HAL open science

A Distributed Algorithm for the Minimum Diameter Spanning Tree Problem

Franck Butelle, Christian Lavault

To cite this version:

Franck Butelle, Christian Lavault. A Distributed Algorithm for the Minimum Diameter Spanning Tree Problem. 2nd International Conference On Principles Of Distributed Systems (OPODIS'98), Dec 1998, Amiens, France. pp.77-88. hal-00465670

HAL Id: hal-00465670

https://hal.science/hal-00465670v1

Submitted on 5 Apr 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A Distributed Algorithm for the Minimum Diameter Spanning Tree Problem

Marc Bui* Franck Butelle† Christian Lavault†

*Université de Technologie de Compiègne
Département Génie Informatique
Centre de Recherches de Royallieu
BP 649, F-60206 Compiègne
E-mail : Marc.Bui@utc.fr

† LIPN, CNRS URA1507, Institut *Galilée*
Université Paris-Nord
Avenue J.-B. Clément
F-93430 Villetaneuse.
E-mail : lavault@ura1507.univ-paris13.fr

Résumé

We present a new algorithm, which solves the problem of distributively finding a minimum diameter spanning tree of any arbitrary positively real-weighted graph. We use a new fast linear time intermediate all-pairs shortest paths routing protocol. The resulting distributed algorithm is asynchronous, it works for arbitrary named network, and achieves $O(n)$ time complexity and $O(nm)$ message complexity.

Keywords : Minimum diameter spanning trees, All-pairs shortest paths, Absolute centers, Shortest path trees.

1 Introduction

Many computer communication networks require nodes to broadcast information to other nodes for network control purposes, which is done efficiently by sending messages over a spanning tree of the network. Now optimizing the worst-case message propagation delays over a spanning tree is naturally achieved by reducing the diameter to a minimum ; especially in high-speed networks (where the message delay is essentially equal to the propagation delay).

The use of a control structure spanning the entire network is a fundamental issue in distributed systems and interconnection networks. Since *all* distributed total algorithms have a time complexity $\Omega(D)$, where D is the network diameter, having a spanning tree of minimum diameter makes it possible to design a wide variety of time efficient distributed algorithms.

1.0.1 The Multiport Model

A distributed system is a standard point-to-point asynchronous network consisting of n communicating processes connected by m bidirectional links. Each process has a local memory and can communicate by sending messages to and receiving messages from its neighbours. The model is for distributed message passing algorithms.

As usual, the network topology is described by a connected undirected graph $G = (V, E)$, devoid of multiple edges and loop-free. G is defined on a set V of vertices representing the processes and $E \subseteq V^2$ is a set of edges representing the bidirectional communication links operating between neighbouring vertices.

1.0.2 Basic Notation

In the sequel, $|V| = n$, and $|E| = m$. We view communication interconnection networks as undirected graphs. Henceforth, we use the terms *graph* (resp. *nodes/edges*) and *network* (resp. *processes/links*) interchangeably. In the remainder of the paper, we denote problems as "the (MDST) problem", "the (MST) problem", "the (GMDST) problem", etc. (See definitions below.) Distributed algorithms are denoted in italics, e.g. "algorithm *MDST*", "protocol *APSP*", etc. Finally, "MDST", "APSPs", and "SPT" abbreviate "Minimum Diameter Spanning Tree", "All-Pairs Shortest Paths", and "Shortest Path Tree", respectively.

1.0.3 The Problem

Let G be a positively real-weighted graph. The **(MDST) problem** is to find a spanning tree of G of minimum diameter.

The only assumptions needed in the MDST finding algorithm presented herein are twofold. (*i*) All the network's processes are assumed to have *distinct* identities (IDs). (In order to simplify the presentation, we also assume the set of all identities to be $\{1, \ldots, n\}$). (*ii*) The network G is regarded as a positively real-weighted graph (i.e., in G the edges' weight $\in \mathbb{R}_+$). Note that when edge weights are (possibly negative) real numbers ($\in \mathbb{R}$), The (MDST) problem is NP-complete. Besides, distinct IDs are needed to compute the APSPs' routing tables of G at each process in the network.

