Expansion connexe dans les réseaux

Nicolas NISSE*

3 septembre 2004

Résumé

Ce stage a pour cadre les problèmes d'encerclement et d'expansion dans un graphe. Ces problèmes sont motivés par leur étroite connexion avec des paramètres fondamentaux de graphes, dont en particulier quelques invariants de la théorie des mineurs comme la largeur d'arborescence et la séparation sommet. Ils sont également motivés par leur étroite connexion avec des questions pratiques comme la sécurité dans les réseaux ou l'extension de territoires. Nous avons repris l'étude de ces paramètres en imposant une contrainte de connexité. Dans ce stage, nous avons principalement investigué la décomposition arborescente connexe, et nous avons généralisé l'étude au cas q-connexe, pour q quelconque.

Mots clés : Décomposition arborescente, décomposition en branches, expansion, encerclement.

Rapport de stage du Master recherche Informatique d'Orsay
Effectué du 5/04 au 3/09 2004
Au sein de l'équipe "Théorie des Graphes et Fondement des Communications"
L.R.I., Bâtiment 490
Université Paris-Sud-Orsay

Responsable de stage : Pierre FRAIGNIAUD (CNRS, LRI)

*LRI, Université Paris-Sud, 91405 Orsay, France. Email : *nicolas.nisse@lri.fr*.

Remerciements

Ces remerciements vont tout d'abord à Pierre Fraigniaud qui m'a accueilli dans son équipe. Je le remercie pour les précieux conseils et toute l'aide qu'il m'a apporté pendant ce stage. Merci également de m'avoir permis d'assiter à de nombreux séminaires et conférences et de m'avoir intégré dans la communauté.

Je remercie également tous ceux qui m'ont apporté leurs conseils : David Ilcinkas, Philippe Gauron, Lynda Gastal, Taoufik Faik, Wadie Benajam.

Merci enfin à tous ceux qui ont participé de près ou de loin à ce stage.

Table des matières

1 Introduction

1.1 Encerclement et expansion

Les problèmes d'encerclement et d'expansion trouvent leurs motivations dans différents domaines, comme l'économie (stratégie de conquête) ou la sécurité dans les réseaux (capture d'un agent hostile). Pour illustrer ces problèmes, considérons le cas d'un réseau dans lequel un intrus doit être capturé par un groupe d'agents mobiles. Les réseaux supportant des agents mobiles sont en effet des environnements très sensibles à l'intrusion d'agents hostiles (par exemple des virus) et les dommages que peuvent générer ces intrus ont motivé de nombreuses recherches sur la détection d'intrus et leur capture par des agents mobiles. Un problème typiquement considéré consiste en l'élaboration d'une stratégie permettant la capture de l'intrus en minimisant le nombre d'agents. Ce problème entre dans la catégorie de problèmes connue sous le nom d'*encerclement* dans un graphe ("graph-searching" en anglais) introduite par Breish [7] et Parson [19]. L'intrus est supposé arbitrairement rapide, et peut connaître en permanence la position des agents à sa recherche. Il est capturé s'il est entouré d'agents. Une autre manière d'étudier le problème est de considérer un réseau contaminé que l'on désire nettoyer. Plus formellement, étant donné un ensemble d'arêtes "contaminées" d'un graphe, et un ensemble de "chercheurs", le but est de "nettoyer" ce graphe. Une stratégie d'encerclement est définie comme une suite d'opérations élémentaires, au nombre de trois. A chaque étape, on effectue une des opérations suivantes :

1. placer un chercheur sur un sommet du graphe ;
2. déplacer un chercheur le long d'une arête ;
3. supprimer un chercheur d'un sommet du graphe.

Une stratégie d'encerclement est une suite de telles opérations qui résulte en le nettoyage complet des arêtes du graphe. Dans le cas du nettoyage *par arête* ("edge-search" en anglais), une arête est nettoyée si elle est traversée par un chercheur. Elle reste "propre" si chacune de ses extrémités est soit "gardée" par un chercheur, soit incidente à des arêtes propres. Etant donné un graphe G, on note $\mathbf{s}(G)$ le nombre minimal de chercheurs nécessaire à une stratégie d'encerclement par arête pour ce graphe, c'est-à-dire une stratégie qui résulte en le nettoyage de toutes les arêtes.

