Construction Locale de Sous-Graphes Couvrants Peu Denses[†]

Bilel Derbel[‡], Cyril Gavoille[§], David Peleg[¶], Laurent Viennot[‖]

Nous présentons un algorithme distribué déterministe qui calcule pour tout graphe simple non pondéré, un sous-graphe couvrant (*spanner*) avec $O(kn^{1+1/k})$ arêtes et un facteur d'étirement $(2k-1,0)$, n étant le nombre de sommets du graphe et k un paramètre entier strictement positif. Si n est inconnu l'algorithme termine en temps $3k-2$, sinon il termine en temps k. En se basant sur cet algorithme pour $k=2$, nous construisons de façon déterministe un sous-graphe couvrant avec $O(\varepsilon^{-2}n^{3/2})$ arêtes et un facteur d'étirement $(1+\varepsilon,2)$ en $O(\varepsilon^{-1})$ temps, $\varepsilon>0$ est un paramètre arbitrairement petit.

Nous complétons nos algorithmes par des bornes inférieures. D'abord, nous montrons que k unités de temps sont nécessaires pour calculer un sous-graphe couvrant avec $o(n^{1+1/(k-1)})$ arêtes et un facteur d'étirement $(2k-1,0)$ pour $k\in\{2,3,5\}$. Ensuite, nous montrons que pour tout $k>1$, si un algorithme distribué construit un sous-graphe couvrant H ayant moins de $n^{1+1/k+\varepsilon}$ arêtes en temps $t\leqslant n^{\varepsilon}$, alors H a un facteur d'étirement au moins $(1+\Omega(k/t),\Omega(kn^{\varepsilon}))$. Nos bornes sont valables aussi bien pour des algorithmes déterministes que probabilistes, avec ou sans la connaissance de n.

Keywords: Algorithmes distribués, sous-graphe couvrant peu dense, complexité en temps.

1 Introduction

Dans ce papier nous nous intéressons à la construction distribuée de structures efficaces de graphes, appelées *spanners*. Les spanners permettent de couvrir les sommets d'un graphe en utilisant peu d'arêtes tout en préservant les distances. Plus formellement, étant donné un graphe $G=(V(G),E(G))$ qui modélise un réseau d'interconnexion, un sous-graphe H de G tel que $V(H)=V(G)$ est un (α,β)-spanner si pour tout couple de sommets u et v, $d_H(u,v)\leqslant\alpha\cdot d_G(u,v)+\beta$, (avec $d_X(u,v)$ désignant la distance dans X entre u et v). Le *facteur d'étirement* du spanner H est définit par le couple (α,β) et sa taille par $|E(H)|$ (le nombre d'arêtes utilisées par le spanner). En pratique, les spanners sont utilisés de façon explicite ou implicite comme structure de base dans plusieurs applications distribuées tels que, les synchroniseurs [2], le routage compacte [1, 10], les forêts couvrantes [3], etc.

Étant donnée un graphe G avec n sommets et un paramètre entier positif $k\geqslant 1$, il est bien connu que G admet un $(2k-1,0)$-spanner de taille $O(n^{1+1/k})$. Ce compromis entre la taille et le facteur d'étirement est considéré comme étant optimal d'après une conjecture d'Erdös [8] prouvée pour $k\in\{1,2,3,5\}$.

Dans ce papier, nous considérons *la complexité en temps* de la construction de spanners optimaux avec des algorithmes *distribués*, c'est à dire dans un modèle de calcul où les sommets sont des entités autonomes de calculs pouvant communiquer en envoyant et en recevant des messages. Le meilleur algorithme distribué déterministe existant [6] permet de construire des $(4k-3,0)$-spanner avec $O(k\cdot n^{1+1/k})$ arêtes en temps $2^{O(k)}\log^{k-1}n$. Le meilleur algorithme probabiliste [5] permet de construire un $(2k-1,0)$-spanner

[†] Version complète disponible sur http://dept-info.labri.fr/~gavoille/podc08.ps

[‡] Laboratoire d'Informatique Fondamentale de Lille (LIFL), Université des Sciences et Technologies de Lille, France. bilel.derbel@lifl.fr. Supported by the equipe-projet INRIA "DOLPHIN".