1.0.4 Related Works and Results

The few literature related to the (MDST) problem mostly deals either with graph problems in the Euclidian plane (Geometric Minimum Diameter Spanning Tree), or with the Steiner spanning tree construction (see [13, 14]). The (MDST) problem is clearly a generalization of the (GMDST) problem.

Surprisingly, despite the importance of having an MDST in arbitrary distributed systems, only few papers have addressed the question of how to design algorithms which construct such spanning trees. Finding and maintaining a *minimum* spanning tree (the (MST) problem) has been extensively studied in the literature (e.g., [1, 11] and [2, 6, 7]). Yet, there exist almost no algorithms which construct an MDST, in spite of the great importance of this issue in distributed applications. (Recently, the problem of maintaining a *small* diameter was however addressed in [15], and the distributed (MDST) problem in [3, 16]).

1.0.5 Main contributions of the paper

To our knowledge, our algorithm appears to be the first which *distributively* solves the (MDST) problem. The algorithm *MDST* is asynchronous and it works for arbitrary named network topologies. It achieves an efficient $O(n)$ time complexity, and $O(nm \log n + nm \log W)$ bits communication complexity, where W is the largest weight of the network's links.

The paper is organized as follows. In Section 2 we present a high-level description and a formal design of the algorithm *MDST*. Section 3 is devoted to the complexity analysis of the algorithm. The paper ends with concluding remarks in Section 4.

2 The Algorithm

2.0.6 Graph Theoretical Terminology

Let $G = (V(G), E(G))$ be a connected, undirected, positively real-weighted graph, where the weight of an edge $e = uv \in E(G)$ is given by $\omega_{uv} \in \mathbb{R}_+$. In the remainder of the paper, we use the graph theoretical terminology and notation. The weight of a path $[u_0, \ldots, u_k]$ of G ($u_i \in V(G)$) is defined as $\sum_{i=0}^{k-1} \omega_{u_i u_{i+1}}$. For all nodes u and v, the *distance* from u to v, denoted $d_G(u, v)$, is the lowest weight of any path length from u to v in G (∞ if no such path exists). The distance $d_G(u, v)$ represents the *shortest path* from u to v, and the largest (maximal) distance from node v to all other nodes in $V(G)$, denoted $s_G(v)$, is the *separation* of node v : viz. $s_G(v) = \max_{u \in V(G)} d_G(u, v)$ [5]. An *absolute center* of G is defined as a node achieving the smallest separation. $D(G)$ denotes the *diameter* of G, defined as $D(G) = \max_{v \in V(G)} s_G(v)$, and $R(G)$ the *radius* of G, defined as $R(G) = \min_{v \in V} s_G(v)$. Finally, $\Psi_G(u)$ represents a *shortest-paths tree* (SPT) rooted at node u : $(\forall v \in V(G)) \quad d_{\Psi_G(u)}(u, v) = d_G(u, v)$. Note that $\Psi_G(u)$ is one among all the shortest-paths trees rooted at node u. The set of all SPTs of G is then denoted $\Psi(G)$. The name of the graph will be omitted when it is clear from the context.

2.1 A High-Level Description

2.1.1 Main Issues

First, we recall (in Lemma 2.1) that the (MDST) problem for a graph G is (polynomially) reducible to the absolute center problem for G. Then, we constructively prove how to compute an absolute center of a weighted graph with the help of its APSPs' routing tables (see Lemma 2.2).

In summary, given a positively weighted graph G, the main issues of our algorithm for the (MDST) problem are threefold :

1. The computation of all APSPs in G ;
2. The computation of an absolute center of G ;
3. The construction of an MDST of G, and the knowledge's transmission of that MDST to each process within the network G.

The resulting algorithm *MDST* is thus made up of the corresponding basic protocol *APSP* (1) and procedure *Gamma_star* (2) designed in Subsection 2.2.

2.1.2 Construction of an MDST

The definition of separation must be generalized to *"dummy nodes"* (so-called in contrast to actual vertices of V). Such a fictitious node may possibly be inserted on any edge $e \in E$. Thus, let $e = uv$ be an edge of weight ω_{uv}, a dummy node γ inserted on e is defined by specifying the weight α of the segment $u\gamma$. According to the definition, the separation $s(\gamma)$ of a *general node* γ, whether it is an actual vertex in V or a dummy node, is clearly given by : $s(\gamma) = \max_{z \in V} d(\gamma, z)$. A node γ^* such that $s(\gamma^*) = \min_\gamma s(\gamma)$ is called an *absolute center* of the graph. Recall that γ^* always exists in a connected graph, and that is not unique in general.

