Construction repartie d'arbre couvrant de diametre minimum

M. Bui, Franck Butelle

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Construction répartie d'arbre couvrant de diamètre minimum

Marc BUI
Franck BUTELLE

N° 2017
Septembre 1993

PROGRAMME 1

Architectures parallèles,
bases de données,
réseaux et systèmes distribués

Construction Répartie d'Arbre Couvrant de Diamètre Minimum

Distributed Minimum Diameter Spanning Tree Construction

Marc BUI* Franck BUTELLE†

Août 1993

*Université de Paris X
200 av. de la République
92000 Nanterre.
Tél. : 40.97.70.79

†INRIA - Action ParaDis
e-mail : Franck.Butelle@inria.fr
Tél. : 39.63.55.48

RESUME

Dans cet article, nous proposons une solution au problème de la construction d'un arbre couvrant de diamètre minimum sur un graphe valué non-orienté. Ce problème est posé dans un contexte nouveau, celui de l'algorithmique répartie et notre solution, présentée sous forme d'algorithme distribué, à une complexité de $O\left(mn^2 \log_2(n+W)\right)$ en nombre de bits échangés (où n est le nombre de nœuds, m le nombre d'arêtes et W le poids maximum d'une arête). Notre approche est validée par des résultats expérimentaux qui confirment que notre algorithme, sur différents réseaux, réalise la construction, ou la reconstruction après panne d'une structure de contrôle efficace.

MOTS-CLES : Algorithme distribué, arbre couvrant, diamètre, graphes valués.

ABSTRACT

In this work, we study the problem of finding a minimum diameter spanning tree construction over an undirected weighted graph, in a distributed way. We present an original asynchronous distributed algorithm which solves this problem with bit-complexity of $O\left(mn^2 \log_2(n+W)\right)$ (n is the number of nodes, m the number of edges and W is the maximal edge weight). Experimental results confirm that our method is an useful one, it allows, over different computer structures, the construction of an efficient recovery control structure after faults.

KEY-WORDS : Distributed algorithm, spanning tree, diameter, weighted graphs.

1 Introduction

Le problème de l'arbre couvrant avec ou sans contraintes est un problème largement étudié et de nombreux algorithmes efficaces, tant séquentiels que distribués, ont étés présentés (voir par exemple [Awe87, BL91, GH85]).

Une variante originale de ce problème consiste à rechercher un Arbre Couvrant de Diamètre Minimum (ACDM). La littérature sur le sujet nous renvoie essentiellement à des problèmes de graphes dans le plan Euclidien ou au problème de la construction d'arbres de Steiner (voir [HLCW91, IRW91]). Mais ce dernier problème n'est qu'un cas particulier du problème de l'ACDM.

Dans le contexte de l'algorithmique distribuée, on considère un système informatique réparti sur plusieurs sites où il existe un réseau (graphe) de processus qui intéragissent seulement en se transmettant des messages.

Un des problèmes majeurs est, dans ce contexte, l'élaboration d'une structure de contrôle efficace couvrant la totalité du réseau. Ce problème peut être résolu en construisant un ACDM sur le graphe valué représentant le réseau de communication. Obtenir un arbre de diamètre minimal permet des gains substantiels sur les délais de transmission des messages de la plupart des algorithmes distribués car leur complexité temporelle est fonction, au moins linéaire, du diamètre.

Après avoir précisé le problème qui, exprimé en ces termes, est nouveau, nous exposons la méthode de résolution en deux étapes. Le modèle de calcul distribué et le détail de notre algorithme sont présentés dans le chapitre 4. Nous calculons ensuite la complexité en nombre de messages et discutons de développements futurs.

2 Le problème

2.1 notations

Nous présentons dans ce chapitre les notations usuelles (voir [Ber70]). Un 1-graphe simple est un graphe dans lequel il y a au plus une arête entre deux sommets distincts et il n'y a pas de boucle (arête (x, x)). Tous les graphes considérés dans cet article sont des 1-graphes simples.

Soit $G = (X(G), U(G), w)$ un graphe non-orienté, valué et connexe ; $|X(G)| = n$; $|U(G)| = m$. $X(G)$ est l'ensemble des nœuds et $U(G)$ est l'ensemble des arêtes. w est une fonction de coût (ou de poids, ou encore de délai de communication entre sites pour un réseau) de $U(G)$ vers $\mathbb{N}^*$ (ensemble des entiers naturels non-nuls).