[§] Laboratoire Bordelais de Recherche en Informatique (LaBRI), Université de Bordeaux, France. gavoille@labri.fr. Supported by the ANR-project "ALADDIN", and the équipe-projet INRIA "CÉPAGE".

[¶] Department of Computer Science and Applied Mathematics, The Weizmann Institute, Rehovot, Israel. david.peleg@weizmann.ac.il. Supported in part by grants from the Israel Science Foundation and the Minerva Foundation.

[‖] INRIA, University Paris 7, France. Laurent.Viennot@inria.fr. Supported by the ANR-project "ALADDIN", and the équipe-projet INRIA "GANG".

avec $O(k \cdot n^{1+1/k})$ arêtes en temps k. Cependant, ce dernier algorithme est de type Monte-Carlo, c'est à dire que les propriétés du spanner ne sont garanties qu'avec une certaine probabilité. Ceci peut être problématique dans le cadre d'applications qui exigent des bornes sûres concernant les propriétés d'un spanner. D'un point de vue plus fondamental, là où les algorithmes probabilistes permettent de calculer plusieurs structures de façon rapide, les algorithmes déterministes se voient ralentis considérablement notamment à cause des difficultés inhérentes aux problèmes de symétrie. Casser la symétrie par des algorithmes déterministes est l'un des défis majeure des calculs distribués. Notre premier résultat est un algorithme distribué extrêmement simple permettant de calculer de façon déterministe un $(2k-1,0)$-spanner de taille $O(k \cdot n^{1+1/k})$ en temps $O(k)$. Notre algorithme possède en plus une propriété particulièrement intéressante : il ne nécessite pas la connaissance de n. Ceci n'est pas le cas des algorithmes (probabilistes ou déterministes) existants qui eux doivent connaître au moins une bonne approximation de n. Ceci rend notre algorithme particulièrement adéquat dans le cadre d'applications pratiques par exemple, dans des applications où le graphe est dynamique c'est à dire où des sommets peuvent apparaître et/ou disparaître.

Par ailleurs, nous nous intéressons à un autre type de spanners dits *additifs*, c'est à dire des spanners où le facteur d'étirement est de la forme $(1,\beta)$. Seulement quelques spanners de ce type sont connus. Plus précisément, il existe un $(1,2)$-spanner avec $O(n^{3/2})$ arêtes [7] et un $(1,6)$-spanner avec $O(n^{4/3})$ arêtes [4]. Cependant des constructions qui essayent de donner une approximation de spanners additifs existent. Les algorithmes distribués les plus rapides [6] permettent de construire un spanner de taille $O(n^{3/2})$ avec un facteur d'étirement $(1+\varepsilon,4)$ en temps $n^{o(1)} + O(\varepsilon^{-1})$ (ou $O(\log n + \varepsilon^{-1})$ temps en moyenne) ou avec un facteur d'étirement $(1+\varepsilon, 8\log n)$ en temps $O(\log n \cdot \varepsilon^{-1})$. Notre deuxième résultat est un algorithme distribué déterministe qui construit un $(1+\varepsilon,2)$-spanner de taille $O(\varepsilon^{-2}n^{3/2})$ en temps $O(\varepsilon^{-1})$. Là aussi, la connaissance de n n'est pas requise.

Les algorithmes distribués que nous présentons sont complètement nouveaux dans le sens où ils ne sont pas une adaptation des techniques séquentielles ou distribués existantes (y compris celles probabilistes).

Finalement, nous complétons nos deux résultats par deux bornes inférieures. Notre première borne montre que tout algorithme distribué (potentiellement probabiliste) nécessite au moins k temps pour calculer un $(2k-1,0)$-spanner de taille $o(n^{1+1/(k-1)})$ pour $k \in \{2,3,5\}$. En se basant sur la conjecture evoquée plus haut, le résultat reste vrai pour tout $k > 1$. Notre deuxième borne inférieure montre que pour tout $k > 1$, il existe un $\varepsilon > 0$ tel que pour tout algorithme distribué construisant un spanner H pour G de taille inférieure à $n^{1+1/k+\varepsilon}$ en temps n^{ε}, il existe au moins deux sommets du graphe tel que $d_H(u,v) \geqslant d_G(u,v) + n^{\Omega(\varepsilon)}$. Ce résultat est en fait un corollaire d'un résultat plus technique qu'on donnera plus loin.