Similarly, the definition of $\Psi(u)$ is also generalized so as to take these dummy nodes into account. Finding an MDST actually amounts to search for an absolute center γ^* of G, and the SPT rooted at γ^* is then an MDST of G. Such is the purpose of the following Lemma.

Lemma 2.1 *[4] The (MDST) problem for a given graph G is (polynomially) reducible to the problem of finding an absolute center of G.*

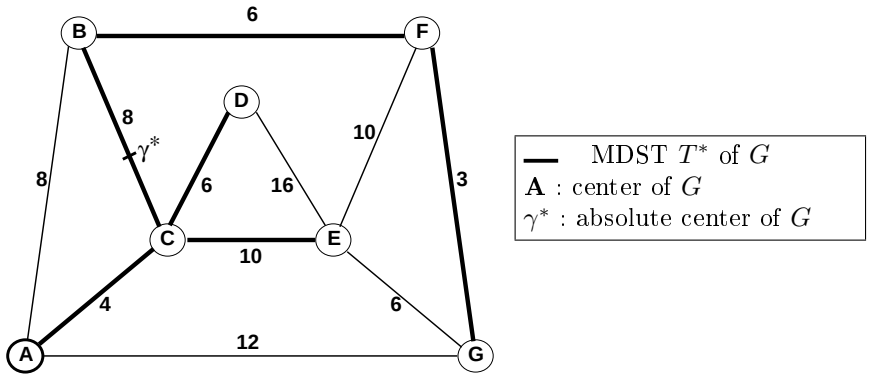

FIG. 1 – Example of an MDST T^* ($D(G) = 22$ and $D(T^*) = 27$). T^* is neither a shortest paths tree, nor a minimum spanning tree of G.

2.1.3 Computation of an absolute center of a graph

According to the results in [5], we use the following Lemma to find an absolute center of G.

Lemma 2.2 *Let $G = (V, E)$ be a weighted graph. An absolute center γ^* of G is constructed as follows :*

(i) On each edge $e \in E$, find a general node γ_e of minimum separation.
(ii) Among all the above γ_es, γ^* is a node achieving the smallest separation.

Proof : (the proof is constructive)

(i) For each edge $e = uv$, let $\alpha = d(u, \gamma)$. Since the distance $d(\gamma, z)$ is the length of either a path $[\gamma, u, \ldots, z]$, or a path $[\gamma, v, \ldots, z]$,

$$s(\gamma) = \max_{z \in V} d(\gamma, z) = \max_{z \in V} \min\{\alpha + d(u, z), \omega_{uv} - \alpha + d(v, z)\}. \tag{1}$$

If we plot $f_z^+(\alpha) = \alpha + d(u, z)$ and $f_z^-(\alpha) = -\alpha + \omega_{uv} + d(v, z)$ in Cartesian coordinates for fixed $z = z_0$, the real-valued functions $f_{z_0}^+(\alpha)$ and $f_{z_0}^-(\alpha)$ (separately depending on α in the range $[0, \omega_e]$) are represented by two line segments $(S_1)_{z_0}$ and $(S_{-1})_{z_0}$, with slope $+1$ and -1, respectively. For a given $z = z_0$, the smallest of the two terms $f_{z_0}^+(\alpha)$ and $f_{z_0}^-(\alpha)$ (in (1)) is thus found by taking the *convex cone* of $(S_1)_{z_0}$ and $(S_{-1})_{z_0}$. By repeating the above process for each node $z \in V$, all convex cones of segments $(S_1)_{z \in V}$ and $(S_{-1})_{z \in V}$ are clearly obtained (see Figure 2).

Now we can draw the *upper boundary* $B_e(\alpha)$ ($\alpha \in [0, \omega_e]$) of all the above convex cones of segments $(S_1)_{z \in V}$ and $(S_{-1})_{z \in V}$. $B_e(\alpha)$ is thus a curve made up of piecewise linear segments, which passes through several local minima (see Figure 2). The point γ achieving the smallest minimum value (i.e., the global minimum) of $B_e(\alpha)$ represents the absolute center γ_e^* of the edge e.