Nous noterons $d_G(u, v)$ la distance (la somme des coûts des arêtes) d'un plus court chemin de u à v et $e_G(v)$, l'excentricité d'un nœud v, définie comme suit :

$$e_G(v) = \max_{k \in X(G)} d_G(v, k)$$

$\delta(G)$ représente le diamètre du graphe G. Il peut être défini par :

$$\delta(G) = \max_{v \in X(G)} e_G(v)$$

Soit $\rho(G)$ le rayon du graphe. Il peut être défini par :

$$\rho(G) = \min_{v \in X(G)} e_G(v)$$

Un centre v_0 du graphe est un sommet d'excentricité minimum, son excentricité est donc égale au rayon du graphe. Nous utiliserons aussi $APCC(x)$ pour l'Arbre des Plus Courts Chemins de x vers les autres nœuds du graphe. Cet arbre est donc tel que :

$$\forall x \in X(G), \forall i \in X(G),\ d_{APCC(x)}(x,i) = d_G(x,i)$$

Soit $E(G)$ l'ensemble des arbres de plus court chemin (APCC) sur G et enfin, soit δ^* le diamètre d'un ACDM sur G.

Dans la suite de cet article nous omettrons le G en indice tant que cela n'entraîne pas d'ambiguïté.

2.2 Expression du problème

Avec les notations précédentes, le problème se décrit comme suit :

Parmi l'ensemble des arbres couvrants de $G = (X(G), U(G), w)$, graphe non-orienté, valué positivement et connexe ; trouver un arbre T tel que son diamètre ($\delta(T)$) soit minimum.

Ce problème, exprimé ainsi, est nouveau, nous avons trouvé dans la littérature un problème similaire en apparence mais finalement très éloigné dans sa résolution. Ce dernier problème est appelé problème de l'arbre couvrant géométrique de diamètre minimum (geometric Minimum Diameter Spanning Tree) qui consiste à construire un arbre couvrant de diamètre minimum sur un ensemble de points du plan Euclidien. Ce problème a été résolu par [HLCW91, IRW91] avec des algorithmes séquentiels en $O(n^3)$. Le problème de l'ACDM est une généralisation de ce dernier, car on peut construire un graphe (complet) avec l'ensemble des points reliés deux à deux par des arêtes de poids égal à la distance Euclidienne.

3 Méthode

Nous présentons d'abord une méthode de construction d'un ACDM dans le cas d'un graphe valué uniquement avec des poids de 1, puis nous considérerons le cas plus général de poids entiers positifs.

3.1 Un cas particulier

Si toutes les arêtes sont de poids 1, alors pour construire un ACDM il suffit de choisir parmi les Arbre de Plus Courts Chemins (APCC) celui de diamètre minimum. Ce qu'exprime le lemme suivant :

Lemme 1 *Si* $w(e) = 1$ *pour tout* $e \in U(G)$, $\min_{T \in E(G)} \delta(T) = \delta^*$.

Preuve : Considérons un ACDM T^*, et v_0 un centre de T^* ; Soit T un arbre des plus courts chemins issus de v_0 ; nous avons alors la propriété suivante :

$$\forall i \in X, d_T(i, v_0) = d_G(i, v_0) \leq d_{T^*}(i, v)$$

Donc le diamètre de T est inférieur ou égal au diamètre de T^* ce qui prouve le lemme précédent. □

Donc, pour construire un ACDM, il suffit de choisir parmi les APCC celui dont le diamètre est minimum.

Remarque : Dans ces conditions, un nœud donnant un APCC de diamètre minimum est un des centres du graphe (mais attention tous les centres du graphe ne donnent pas forcément des ACDM, car les arbres des plus courts chemins issus d'un nœud n'ont pas tous le même diamètre (à 1 près)).

Cette remarque nous donne l'algorithme simplifié suivant qui construit un ACDM sur un graphe dont les poids des arêtes sont tous égaux à 1.