Plusieurs autres types de nettoyage ont été considérés. Par exemple, dans la variante proposée par Kirousis et Papadimitriou [17], une arête est nettoyée si chacune de ses deux extrémités est occupée par un chercheur (elle n'a pas besoin d'être traversée). On parle alors de nettoyage *par sommet* ("node-search" en anglais), et on note $\mathbf{ns}(G)$ le nombre minimal de chercheurs nécessaires à une stratégie d'encerclement par sommet pour le graphe G. Un autre modèle, défini par Bienstock et Seymour [3], combine les deux versions précédentes : une arête est nettoyée si elle est traversée par un chercheur, ou si ses deux extrémités sont occupées par des chercheurs. On parle alors d'encerclement *mixte* ("mixed-search" en anglais).

Megiddo, Hakimini, Garey, Johnson et Papadimitriou [18] ont démontré que le problème de déterminer si $\mathbf{s}(G) \leq k$ est NP-difficile dans le cas général. Ils ont également montré que le problème devient linéaire dans le cas particulier des arbres, et ont proposé un algorithme linéaire calculant $\mathbf{s}(T)$ pour tout arbre T, ainsi qu'un algorithme en $O(n \log n)$ retournant une stratégie d'encerclement optimale pour tout arbre[1]. Skodinis [22] a ensuite donné un algorithme linéaire renvoyant une stratégie d'encerclement optimale dans le cas des arbres. Lapaugh [16] a prouvé que s'il existe une stratégie d'encerclement utilisant au plus k chercheurs, alors il existe une stratégie d'encerclement "monotone" utilisant au plus k chercheurs. Une stratégie d'encerclement est dite *monotone* s'il n'y

[1] Dans ce rapport, log désigne le logarithme en base 2.

a pas de recontamination, c'est-à-dire qu'une arête reste propre une fois nettoyée. Autrement dit, Lapaugh a démontré que la recontamination n'apporte rien. La combinaison de [18] et [16] montre que le problème de déterminer si $\mathbf{s}(G) \leq k$ est NP-complet.

Le problème de l'encerclement revient en fait à trouver un ordre dans lequel parcourir les sommets ou les arêtes d'un graphe. Ce problème est donc étroitement lié à une notion de numérotation des sommets d'un graphe ("linear layout" en anglais). Plus précisément, étant donné un graphe $G = (V, E)$ de n sommets, et une numérotation $L : V \to \{1, \cdots, n\}$ des sommets, on note $V_L(i)$ l'ensemble des sommets x tels que (1) $L(x) \leq i$ et (2) il existe une arête $\{x, y\} \in E$ avec $L(y) > i$. La *séparation sommet* $\mathbf{vs}_L(G)$ de G pour la numérotation L est définie comme $\max_{i=1,\cdots,n} V_L(i)$. La séparation sommet $\mathbf{vs}(G)$ de G est égale au minimum de $\mathbf{vs}_L(G)$ pour toutes les numerotations L. Ellis, Sudborough et Turner [9] ont fait le lien formel entre $\mathbf{s}(G)$ et $\mathbf{vs}(G)$, en prouvant que

$$\mathbf{vs}(G) \leq \mathbf{s}(G) \leq \mathbf{vs}(G) + 2$$

pour tout graphe G. Ils ont également trouvé un algorithme linéaire calculant $\mathbf{vs}(T)$ pour tout arbre T. Par ailleurs, Kirousis et Papadimitriou [17] ont prouvé que