(ii) By definition of the γ_e^*s, $\min_\gamma s(\gamma) = \min_{\gamma_e^*} s(\gamma_e^*)$, and γ^* achieves the smallest separation. Therefore, an absolute center of the graph is found at any point where the minimum of all $s(\gamma_e^*)$s is attained. □

By Lemma 2.2, we may consider this method from an algorithmic viewpoint. For each $e = uv$, let $\mathcal{C}_e$ be the set of pairs $\{(d_1, d_2) \ / \ (\forall z \in V) \ d_1 = d(u, z), d_2 = d(v, z)\}$ Now, a pair (d_1', d_2')

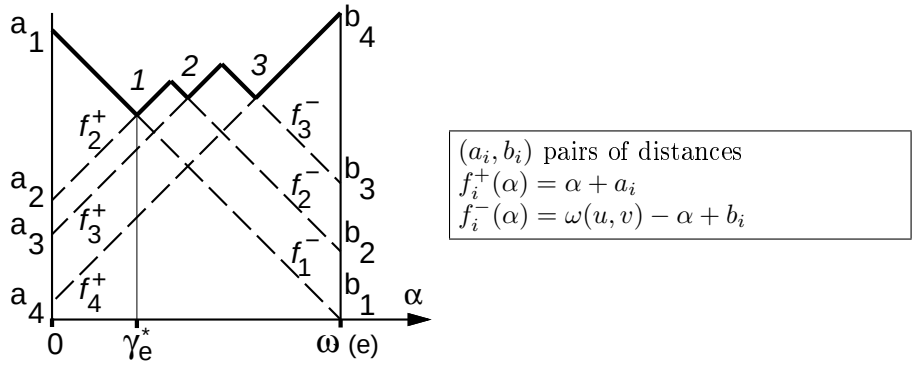

FIG. 2 – Example of an upper boundary $B_e(\alpha)$

is said to *dominate* a pair (d_1, d_2) iff $d_1 \leq d_1$', and $d_2 \leq d_2$' (viz. the convex cone of (d_1', d_2') is over the convex cone of (d_1, d_2)). Any such pair (d_1, d_2) will be ignored when it is dominated by another pair (d_1', d_2'). Notice that the local minima of the upper boundary $B_e(\alpha)$ (numbered from 1 to 3 in Figure 2) are located at the intersection of segments $f_i^-(\alpha)$ and $f_{i+1}^+(\alpha)$, when all dominated pairs are removed. If we sort the set $\mathcal{C}_e$ in descending order with respect to the first term of each remaining pair (d_1, d_2), we obtain the list $L_e = ((a_1, b_1), \ldots, (a_{|L_e|}, b_{|L_e|})$ consisting in all such remaining ordered pairs. Hence, the smallest minimum of $B_e(\alpha)$ for a given edge e clearly provides an absolute center γ_e^*. (See Procedure *Gamma_star*(e) in Subsection 2.2). By Lemma 2.2, once all the γ_e^*s are computed, an absolute center γ^* of the graph is obtained. By Lemma 2.1, finding an MDST of the graph reduces to the problem of computing γ^*.

2.1.4 All-Pairs Shortest-Paths Protocol (protocol *APSP*)

In the previous paragraph, we consider distances $d(u, z)$ and $d(v, z)$, for all $z \in V$ and each edge $e = uv$. The latter distances must be computed by a distributed routing algorithm, for which purpose the new protocol *APSP* is designed in Subsection 2.2. It is simple and time efficient. (Recall that the only assumption needed here is that all processes have distinct IDs).

Note that various compact routing methods could *a priori* be more (space and message) efficient and practical solutions to the problem, e.g., Interval routing, Prefix and Boolean routing, Multi-label routing, etc. Unfortunately, while we do need shortest paths routing tables to achieve a minimum diameter value, most graphs do *not* have *optimum* compact routing schemes in terms of shortest paths. More precisely, *arbitrary* graphs cannot have such optimum (APSPs) compact routing schemes, and whence, no exact MDST of such graphs can be completed by using compact routings. (See e.g. [8, 17], and Section 4).