Algorithme **ACDM1**

1. CALCULER L'ENSEMBLE $C(G)$ DES CENTRES DU GRAPHE,
2. POUR CHAQUE CENTRE v_0 CALCULER LE DIAMÈTRE DE L'$APCC(v_0)$
3. LE CENTRE DONNANT LE MEILLEUR DIAMÈTRE DONNE L'ACDM.

Remarque : L'ACDM dans le cas où les poids sont tous égaux à 1, a comme diamètre 2ρ ou $2\rho - 1$. Donc, dans la recherche du meilleur $APCC(v_0)$ on peut s'arrêter si la borne inférieure $(2\rho - 1)$ est atteinte.

Une implémentation possible de cet algorithme passe par le calcul préalable des plus courts chemins. Tarjan a proposé [Tar83] une implémentation qui permet d'obtenir dans le cas présent une complexité temporelle en $O(m+n)$, ensuite il faut, pour chaque nœud, estimer le diamètre de l'APCC issu de ce nœud. Cette dernière étape est aisément réalisable en $O(n^2)$.

3.2 Cas général

Nous ne considérerons que le cas où les poids des arêtes sont des entiers naturels (non nuls), ceci est assez proche de la réalité des réseaux de communication car, si l'on considère les poids des arêtes comme étant des délais moyens de communication point à point, la précision de la mesure n'est jamais infinie et donc on peut toujours se ramener au cas ou les poids sont des entiers naturels.

La difficulté du problème de construction d'un ACDM vient de la remarque suivante :

Remarque : Lorsque les poids des arêtes sont des entiers naturels quelconques, il existe des graphes G tels que le diamètre d'un ACDM est plus petit que le plus petit des diamètres des arbres des plus courts chemins ($\min_{T \in E(G)} \delta(T) > \delta^*$).

Cette remarque est encore vraie si $E(G)$ est l'ensemble de arbres de poids minimum. Il nous suffit de considérer l'exemple de la figure 1 : sur cet exemple, l'ACDM n'est ni un arbre des plus courts chemins ni un arbre de poids minimum.

Pour construire un ACDM, une idée simple est de transformer le graphe G en un graphe G' valué uniquement avec des poids de 1. Construisons donc le graphe $G' = (X(G'), U(G'))$, à partir de G, de la façon suivante : transformons les arêtes

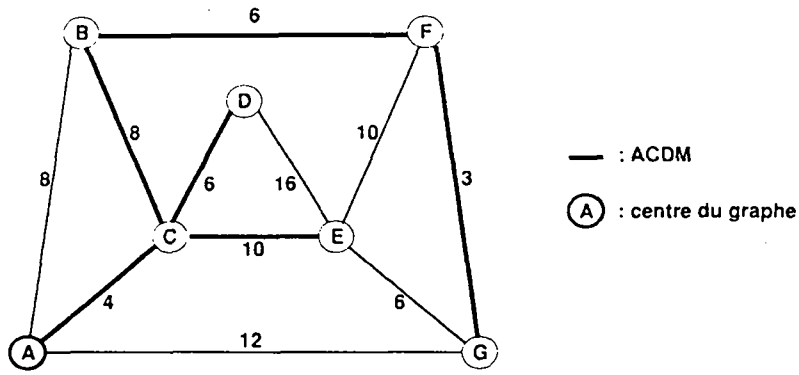

Figure 1 : Exemple d'Arbre Couvrant de Diamètre Minimum; $\delta(G) = 22$, $\delta^* = 27$.

$e = (i, j)$ de poids p supérieur à 1 en p arêtes de poids 1 et créons $p - 1$ nœuds de sorte que le chemin de i à j dans G' ait une longueur p.

Nous obtenons alors un graphe sur lequel on peut appliquer l'algorithme **ACDM1** précédant. Nous allons chercher à améliorer cette méthode en limitant le nombre de nœuds générés (que nous appellerons nœuds fictifs par opposition aux nœuds réels que sont les nœuds de G).

Remarquons d'abord qu'il n'est pas nécessaire de calculer les diamètres de tous les APCC de tous les nœuds de G'. En effet, il nous suffit de calculer les diamètres des APCC des nœuds de G' vers les nœuds de G. De plus, le diamètre d'un APCC d'un nœud fictif d'une arête (i, j) est fonction uniquement des plus courts chemins dans G à partir de i et de j. C'est pourquoi notre algorithme commencera par le calcul de toutes les distances $d(i, j)$.