2 Les bornes Supérieures

Nous considérons des graphes simples non pondérés. Nous considérons le modèle de calcul classique de Linial [9]. Les sommets possèdent des identifiants uniques et ils effectuent des calculs de façon synchrone. Chaque unité de temps, un sommet peut envoyer ou recevoir des messages de taille non limitée, et effectuer n'importe quel calcul local. La complexité en temps est ainsi définit comme étant le nombre total d'unités de temps depuis le début de l'algorithme jusqu'à sa terminaison. On note cependant que puisque on ne s'intéresse qu'au temps de calcul et non au nombre de messages échangés, un algorithme synchrone peut être adapté à un modèle asynchrone tout en gardant la même complexité en temps.

Pour tout sommet u d'un graphe G, et pour tout entier positif r, nous définissons $B_G(u,r)$ le sous-graphe de G induit par les sommets à distance au plus r de u dans G. Nous omettrons de préciser G dans notre notation quand il n'y a pas d'ambiguïté.

2.1 *Algorithme* SPAN$_k$ *: facteur d'étirement* $(2k-1,0)$

L'algorithme SPAN$_k$ est décrit de façon formelle dans la figure 1. L'idée générale de l'algorithme est la suivante. Tout sommet u maintient une région $R(u)$ qui grossit à chaque itération. La valeur courante de la région de u est C et l'ensemble des voisins non encore couverts est W. À chaque itération certains voisins dans W migrent vers l'ensemble L, c'est à dire qu'il seront dans le voisinage direct de u dans le spanner. En effet, en analysant les régions $R(w)$ de ses voisins, le sommet u peut déterminer s'il existe un chemin dans le spanner (en cours de construction) qui mène vers un sommet de W. Si c'est le cas, ce sommet est enlevé de W. Sinon, il migre dans L. Bien entendu, l'algorithme fait attention à ne pas rajouter beaucoup de sommets dans l'ensemble L ce qui permet de garantir un spanner avec la taille requise.

$L := C := R(u) := \{u\}$, $W := B(u,1)$, $\sigma := |B(u,2k-1)|^{1/k}$
pour $i := 1$ *to* k **faire**
  Le sommet u envoie $R(u)$ à ses voisins, et il reçoit $R(w)$ de ses voisins w
  tant que $\exists w \in W$ *tel que* $R(u) \cap R(w) \neq \varnothing$ **faire**
    $W := W \setminus \{w\}$
  tant que $\exists w \in W$ *et* $|L| < i\sigma$ **faire**
    $W := W \setminus \{v \in W \mid R(v) \cap R(w) \neq \varnothing\}$
    $L := L \cup \{w\}$
    $C := C \cup R(w)$
  $R(u) := C$

Algorithme 1: Algorithme $\text{SPAN}_k(u)$ pour un sommet u. L'ensemble L calculé par u contient les voisins de u dans le spanner.

En montrant par récurrence qu'à la fin d'une itération donnée i, l'algorithme SPAN_k, l'une des deux propriétés suivantes est vraie : soit $|R(u)| \geqslant |B(u,2k-i)|^{i/k}$ ou $R(v) \cap R(u) \neq \varnothing$ pour tout $v \in B(u,1)$, on obtient :

Théorème 1 *L'algorithme* SPAN_k *est un algorithme distribué et déterministe qui calcule pour tout graphe à n sommets un* $(2k-1,0)$*-spanner avec* $kn^{1+1/k}$ *arêtes en temps k. Si n n'est pas connu, alors le temps de calcul est* $3k-2$*.*

2.2 Algorithme $\text{SPAN}_{2,t}$: facteur d'étirement $(1+\varepsilon,2)$

L'algorithme $\text{SPAN}_{2,t}$ ci-dessous est une extension de l'algorithme précédent dans le cas $k=2$.