$$\mathbf{ns}(G) = \mathbf{vs}(G) + 1.$$

Les notions d'encerclement dans un graphe présentent un autre intêret pour leurs relations étroite avec des propriétés intrinsèques de graphes, invariantes par mineur. En particulier, les largeurs des décompositions arborescentes **tw** ("treewidth" en anglais), linéaires **pw** ("pathwidth" en anglais), et en branches **bw** ("branchwidth" en anglais) [10, 20, 21] peuvent toutes être définies en termes d'encerclement. Par exemple, une fois que l'on a déterminé une décomposition linéaire d'un graphe, on peut en déduire une stratégie d'encerclement dans le cas où les chercheurs n'ont aucune connaissance de la position de l'intrus. Ainsi, Kinnersley [15] a démontré que, pour tout graphe G,

$$\mathbf{pw}(G) = \mathbf{vs}(G).$$

Dans le cas où les chercheurs voient l'intrus, une décomposition arborescente du graphe permet de déterminer une stratégie d'encerclement utilisant $\mathbf{tw}(G)$ chercheurs. Enfin, dans le cas où l'intrus est immobile, on est ramené au problème de l'exploration de graphe [13].

La notion d'expansion est également en étroite relation avec le problème d'encerclement. Elle a été définie comme suit. Soit un ensemble M, soit δ une fonction associant un entier à chaque sous-ensemble de M. Une k-expansion de M est une famille $(X_0, \cdots, X_r)$ de sous-ensembles de M. satisfaisant :

1. $X_0 = \emptyset$ et $X_r = E(G)$;
2. pour tout $0 \leq i \leq r$, $|\delta(X_i)| \leq k$;
3. pour tout $0 \leq i \leq r$, $|X_{i+1} - X_i| \leq 1$.

Une k-expansion d'un graphe G est une k-expansion de $E(G)$, avec δ la fonction qui, à tout $X \subset E(G)$ associe le nombre de sommets à la frontière entre X et $E(G) \setminus X$, c'est-à-dire le nombre de sommets incidents à une arête de X et à une arête de $E(G) \setminus X$. L'*expansion* d'un graphe G, notée $\mathbf{x}(G)$, est le plus petit k tel qu'il existe une k-expansion de G. L'expansion est dite *monotone* si elle vérifie de plus :

$$\text{pour tout } i,\ 0 \leq i < r, X_i \subset X_{i+1}.$$

On note $\mathbf{mx}(G)$ le plus petit k tel qu'il existe une k-expansion monotone de G.

L'expansion est très liée à l'encerclement puisque, pour tout graphe G,

$$\mathbf{x}(G) \leq \mathbf{s}(G) \leq \mathbf{x}(G) + 1 \text{ et } \mathbf{x}(G) - 1 \leq \mathbf{pw}(G) \leq \mathbf{x}(G) + 1.$$

La notion d'expansion a notamment permis à Bienstock et Seymour [3] de simplifier significativement la preuve de Lapaugh en prouvant l'égalité $\mathbf{mx}(G) = \mathbf{x}(G)$ pour tout graphe G, d'où on déduit $\mathbf{ms}(G) = \mathbf{s}(G)$ où $\mathbf{ms}(G)$ dénote le nombre minimal de chercheurs dans une stratégie d'encerclement monotone.

Les définitions d'encerclement et d'expansion dans un graphe supposent que les chercheurs peuvent être déplacés dans le réseau sans contraintes. Ainsi, le modèle prévoit qu'à chaque étape un chercheur peut être placé n'importe où dans le réseau. De plus, un chercheur est supposé soit pouvoir se déplacer librement dans la partie contaminée du réseau, soit pouvoir être retiré du réseau puis placé ailleurs dans le réseau. Or, dans la pratique, le déplacement des chercheurs peut être limité (on ne déplace pas facilement un agent mobile ou un secouriste d'un point arbitraire d'un réseau ou d'une grotte à un autre). Egalement, il peut être préférable, pour des raisons de sécurité ou de contrôle, que les chercheurs ne se dispersent pas en petits groupes séparés par des zones contaminées. Ces contraintes ne sont cependant pas prises en compte par les stratégies d'encerclement définies ci-dessus. C'est dans le but d'améliorer ce défaut du modèle que Barrière, Flocchini, Fraigniaud et Santoro [1] ont introduit la notion de stratégie d'encerclement connexe. Dans ce cadre, à chaque étape, la partie "nettoyée" du graphe doit former un sous-graphe connexe. On note $\mathbf{cs}(G)$ (resp., $\mathbf{mcs}(G)$) le plus petit k tel qu'il existe une stratégie d'encerclement connexe (resp., monotone connexe) dans G utilisant k chercheurs.