2.1.5 Construction and knowledge transmission of an MDST

At the end of protocol *APSP*, every node knows the node r with the smallest ID and a shortest path in G leading to r. Now consider the set of paths $[u, r]$ from each node $u \in V$ to r. This set forms a tree rooted at r, and since it is an SPT of G, the information is exchanged *optimally* in the SPT $\Psi(r)$. Hence, the diameter $D(G)$ and the radius $R(G)$ can be computed through $\Psi(r)$. The number of messages needed to compute $D(G)$ and $R(G)$ represents the number of messages needed for an extremum searching in the tree $\Psi(r)$, that is at most $O(n)$.

Whenever the computation of an absolute center γ^* of G is completed, the endpoint of γ^*'s edge with the smallest ID sends a message to r carrying γ^*'s ID. Upon receipt of the message, r forwards this information over $\Psi(r)$. Therefore each node of G knows the route to γ^*, and the MDST is built as a common knowledge of all processes in G.

2.2 The algorithm

The distributed algorithm *MDST* finds an MDST of an input graph $G = (V, E)$ by computing the diameter of the SPTs for all nodes. Initially, an edge weight ω_{uv} is only known by its two endpoints u and v.

Algorithm *MDST* (for process u)

Type elt : **record** $alpha_best$, $upbound$: real ; id_1, id_2 : integer **end** ;
Var Λ : set of elt ; φ, φ_u^* : elt ; D, R, α, $localmin$: real ;
d_u : array of weights ; *(* $d_u[v]$ estimates $d(u,v)$ *)*

1. **For all** $v \in V$ Compute $d_u[v]$, D and R ; *(* from protocol APSP *)*
2. $\varphi.upbound \leftarrow R$;
3. **While** $\varphi.upbound > D/2$ **do for** any edge uv s.t. $v > u$
   (a) $(\alpha, localmin) \leftarrow Gamma_star(uv)$;
   (b) **If** $localmin < \varphi.upbound$ **then** $\varphi \leftarrow (\alpha, localmin, u, v)$;
4. $\Lambda \leftarrow \{\varphi\}$;
5. **Receive** $\langle\varphi\rangle$ from all sons of u in $\Psi(r)$ (r is s.t. $r = \min\{v \in V\}$) ;
   $\Lambda \leftarrow \Lambda \cup \{\varphi\}$;
6. Minimum finding :
   (a) Compute φ_u^* s.t. $\varphi_u^*.upbound = \min_{\varphi \in \Lambda} \varphi.upbound$;
   (b) **Send** $\langle\varphi_u^*\rangle$ to father in $\Psi(r)$;
   (c) **If** $u = r$ **then** upon reception of $\langle\varphi\rangle$ from all sons of r, r forwards $\langle\varphi_u^*\rangle$ to all other nodes.

Now we describe the basic procedures used in this algorithm : first the protocol *APSP* and next the procedure $Gamma_star(uv)$.

Protocol *APSP* (for process u)

Var d_u : array of weights ; $rout_u$: array of integers ; */* routing table */*
For all $v \in (V - \{u\})$ **do** $rout_u[v] \leftarrow 0$; $d_u[v] \leftarrow +\infty$ **end** */* Init */*
$marked \leftarrow \{u\}$; **Send** $\langle u, 0\rangle$ to all neighbours
Repeat
  For all answers $\langle x, d\rangle$ (from neighbour v) **do**
    If $x \neq u$ **and** $d_u[x] > d + \omega_{uv}$ **then**
      $d_u[x] \leftarrow d + \omega_{uv}$; $rout_u[x] \leftarrow v$;
  If any change occurs in the tables **then**
    Let v' be s.t. $d_u[v'] = \min_{\{v \notin marked\}} d_u[v]$
    $marked \leftarrow marked \bigcup \{v'\}$;
    Send $\langle v', d_u[v']\rangle$ to all neighbours
until no change

Assume the list L_e (defined above in Paragraph 2.1.3(a)) to be already constructed — e.g. with a heap whenever the routing tables are computed. The following procedure computes the value of γ_e^* for any fixed edge e.