$R(u) := \{u\}$, $r := \lfloor t/2 \rfloor$, $\sigma := |B(u,2r+1)|^{1/2}$
Le sommet u envoie $R(u)$ à ses voisins, et reçoit $R(w)$ de ses voisins w
Le sommet u choisit $\min\{\deg(u), \lceil\sigma\rceil - 1\}$ voisins et les rajoute à l'ensemble $R(u)$
$H_u := (R(u), \bigcup_{w \in R(u)\setminus\{u\}} \{u,w\})$
si $R(u) = B(u,1)$ **alors** $F_u :=$ vrai **sinon** $F_u :=$ faux
Le sommet u envoie $(R(u), F_u)$ aux sommets de $B(u,r)$ et reçoit $(R(w), F_w)$ des sommets $w \in B(u,r)$
pour $\ell := 1$ *à* r **faire**
  $W_\ell := \{w \in B(u,\ell) \mid F_w \text{ is false}\}$
  tant que $\exists w \in W_\ell$ **faire**
    $W_\ell := W_\ell \setminus \{v \mid R(v) \cap R(w) \neq \varnothing\}$
    Ajouter à H_u un plus court chemin dans G reliant u et w

Algorithme 2: Algorithme $\text{SPAN}_{2,t}(u)$ pour un sommet u et pour $t = 1 + \lceil 4/\varepsilon \rceil$ où $\varepsilon \in (0,4]$ est un paramètre. Le spanner global est l'union des sous-graphes H_u calculés par chaque sommet u.

L'idée générale de l'algorithme $\text{SPAN}_{2,t}$ exécuté par un sommet u est de rajouter des plus courts chemins entre u et certains sommets du voisinage à distance $O(t)$ de u. Ainsi, lorsque une longue distance est à approximer, on n'est pas obligé de payer un étirement 3 pour chaque arête. Au contraire, on essaye de faire des sauts en utilisant les plus courts chemins qu'on a calculé. En remarquant que les régions des sommets enlevés par la boucle sont disjointes et de taille supérieure à σ, on peut alors montrer le théorème suivant :

Théorème 2 *Pour tout* $\varepsilon \in (0,4]$*, il existe un algorithme distribué et déterministe tel que pour tout graphe à n sommets (n n'est pas connu des sommets) calcule un* $(1+\varepsilon,2)$*-spanner avec* $O(\varepsilon^{-2}n^{3/2})$ *arêtes en temps* $O(\varepsilon^{-1})$.

3 Les Bornes Inférieures

Les preuves de nos bornes inférieures sont techniques et trop longues pour être exposées en détail. Nous allons donc nous contenter de les énoncer et de donner la technique générale que nous avons utilisée.

Un algorithme distribué éventuellement probabiliste opère toujours en cycles se composant d'une étape de communication suivie d'un calcul local. La technique que nous utilisons se base sur l'observation suivante. À un temps τ , un sommet u ne peut connaître que les identifiants des sommets à distance τ et les $\tau - d_G(u,v) + 1$ premiers choix aléatoires effectués par un sommet v. On peut ainsi définir *la vue* d'un sommet u, c'est à dire toute l'information que u peut avoir à sa disposition au bout d'un certain temps τ. Ceci nous permet de définir l'état d'un sommet comme étant une fonction de sa vue. Ainsi, deux sommets ayant la même vue auront forcément le même état indépendamment du graphe auquel ils appartiennent.

La preuve de la première borne consiste à trouver deux familles de graphe pour lesquelles on peut trouver deux sommets ayant la même vue au bout d'un temps τ donné. Mais, où un algorithme correct doit attribuer deux états différents à ces deux sommets. Typiquement, pour le problème des spanners, un sommet doit décider de n'enlever aucune arête, alors que l'autre doit en enlever au moins une. Ceci permet donc de prouver qu'un temps τ ne suffit pas pour résoudre le problème correctement.