Pour limiter l'exploration des nœuds fictifs, nous pouvons éliminer les arêtes les moins intéressantes. A cette fin, nous devons borner le diamètre δ^* d'un ACDM sur G. La borne inférieure est évidente : le diamètre de l'ACDM est a fortiori au moins égal au diamètre du graphe.

Nous allons considérer comme borne supérieure le plus petit des diamètres des arbres de plus court chemin ($\min_{T \in E(G)} \delta(T)$).

Pour calculer cette borne, il nous faut calculer $\delta(APCC(x))$ pour tous les nœuds x de G. L'algorithme suivant calcule cette valeur (ou une borne supérieure) pour x donné en utilisant les $d(x, k)$ et un tableau $par(x, k)$ représentant la table de routage de x. La table de routage $par(x, k)$ donne le voisin de x par lequel on obtient un des plus courts chemins pour aller de x vers k.

Algorithme **DIAM-APCC(x)**

1. SOIT K_1 ET K_2 LES NŒUDS LES PLUS ÉLOIGNÉS DE x (TELS QUE $d(x,k_1) \geq d(x,k_2)$).
2. SI $par(x,k_1) \neq par(x,k_2)$ ALORS $\delta(APCC(x)) = d(x,k_1) + d(x,k_2)$
3. SINON SOIT $y \in X(G)$ TEL QUE $par(x,k_1) \neq par(x,y)$ ET TEL QUE $d(x,y)$ SOIT MAXIMUM.
   SOIT $t \in X(G)$ TEL QUE $t = par(x,k_1) = par(x,k_2)$.
   (A) SI $d(x,k_1) + d(x,k_2) - 2d(x,t) > d(x,k_1) + d(x,y)$ ALORS $\delta(APCC(x)) \leq d(x,k_1) + d(x,k_2) - 2d(x,t)$ ET $\delta(APCC(t)) \leq \delta(APCC(x))$ STOP[a].
   (B) SINON $d(x,k_1) + d(x,k_2) - 2d(x,t) \leq d(x,k_1) + d(x,y)$ ET $\delta(APCC(x)) = d(x,k_1) + d(x,y)$.

[a] Dans ce cas, il n'est pas utile de calculer la valeur exacte de $\delta(SPT(x))$ car cet algorithme est appliqué à tous les nœuds donc en particulier à t qui est déjà plus intéressant que x.

Remarque : L'application de cet algorithme sur le graphe de la figure 1 donne un diamètre de 28 pour un APCC à partir du nœud A et cette valeur donne la borne supérieure sur le diamètre de l'ACDM. D'autres exemples nous ont prouvé que les centres du graphe ne sont pas forcément les nœuds donnant les meilleurs diamètres.

L'algorithme suivant construit un ACDM sur un graphe valué positivement.

Algorithme **ACDMp**

1. CALCULER LES DISTANCES $d(i,j)$ POUR TOUT COUPLE i,j DE NŒUDS DE G.
2. CALCULER LA BORNE SUPÉRIEURE : $bornesup \leftarrow \min_{x \in X} \delta(APCC(x))$ *(Utiliser DIAM-APCC(x), pour tout $x \in X$).*
3. $S \leftarrow U$.
4. TANT QUE $bornesup > \delta(G)$ ET QU'IL EXISTE AU MOINS UNE ARÊTE $e \in S$ (TELLE QUE e EST NON-ÉLIMINÉE) :
   (A) CALCULER EN CHAQUE NŒUD FICTIF DE e LE DIAMÈTRE d DE L'APCC DE CE NŒUD VERS LES NŒUDS DE G.
   (B) SI $d < bornesup$ ALORS $bornesup \leftarrow d$.
   (C) $S \leftarrow S \cup \{e\}$
5. LE NŒUD x (RÉEL OU FICTIF) TEL QUE $\delta(APCC(x)) = bornesup$ DONNE UN ACDM.