Procedure ***Gamma_star***(e)

var min, α : real **Init** $min \leftarrow +\infty$; $\alpha \leftarrow 0$;
For i=1 to $|L_e|$ **do**
  compute the intersection (x, y) *of segments* f_i^- *and* f_{i+1}^+ *:*
  $x = \frac{1}{2}(\omega_e - a_i + b_{i+1})$; $y = \frac{1}{2}(\omega_e + b_{i+1} + a_i)$
  if $y < min$ **then** $min \leftarrow y$; $\alpha \leftarrow x$;
Return(α,min)

Example : consider the graph G depicted in figure 1. The MDST T^* is neither a shortest-paths tree $\Psi(r)$, nor a minimum spanning tree of G.

2.3 Improvements

In practice, some improvements in the algorithm *MDST* can easily be carried out. For example, reducing the enumeration of dummy nodes may be performed by discarding several edges of G from the exploration process. To be able to discard an edge, we only need to compute an upper bound on the diameter of an MDST of G. More precisely, we first compute $D(\Psi(u))$ for all nodes $u \in V$, and then keep the minimum value $\min_{T \in \Psi(G)} D(T)$, where the minimum is taken over all SPTs of G.

Example : When applied to the example in figure 1, $D(\Psi(\mathrm{A})) = 28$ is an upper bound on the diameter (27) of an MDST of G. The upper bound (28) makes it possible to discard the edges EF, and AB, AC, BF, CD, DE, EG, FG from the exploration.

3 Analysis

The following Lemma 3.1 provides the complexity of protocol *APSP*, as well as its termination proof. Theorem 3.1 derives the time and the communication complexity of the algorithm *MDST* from Lemma 3.1.

Lemma 3.1 *The All-Pairs Shortest Path protocol APSP process terminates. It runs in* $O(n)$ *phases and uses* $O(nm)$ *messages to compute the routing tables at each node of* G*. Its message size is at most* $O(\log n + \log W)$*, where* W *is the largest weight of edges.*

Proof : Consider the set *marked* defined in the protocol *APSP*. The set *marked* keeps growing as long as the completion of the protocol is not performed. Since only one node is added to *marked* at each phase of protocol *APSP* and since the number of nodes in the network is finite, the protocol does process terminate.

Now, the above proof of termination of protocol *APSP* also provides the number of phases $O(n)$ involved. Since each node's identity is sent exactly once along each one of its adjacent edges, the number of messages exchanged at every node is $2m$, and the message complexity of the whole protocol *APSP* is $2nm$. □

Theorem 3.1 *The algorithm MDST solves the (MDST) problem for any distributed positively weighted network* G *in* $O(n)$ *time. Its communication complexity is* $O(nmK)$ *bits, and its space complexity is at most* $O(nK)$ *bit, where* $K = O(\log n + \log W)$ *and* W *is the largest weight of edges in* G*.*

4 Concluding Remarks and Open Problems

Given a positively weighted graph G, our asynchronous algorithm *MDST* constructs an MDST of G and distributively forwards the control structure all over the named network G. This new algorithm is time and message efficient : the complexities are $O(n)$ and $O(nm)$, respectively, which in some sense seems "almost" the best achievable (though not optimal) in a distributed setting.

By contrast, the space complexity is far from satisfactory. This is due to the general assumptions of dealing with universal (APSPs) routings on arbitrary network topologies (see the protocol *APSP*). Typically, the necessary information is stored in a routing table with n entries, one entry for every possible destination, and one table at every node. With the expansion of networks, keeping routing tables becomes untenable, as at each node, the table takes $\Omega(n)\lceil \lg \delta \rceil$ bits (measured in $\lg n$-size words) for a node of degree δ. For the entire network, this leads to a total of $\sum_{u \in V} n \lg(\delta_u) = O(n^2 \log(m/n))$ bits. It was recently proved that reasonable APSP routing schemes require at least $\Omega(n^2)$ bit [10].