Barrière et co. [1] ont prouvé que, pour tout arbre T, $\mathbf{mcs}(T) = \mathbf{cs}(T)$. D'autre part, Barrière, Fraigniaud, Santoro et Thilikos [2] ont montré que pour tout arbre T, $\mathbf{s}(T) = \mathbf{ms}(T) \leq \mathbf{cs}(T) = \mathbf{mcs}(T) \leq 2 * \mathbf{s}(T)$. Fomin, Fraigniaud et Thilikos [12] ont également défini l'expansion connexe (resp., monotone connexe) comme une expansion $(X_0, \cdots, X_r)$ telle que pour tout i, $1 \leq i \leq r$, X_i induit un sous-graphe connexe de G. On note $\mathbf{cx}(G)$ (resp., $\mathbf{mcx}(G)$), le plus petit k tel qu'il existe une k-extension connexe (resp. monotone connexe) dans G. Fomin et co. [12] ont montré que $\mathbf{cx}(G)/\mathbf{x}(G) \leq 1 + \log|E(G)|$ pour tout graphe connexe G. Pour prouver cette borne, ils ont défini la notion de décomposition en branche *connexe* $\mathbf{cbw}$ d'un graphe. Ils ont alors démontré que, pour tout graphe 2-connexe, $\mathbf{cbw}(G) = \mathbf{bw}(G)$, puis ont construit une stratégie d'expansion connexe de frontière au plus $(1 + \log|E(G)|) * \mathbf{cbw}(G)$.

1.2 Objectifs du stage

Du bref survol de la litérature ci-dessus, il apparait que plusieurs questions restent ouvertes. On a vu par exemple que, pour tout arbre T, $\mathbf{cs}(T) \leq 2 * \mathbf{s}(T)$, mais que pour tout graphe connexe G, la meilleure borne connue est $\mathbf{cx}(G) \leq (1 + \log|E(G)|)\mathbf{x}(G)$. Nous avons évoqué plus haut une question naturelle qui est de savoir s'il existe une constante qui borne le rapport $\mathbf{cx}(G)/\mathbf{x}(G)$ dans le cas des graphes connexes quelconques. De même, on sait que, pour tout graphe, la recontamination n'apporte rien pour l'encerclement ou l'expansion. Dans le cas connexe, ce résultat reste vrai pour le cas spécifique des arbres. A ce jour, les recherches ont cependant échouer è étendre ce résutat aux graphes connexes quelconques.

Il s'agit également d'étendre les définitions d'expansion et de décomposition de graphe en considérant des graphes q-connexes avec $q \geq 1$ et d'étudier comment évoluent les différentes égalités et inégalités existantes dans le cas général et dans le cas connexe. Dans [12], Fomin, Fraigniaud et Thilikos s'interrogeaient sur l'existence d'une fonction f telle que pour tout graphe $f(q)$-connexe G, il existe une décomposition en branche k-connexe de G, de largeur $\mathbf{bw}(G)$. Il s'avère que définir

une telle décomposition en branches est délicate, nous avons donc préféré réaliser cette étude en privilégiant la décomposition arborescente.