Les arêtes (i,j) que l'on peut "éliminer" de l'énumération sont celles vérifiant les conditions suivantes :

- Si $d(i,j) < w(i,j)$. En effet, cela signifie qu'il existe un chemin de longueur inférieure pour aller de i à j qui n'utilise pas cette arête.

  Parmi celles qui n'ont pas été éliminées par cette méthode, calculons le rayon de l'arbre obtenu pour un nœud fictif v de (i,j) à la distance a de i :

$$\rho(T(a)) = max_{k \in X(G)} (min(a + d(i,k); d(j,k) + d(i,j) - a))$$

  On peut minorer cette expression, sachant que $d(i,j) > a \geq 1$:

$$\rho(T(a)) \geq 1 + max_{k \in X(G)} (min(d(i,k); d(j,k)))$$

  De plus, sur un arbre où les poids sont tous de 1, nous avons : $\left\lceil \frac{\delta}{2} \right\rceil = \rho$.

  Donc, l'arête (i,j) peut-être éliminée dans le cas suivant :

- si $2 \max_{k \in X(G)} (min(d(i,k); d(j,k))) + 1 >= bornesup$.

Sur notre exemple, cet algorithme n'explore que les arêtes (A,G) (pas d'amélioration), (B,C) qui donne l'ACDM de la figure 1 et (C,E) qui donne un autre ACDM (échange de l'arête (B,F) par l'arête (E,G)).

L'étape 1 de l'algorithme est en $O(mn \log_{(2+m/n)} n)$ (*all pairs shortest paths* voir par exemple [Tar83]) et l'étape 2 est en $O(n^2)$. Enfin, l'étape 4 consiste, dans le pire des cas, pour toute arête, à vérifier que l'on ne peut l'éliminer de l'énumération et ensuite de réaliser cette énumération en calculant les diamètres des $APCC(v)$ pour tout nœud fictif v. En tout, $O(mn(\log_{(2+m/n)} n + W))$ unités de temps.

4 L'algorithme distribué

4.1 Hypothèses

Nous supposerons assurées les hypothèses suivantes :

- Les communications sont asynchrones non bloquantes (Il est possible d'utiliser un réseau de communication synchrone à condition de pouvoir exécuter plusieurs processus par nœud).
- Chaque site dispose d'un tampon de réception des messages borné.
- Les liaisons sont bidirectionnelles – les messages peuvent se croiser sur une ligne.
- Il n'y a ni perte, ni duplication, ni modification de message (fonctionnement non byzantin).
- Les délais de transmission sont finis (mais non bornés).
- Si des messages arrivent simultanément, ils sont pris en compte séquentiellement.
- Il n'y a pas de panne de site ni de ligne de communication. Cette hypothèse peut être levée (ou tout du moins relâchée) dans certains cas que l'on précisera plus loin.

- Il n'y a pas de déséquencement des messages – les messages ne peuvent se doubler sur la ligne ; c'est la discipline FIFO "First In, First Out" (on peut toujours se ramener à ce cas par l'ajout d'estampilles, voir [Lam78]).
- Les seules connaissances accessibles pour un nœud sont son identité et les portes d'Entrée/Sortie (distinctes) qui mènent vers ses voisins (dans les algorithmes qui suivent, par souci de lisibilité, nous ne ferons plus la distinction entre identité de porte et identité de voisin).
- Toutes les identités des nœuds sont distinctes.
- Le graphe représentant le réseau des sites est connexe.
- Un sous-ensemble non vide de sites démarrent l'algorithme de façon spontanée. Les autres sites du réseau s'éveillent dès la réception d'un message.

4.2 Spécification

L'algorithme suivant construit, en environnement distribué, un ACDM sur le graphe sous-jacent au réseau, les nœuds étant associés aux sites et les arêtes aux liens de communication. Le poids d'une arête n'est initialement connue que par les deux sites extrémités de l'arête.

Algorithme **DISTRI-ACDMp**

- CALCULER LES $d(i,j)$ ET $par(i,j)$ [a].