La preuve de la deuxième borne est techniquement différente. Elle consiste à construire un graphe pour lequel on peut dire précisément, en s'appuyant sur les vues de certains sommets, quelles sont les arêtes qui doivent être enlevées du graphe pour garantir la taille désirée. Ensuite, il s'agit de donner un chemin qui passe par ces arêtes et pour lequel la distance dans le spanner est forcément plus grande qu'une valeur qu'on sait calculer très précisément. Ceci nous permet d'obtenir une borne sur le facteur d'étirement.

Les deux preuves s'appuient sur l'existence de graphes biparties dont la longueur du plus petit cycle est $2k+2$ et ayant $c \cdot n^{1+\Psi(1/k)}$ arêtes, avec c une constante, $\Psi(k) = k$ si $k \in \{1,2,3,5\}$, et $\Psi(k) = 3k/2$ sinon. De plus, si la conjecture d'Erdos [8] est vraie alors $\Psi(k) = k$ pour tout k.

Théorème 3 *Soit n,k des entiers vérifiant $1 < k \leqslant \log n$. Soit un algorithme distribué A qui calcule pour tout graphe à n sommets un spanner de taille plus petite que $c \cdot n^{1+1/\Psi(k-1)}$ et un facteur d'étirement $\Psi(k)$. Alors, l'algorithme A a une complexité en temps au moins k.*

Théorème 4 *Soit k,ε tel que $1 < k \leqslant \log n$, et $1/\varepsilon > 3k^2 - k$. Soit un algorithme distribué A qui calcule pour tout graphe à n sommets un spanner de taille plus petite que $\lambda \cdot n^{1+1/\Psi(k)}$ en temps t, pour $k-3 \leqslant t \leqslant n^{\varepsilon}$ et $\lambda \leqslant n^{\varepsilon}$. Alors, il existe un graphe G et deux sommets u,v à distance $d_G(u,v) = \Omega(tn^{\varepsilon})$ tel que dans le spanner H calculé par l'algorithme A, on a :*

$$d_H(u,v) \ > \ \left(1 + \frac{k-1}{t+1}\right) \cdot d_G(u,v) + 2k - 3 \ .$$

References

[1] Ittai Abraham, Cyril Gavoille, and Dahlia Malkhi. On space-stretch trade-offs: Upper bounds. In 18^{th} *Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 207–216. ACM Press, July 2006.

[2] Baruch Awerbuch. Complexity of network synchronization. *Journal of the ACM*, 32(4):804–823, 1985.

[3] Baruch Awerbuch, Bonnie Berger, Lenore Jennifer Cowen, and David Peleg. Fast distributed network decompositions and covers. *Journal of Parallel and Distributed Computing*, 39(2):105–114, 1996.

[4] Surender Baswana, Telikepalli Kavitha, Kurt Mehlhorn, and Seth Pettie. New constructions of (α,β)-spanners and purely additive spanners. In 16^{th} *Symp. on Dis. Algorithms (SODA)*, pages 672–681. ACM-SIAM, 2005.

[5] Surender Baswana and Sandeep Sen. A simple and linear time randomized algorithm for computing sparse spanners in weighted graphs. *Random Structures and Algorithms*, 30(4):532–563, July 2007.

[6] Bilel Derbel, Cyril Gavoille, and David Peleg. Deterministic distributed construction of linear stretch spanners in polylogarithmic time. In 21^{st} *Symp. on Dist. Comp. (DISC)*, pages 179–192, 2007.

[7] M. Elkin and D. Peleg. $(1+\varepsilon,\beta)$-spanner constructions for general graphs. *J. on Comp.*, 33(3):608–631, 2004.

[8] P. Erdös and M Simonovits. Compactness results in extremal graph theory. *Combi.*, 2(3):275–288, 1982.

[9] Nathan Linial. Locality in distributed graphs algorithms. *SIAM Journal on Computing*, 21(1):193–201, 1992.

[10] D. Peleg and E. Upfal. A trade-off between space and efficiency for routing tables. *J. ACM*, 36(3):510–530, 1989.