1.3 Résultats

Notre première contribution a consisté à reprendre l'étude du rapport $\mathbf{cx}(G)/\mathbf{x}(G)$ sans utiliser l'intermédiaire relativement artificiel de la décomposition en branches, mais en se focalisant uniquement sur les notions directement liées à l'encerclement, soit les décompositions linéaires et arborescentes. Ainsi, nous avons défini la décomposition arborescente *connexe* pour tout graphe connexe. Nous prouvons alors notre premier résultat concernant la largeur d'arborescence connexe, notée **ctw** : pour tout graphe connexe G

$$\mathbf{tw}(G) = \mathbf{ctw}(G). \tag{1}$$

Nous présentons un algorithme polynomial qui à partir d'une décomposition arborescente quelconque de largeur k renvoie une décomposition arborescente connexe de largeur $\leq k$. Nous utilisons ensuite l'égalité 1 pour établir une nouvelle preuve du résultat de Fomin, Fraigniaud et Thilikos [12] : $\mathbf{cx}(G)/\mathbf{x}(G) \leq 1 + \log |E(G)|$.

Nous définissons et présentons également plusieurs résultats concernant la décomposition linéaire connexe. Ainsi, nous savons que la largeur linéaire et l'expansion d'un graphe ont même ordre de grandeur. Nous montrons que l'ordre de grandeur de la largeur linéaire connexe et celui de l'expansion connexe peuvent être différents. En effet, nous verrons que la largeur linéaire connexe impose la connexité de la partie nettoyée du réseau mais aussi celle de la partie encore contaminée. En particulier, nous prouvons que pour tout graphe connexe G :

$$\mathbf{cx}(G) \leq \mathbf{cpw}(G) + 1. \tag{2}$$

et que, pour tout n, il existe un arbre T de n sommets tel que :

$$\lceil n/3 \rceil \leq \mathbf{cpw}(T)/\mathbf{cx}(T). \tag{3}$$

Nous prouvons que déterminer la largeur linéaire connexe d'un graphe est aussi difficile que de déterminer sa largeur linéaire. Plus exactement, nous prouvons que :

$$\textit{Le probleme de savoir si } \mathbf{cpw}(G) \leq k \textit{ est } NP - difficile. \tag{4}$$

La section 2.5.3 décrit un algorithme polynomial qui renvoie la décomposition linéaire connexe optimale de tout arbre. La dernière partie est consacrée à l'extension des notions de k-expansion et décomposition arborescente dans le cas q-connexe avec $q \geq 1$. En particulier, nous étendons le résultat selon lequel pour tout arbre T, $\mathbf{mcx}(T) = \mathbf{cx}(T)$. Ainsi, nous généralisons ce résultat á la q-connexité en définissant la classe C_q des graphes q-connexes et B_{q+1}-libre, où B_q est le graphe composé de deux sommets reliés par q arêtes. Nous définissons également les notions d'expansion q-connexe (resp. monotone q-connexe), notée $\mathbf{cx}_q$ (resp. $\mathbf{mcx}_q$). Nous prouvons alors que, pour tout graphe $G \in C_q$, $\mathbf{mcx}_q(T) = \mathbf{cx}_q(T)$. Pour $q = 1$, on retrouve le résultat concernant les arbres puisque C_1 est exactement la classe des arbres.

Pour finir, nous définissons la largeur d'arborescence q-connexe (resp. linéqire q-connexe), notée $\mathbf{ctw}_q$ (resp. $\mathbf{cpw}_q$)et nous prouvons que pour tout graphe q-connexe G :

$$\mathbf{cx}_q(G) \leq \mathbf{cpw}_q(G) + 1. \tag{5}$$

Nous montrons également que pour tout $q > 1$, et pour tout n, il existe un graphe q-connexe d'au moins n sommets G, tel que :

$$\mathbf{tw}(G) = q \text{ et } \mathbf{ctw}_q(G) = n - 1. \tag{6}$$

On voit ainsi qu'imposer la q-connexité lors de la stratégie d'encerclement est une contrainte très forte.