POUR TOUT SITE i:

1. *bornesup* ←**DIAM-APCC**(i).
2. RECHERCHE DU MINIMUM DES BORNES SUPÉRIEURES :
   (A) SI *bornesup* $> 2\rho(G)$ ALORS LE SITE i EST PASSIF[b].
   (B) SI *bornesup* $> \delta(G)$ ALORS LE SITE i ENVOIE *bornesup* VERS LE LEADER.
   (C) SINON (borne supérieure = borne inférieure) ALLER EN 4.
3. POUR TOUTES LES ARÊTE (i,j) TELLES QUE $j > i$: SI L'ARÊTE (i,j) N'EST PAS ÉLIMINÉE[c] FAIRE :
   (A) CALCULER LE MEILLEUR $d = \delta(APCC(x))$ OÙ x EST UN NŒUD FICTIF DE L'ARÊTE (i,j) *(ce calcul nécessite la participation du site j).*
   (B) SI $d <$ *bornesup* ALORS *bornesup* $\leftarrow d$
4. ENVOI DE *bornesup* VERS LE LEADER.

- LE LEADER ATTEND DE RECEVOIR TOUS LES MESSAGES AVANT DE LANCER LA TERMINAISON DE L'ALGORITHME.

[a] Divers algorithmes de construction s'offrent à nous, que ce soit d'un point de vue all-pairs-shortest-paths ou tables de routage. Nous avons choisi d'utiliser [MS79] surtout parce qu'il fournit une terminaison (chaque site sait, au bout d'un temps fini, qu'il n'apprendra rien de plus de ses voisins et donc peut lancer la suite de l'algorithme) et aussi parce qu'il est adaptatif. Nous calculons, en même temps que les tables de routage, $\delta(G)$ et $\rho(G)$ (modifications diffusées en même temps que les chemins améliorants). A ce niveau de l'exécution de l'algorithme, l'ensemble des identités des sites du réseau est connu de chaque site. Par conséquent, uniquement dans le but de simplifier l'écriture de la suite de l'algorithme, nous pouvons numéroter les sites, du plus petit au plus grand, avec les nombres de 1 à n, 1 étant le leader. L'arborescence fournie par les $d(1,k)$ est utilisée comme arbre couvrant pour limiter le nombre de messages échangés, organiser la recherche de la borne supérieure et décider de la terminaison de l'algorithme.

[b] Un nœud passif ne fera que transmettre les messages lui arrivant vers le leader.

[c] Conditions définies au paragraphe 3.2.

5 Analyse de l'algorithme

Nous allons analyser l'algorithme **DISTRI-ACDMp** du point de vue de la quantité d'information échangée plutôt que du point de vue du nombre de messages. La quantité d'information échangée est la somme des nombres de bits des messages échangés. Les messages doivent (pour la plupart) comporter l'identité de l'émetteur ainsi qu'un poids. Nous noterons T la taille (en bits) d'un message ici égale à $O(\log_2 W + \log_2 n)$ où W représente le poids maximum d'une arête.

L'étape 1 de l'algorithme consiste en la construction des plus courts chemins. Si l'on connaît une borne N sur le nombre de nœuds du réseau, cette étape, avec

l'algorithme de Merlin & Segall [MS79], est exécutée avec $O(mn^2T)$ bits échangés. L'étape 2 ne comporte qu'un calcul local à chaque site. L'étape 3 est équivalente à la recherche de minimum sur un arbre de n nœuds et est donc en $O(n)$ messages. Dans l'étape 4, l'algorithme effectue l'analyse de l'arête (i,j), $j > i$, en chaque site i. Cette exploration nécessite les tables de routage de i et de j. Suite à la demande de i, j envoie sa table de routage ainsi que les $d(j,k)$ au site i. Dans le pire des cas, la quasi totalité des arêtes est explorée. Finalement l'étape 4 provoque l'échange de $O(m)$ messages de $O(nT)$ bits. L'étape 5 est équivalente à l'étape 4. Enfin, la diffusion du mesage de terminaison par le leader ne représente qu'une diffusion en $O(nT)$ bits pour que tous connaissent δ^* et le nœud (réel ou fictif) permettant sa génération.

En tout, l'algorithme **DISTRI-ACDMp** a la même complexité que l'algorithme de calcul des plus courts chemins : $O(mn^2T)$ bits échangés.