A more *compact* way of representing the tables is needed. So, in contrast with the above universal method, one may use various compact routing methods. They are proved more efficient, e.g. interval routing uses $O(\delta \log n)$ bits per node (hence $O(m \log n)$ total bits), and constructs routing tables in $O(m)$ messages. Moreover, when used for fixed classes $\mathcal{G}$ of graphs having optimum (APSPs) compact routing schemes, the construction of an MDST of $G \in \mathcal{G}$ may be performed by using interval, linear interval, prefix or boolean routing [8, 9, 12, 17]. This, according to the classes $\mathcal{G}$ of graphs considered : i.e., rings, trees, dimensional grids and tori, hypercubes, complete (bipartite) graphs, etc. [17]. For such graphs, the routing tables can (possibly) be represented with *compact optimum* routing (APSPs) schemes. For example, with no more than $2 \log n$ bits of strings as labels for the nodes, Boolean routing designs predicates which are optimum for wide classes of graphs [9]. The characterization of these classes of graphs which enjoy above properties is therefore a fundamental open problem.

Along the same lines, keeping compact (possibly optimum) routings for *self-stabilizing algorithms*, and designing efficient *self-stabilizing compact routing algorithms* are also a fundamental open problems in distributed systems where transient failures are considered.

Acknowledgments

We would like to thank Beryl T. Atkins for his very careful reading of our manuscript, and his suggestions and improvements of the paper as far as linguistic correctness is concerned.

Références

[1] B. Awerbuch. Optimal distributed algorithms for minimum weight spanning tree, counting, leader election and related problems. In *Proc. ACM STOC*, pages 230–240, 1987.

[2] B. Awerbuch, I. Cidon, and S. Kutten. Communication-optimal maintenance of replicated information. In *Proc. IEEE FOCS*, pages 492–502, 1990.

[3] M. Bui and F. Butelle. Minimum diameter spanning tree. In *Proc. Int. Workshop on Principles of Parallel Computing (OPOPAC'93)*, pages 37–46. Hermès & Inria, 1993.

[4] P. M. Camerini, G. Galbiati, and F. Maffioli. Complexity of spanning tree problems : Part I. *European J. of Operational Research*, 5 :346–352, 1980.

[5] N. Christophides. *Graph Theory : An algorithmic approach.* Computer Science and Applied Mathematics. Academic Press, 1975.

[6] D. Eppstein, Z. Galil, G. F. Italiano, and A. Nissenzweig. Sparsification – a technique for speeding-up dynamic graph algorithms. In *Proc. 33rd IEEE FOCS*, pages 60–69, 1992.

[7] D. Eppstein, G. F. Italiano, R. Tamassia, R. E. Tarjan, J. Westbrook, and M. Yung. Maintenance of a minimum spanning forest in a dynamic plane graph. *J. of Algorithms*, 13 :33–54, 1992.

[8] M. Flammini. *Compact routing models : some complexity results and extensions.* PhD. Thesis, Dip. Informatica e Sistemistica, Un. of Roma "La Sapienza", 1995.

[9] M. Flammini, G. Gambosi, and S. Salomone. Boolean routing. In *Proc. 7th Int. Workshop on Distributed Algorithms (WDAG'93), LNCS 725*, pages 219–233, 1993.

[10] P. Fraigniaud and C. Gavoille. Memory requirement for universal routing schemes. Technical report, LIP 95-05, ENSL, 1995.

[11] R. G. Gallager, P. A. Humblet, and P. M. Spira. A distributed algorithm for minimum weight spanning trees. *ACM TOPLAS*, 5(1) :66–77, 1983.

[12] C. Gavoille and E. Guevremont. Worst-case bounds for shortest path interval routing. Technical report, LIP 95-02, ENSL, 1995.

[13] J.-M. Ho, D. T. Lee, C.-H. Chang, and C. K. Wong. Minimum diameter spanning trees and related problems. *SIAM J. on Computing*, 20(5) :987–997, 1991.

[14] E. Ihler, G. Reich, and P. Widmayer. On shortest networks for classes of points in the plane. In *Proc. Int. Workshop on Computational Geometry – Methods, Algorithms and Applications*, Lecture Notes in Computer Science, pages 103–111, 1991.

[15] G. F. Italiano and R. Ramaswani. Maintaining spanning trees of small diameter. In *Proc. ICALP'94*, pages 227–238, 1994.

[16] C. Lavault. *Évaluation des algorithmes distribués — analyse, complexité, méthode.* Hermès, 1995.

[17] R. Tan and J. van Leeuwen. Compact routing methods : a survey. In *Proc. 1st Coll. on Structural Information and Communication Complexity (SIROCCO'94), Carleton Un. Press*, pages 99–109, 1995.