2 Largeur d'arborescence et expansion connexe

Cette section se découpe principalement en trois sous-parties. Dans la section 2.2, nous définissons la notion de décomposition arborescente connexe. Nous allons voir qu'il existe deux définitions "naturelles" de ce concept mais qu'une seule de ces définitions est porteuse de développement interessant. Pour cette définition, nous prouvons l'égalité entre la largeur d'arborescence et la largeur d'arborescence connexe. La section 2.4 présente une nouvelle preuve de l'inégalité $\mathbf{cx}(G)/\mathbf{x}(G) \leq 1+\log n$ pour tout graphe connexe G de n sommets. Enfin, la section 2.5 est consacrée à la décomposition linéaire connexe. Nous commençons par quelques rappels.

2.1 Quelques rappels

Les notions de décompositions arborescente, linéaire ou en branches ont été introduites par Robertson et Seymour [10, 20, 21] dans le cadre de la théorie des mineurs. La définition suivante est centrale dans notre exposé.

Definition 1 *Soit G un graphe, T un arbre et $X = (X_v)_{v \in V(T)}$ une famille d'ensemble de sommets de G indexée par les sommets v de T. La paire (T, X) est appelée* décomposition arborescente *de G si elle satisfait les conditions suivantes :*

(C_1) $V(G) = \bigcup_{v \in V(T)} X_v$;

(C_2) *pour toute arête $e \in E(G)$, il existe un sommet $v \in V(T)$ tel que les deux extrémités de e sont dans X_v (dans la suite, on notera par abus de langage $e \in X_v$) ;*

(C_3) $(X_{v_1} \bigcap X_{v_3}) \subseteq X_{v_2}$ *pour tout $v_1, v_2, v_3 \in V(T)$ tels que v_2 sur le chemin reliant v_1 à v_3.*

Les conditions (C_1) et (C_2) de la définition 1 signifient que G est un sous-graphe du graphe obtenu en remplaçant chaque sommet X_v de $V(T)$ par une clique liant les sommets de G appartenant à X_v. La condition (C_3) implique que pour tout $v \in V(G)$, l'ensemble $\{t \in V(T) : v \in X_t\}$ forme un sous-arbre de T. La largeur d'une décomposition arborescente (X, T) est égale à $\max\{|X_v| - 1 \mid v \in V(T)\}$. La *largeur d'arborescence* (ou *treewidth*) $\mathbf{tw}(G)$ de G est la largeur minimale parmi toutes les décompositions arborescentes de G. Ce stage est particulièrement lié à un cas particulier de décomposition arborescente :

Definition 2 *On appelle décomposition linéaire d'un graphe G une décomposition arborescente (T, X) telle que T est un chemin. On note $\mathbf{pw}(G)$ la largeur minimale parmi toutes les largeurs des décompositions linéaires de G (la notation vient de l'anglais* "pathwidth"*).*

Enfin, nous rappelons un dernier type de décomposition, parfois plus facilement manipulable que la décomposition arborescente.

Definition 3 *Une décomposition en branche d'un graphe G est un arbre T dont les sommets internes sont de degré 3 et une bijection entre les feuilles de T et les arêtes de G.*

Soit T une décomposition en branches d'un graphe G. Etant donnée une arête e de T, il résulte de la suppression de e dans T deux arbres $T_1^{(e)}$ et $T_2^{(e)}$. On définit alors une *e-coupe* comme la paire $(E_1^{(e)}, E_2^{(e)})$ où $E_i^{(e)} \subset E(G)$ est l'ensemble des feuilles de $T_i^{(e)}$. On pose alors $\omega(T) = max_e|\delta(E_1^{(e)})|$ (où δ est la fonction "frontière" définie plus haut) pour désigner la largeur de la décomposition en branche T de G. La largeur minimale de toutes les largeurs des décompositions en branches (ou *branchwidth*) de G, notée $\mathbf{bw}(G)$, est égale à $min\{\omega(T) \mid T$ est une décomposition en branche de $G\}$. Une décomposition en branche est dite connexe, lorsque pour toute arête e de T, $E_1^{(e)}$ et $E_2^{(e)}$ induisent chacun des sous-graphes connexes de G. On note $\mathbf{cbw}(G)$ la largeur minimale de toutes les décompositions en branches connexes de G.