En cas de panne franche de ligne (ou modification de son poids ou encore de suppression de cette ligne), la reconstruction de l'ACDM passe par la reconstruction des tables de routages, or, l'algorithme de [MS79] est adaptatif donc résistant à ce type de panne sans entraîner une reconstruction complète. Dans le meilleur cas l'arête impliquée n'appartient pas à l'ACDM et est ignorée dans les conditions définies au paragraphe 3.2, dans les autres cas, il faudra recommencer la construction à partir de l'étape 1.

6 Conclusion

Nous avons présenté dans cet article une solution au problème de l'élaboration d'une structure de contrôle efficace (*i.e.* un arbre couvrant de diamètre minimum) sur un réseau de sites communiquant entre eux.

Les intérêts d'une telle construction sont multiples, elle offre, entre autres, une structure minimale pour un routage efficace prenant en compte d'éventuels changements de topologie du réseau (addition de liens ou sites) et peut être adoptée comme protocole de restructuration après panne.

L'algorithme que nous avons présenté fournit une solution originale à un problème re-formulé dans un cadre nouveau, il est basé sur des principes de calcul distribué utilisant des messages asynchrones, de plus cet algorithme est adaptatif et résistant aux pannes et a la même complexité que l'algorithme de calcul des plus courts chemins.

De nombreux problèmes attenants sont en cours d'étude notamment la résolution en distribué du problème de l'ACDM de poids minimum qui est un problème NP-complet (*c.f.* [GJ79]).

Bibliographie

[Awe87] Awerbuch (Baruch). – Optimal distributed algorithms for minimum weight spanning tree, counting, leader election and related problems. *In : ACM Symp. Theory Comp.*, pp. 230–240. – 1987.

[Ber70] Berge (Claude). – *Graphes et Hypergraphes.* – Paris, Dunod, 1970, *Monographies universitaires de mathématiques.* English translation : Graphs and Hypergraphs (North-Holland, Amsterdam 1973).

[BL91] Butelle (Franck) et Lavallée (Ivan). – *Un algorithme distribué d'élection non pré-déterminée et d'arbre couvrant.* – RR n° 1444, INRIA – France, 1991.

[GH85] Graham (R. L.) et Hell (P.). – On the history of the minimum spanning tree problem. *Annals of the History of Computing*, vol. 7, n° 1, 1985, pp. 43–57.

[GJ79] Garey (M. R.) et Johnson (D. S.). – *Computers and intractability: A guide to the theory of NP-completeness.* – W. H. Freeman, 1979.

[HLCW91] Ho (J.-M.), Lee (D. T.), Chang (C.-H.) et Wong (C. K.). – Minimum diameter spanning trees and related problems. *SIAM Journal on Computing*, vol. 20, n° 5, octobre 1991, pp. 987–997.

[IRW91] Ihler (E.), Reich (G.) et Wildmayer (P.). – On shortest networks for classes of points in the plane. *In : Int. Workshop on Computational Geometry - Meth., Alg. and Appli.*, pp. 103–111. – mars 1991.

[Lam78] Lamport (Leslie). – Time, clocks and the ordering of events in a distributed system. *Communications of the ACM*, vol. 21, n° 7, 1978, pp. 558–565.

[MS79] Merlin (P.) et Segall (A.). – A failsafe distributed routing protocol. *IEEE Transactions on Computers*, vol. COM-27, n° 9, septembre 1979, pp. 1280–1287.

[Tar83] Tarjan (R. E.). – *Data structures and network algorithms.* – SIAM, 1983.

Unité de Recherche INRIA Rocquencourt
Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 LE CHESNAY Cedex (France)

Unité de Recherche INRIA Lorraine Technopôle de Nancy-Brabois - Campus Scientifique
615, rue du Jardin Botanique - B.P. 101 - 54602 VILLERS LES NANCY Cedex (France)
Unité de Recherche INRIA Rennes IRISA, Campus Universitaire de Beaulieu 35042 RENNES Cedex (France)
Unité de Recherche INRIA Rhône-Alpes 46, avenue Félix Viallet - 38031 GRENOBLE Cedex (France)
Unité de Recherche INRIA Sophia Antipolis 2004, route des Lucioles - B.P. 93 - 06902 SOPHIA ANTIPOLIS Cedex (France)

EDITEUR
INRIA - Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 LE CHESNAY Cedex (France)

ISSN 0249 - 6399