c})$ of clusters, a process of a minimum spanning tree construction can be seen simply as a process of successive elimination of clusters, merging clusters to one.

The expected number of phases required to reduce the number of clusters from n_c to one is given by the following proposition.

Proposition4. Let $\mathbf{G_c}=(\mathbf{V_c}, \mathbf{E_c})$ a cluster graph such that $|\mathbf{V_c}|=n_c$ and $|\mathbf{E_c}|=m_c$, and let $X(n_c)$ denotes the expected number of phases required to obtain a spanning tree. We have:

$$X(nc) = 1 + \sum_{k=kmin}^{kmax} Pr(M(G) = k)X(nc - k)$$

Where:

• k_{min} is the minimum cardinality of maximal matching,

• k_{max} is the maximum cardinality of maximal matching.

Remarque1. Note that by definition X (1) = 0.

Proposition5. The lower bound of expected number X (n_c) of phases required to construct a minimum spanning tree is $\log_2(n_c)$.

Proposition 6. The upper bound of the expected number $X(n_c)$ of phases required to construct a minimum spanning tree is n_c-1.

B) Minimality proof

It is easy to see that the spanning tree produced is minimal. Our algorithm can be considered as distributed Borůvka's algorithm. Indeed, the {\it Handshake Procedure} selects the edges, as a spanning sub-tree, which have the minimum values in the whole graph. The ends of each edge are merged in one end. And again the the *Handshake Procedure* is executed on the residual graph selection edges with minimum values. This process is reiterated until the residual graph is reduced to one component. The edges on with the handshakes are occurred are in the spanning tree. The latest is minimal since the selected edges at each iteration are minimal.

VI. CONCLUSION

Our algorithm proceed by round. In the first round each vertex constitute it-self a group. The algorithm of handshake is applied between the groups and for a handshake between two groups; those groups are merged in one group. In the nest round, the handshake algorithm is applied again. So the process continues until all groups are merged to only one. In this stage, the edges where the handshakes are occurred between groups form a minimal spanning tree reached.

REFERENCES

1. *H. Bersini. And the winner is...not the fittest [coevolution]. In Proceedings of the Evolutionary Computation on 2002. CEC '02. Proceedings of the 2002 Congress - Volume 02, CEC '02, pages 1510–1515, Washington, DC, USA, 2002. IEEE Computer Society.*
2. *H. Bersini. Des réseaux et des sciences: Biologie, informatique, sociologi : l'omniprésence des réseaux. Culture scientifique. Vuibert, 2005.*
3. *Yoann PIGNÉ. Modélisation et traitement décentralisé des graphes dynamiques. PhD thesis, UNIVERSITÉ DU HAVRE, 2008.*
4. *Mohamed Mejdi HIZEM. Recherche de chemin dans un graphe à pondération dynamique : Application à l'optimisation d'itinéraire dans les réseaux routiers. PhD thesis, ECOLE CENTRALE DE LILLE, 2008.*
5. *R.L. GRAHAM and P. HELL. On the history of the minimum spanning tree problem. Ann. Hist. Comput., page 43–57, 1985.*
6. *Otakar Boruvka. « o jistém problému minimálním (about a certain minimal problem) ». Práce mor. P˘rírodov˘ed. spol. v Brn˘e III, vol. 3:p. 37–58, 1926.*
7. *M. L. FREDMAN and R. E. TARJAN. Fibonacci heaps and their uses in improved network optimization algorithms. J. ACM 34, page 596–615, 1987.*
8. *Shehla ABBAS. Distributed Calculations using Mobile Agents, PHD thesis. École doctorale de mathématique et informatique, Université de Bordeaux I, 2008.*

9. *L. Lovasz and M.D. Plummer. Matching Theory. AMS Chelsea Publishing, North-Holland, Amsterdam New York, 1986.*
10. *Yves Métivier, Nasser Saheb, and Akka Zemmari. Analysis of a randomized rendezvous algorithm. Inf. Comput, 184(1):109–128, 2003.*
11. *A. El Hibaoui, Y. Métivier, and N. Zemmaril A. Robson, J.M.and Saheb. Analysis of a randomized dynamic timetable handshake algorithm. 2009.*