2.2 Décomposition arborescente connexe

Dans cette partie, nous voulons définir une décomposition arborescente connexe (T, X), de la même façon que Fomin, Fraigniaud et Thilikos [12] ont défini une décomposition en branches connexe à partir des travaux réalisés par Robertson et Seymour sur les *Carving*. Leur définition les a conduit à démontrer que, pour tout graphe 2-connexe G, $\mathbf{cbw}(G) = \mathbf{bw}(G)$. Il est donc naturel d'essayer d'obtenir une égalité du même style entre largeur d'arborescence et largeur d'arborescence connexe. Une première idée de définition serait d'imposer la connexité des sous-graphes induits par chaque ensemble $X_v, v \in V(T)$. La proposition suivante montre cependant qu'avec une telle définition, on ne peut pas avoir égalité entre largeur d'arborescence et largeur d'arborescence connexe. Dans cette proposition, pour tout graphe connexe G, on note $\mathbf{tw'}(G)$ la largeur minimale de toutes les largeurs des décompositions arborescentes (T, X) de G telles que pour tout $v \in V(T)$, le sous-graphe induit par X_v soit connexe.

Proposition 1 *Pour tout $n \geq 3$, il existe un graphe G de n sommets tel que $\mathbf{tw}(G) = 2$ et $\mathbf{tw'}(G) = \lceil n/2 \rceil$.*

Preuve. Soit $n \geq 3$ et soit C_n un anneau de n sommets. On note $a_0, \cdots, a_{n-1}$ les sommets de l'anneau. On sait que $\mathbf{tw}(C_n) = 2$. On va montrer que $\mathbf{tw'}(C_n) = \lceil n/2 \rceil$. Si $n \leq 4$ le résultat est trivial. Supposons donc que $n \geq 5$. Montrons tout d'abord que $\mathbf{tw'}(C_n) \leq \lceil n/2 \rceil$. On pose $X_1 = \{a_0, a_1, \cdots, a_{\lceil n/2 \rceil}\}$ et $X_2 = \{a_{\lceil n/2 \rceil}, \cdots, a_{n-1}, a_0\}$. (X_1, X_2) est une décomposition arborescente de C_n de largeur $\lceil n/2 \rceil$, vérifiant la connexité des sous-graphes induits par chaque ensemble X_1 et X_2.

Montrons maintenant que $\lceil n/2 \rceil \leq \mathbf{tw'}(C_n)$. Supposons dans le but d'arriver à une contradiction que $\mathbf{tw'}(C_n) < \lceil n/2 \rceil$. Soit (T, X) une décomposition arborescente de C_n de largeur $k < \lceil n/2 \rceil$ et telle que pour tout $v \in V(T)$, X_v induit un sous-graphe connexe de G. Si $|V(T)| = 1$, alors la largeur est égale à $n - 1$ de manière évidente. Si $|V(T)| = 2$, alors $k < \lceil n/2 \rceil$ implique que X_1 et X_2 contiennent chacun au plus $\lceil n/2 \rceil$ sommets, c'est-à-dire au plus $\lceil n/2 \rceil - 1$ arêtes, soit en tout, au plus $n - 1$ arêtes. Donc il existe $e \in E(C_n)$ qui n'appartient pas à la décomposition, ce qui n'est pas possible. Donc $|V(T)| \geq 3$. A partir d'une décomposition arborescente (T, X) de C_n, vérifiant la connexité des sous-graphes induits par chaque ensemble de X, on peut obtenir une décomposition (T', X') de même largeur, vérifiant la même propriété, et telle que, pour tout $0 \leq i \leq n - 1$, il existe au plus deux sommets $v \in T'$ tels que $a_i \in X'_v$. Supposons en effet qu'il existe $0 \leq i \leq n - 1$ tel que $|\{v \in T \mid a_i \in X_v\}| \geq 3$. Soit alors $A = \{v \in T \mid a_i \in X_v\}$. Pour tout $v \in A$, il existe $0 \leq h, \ell \leq n - 1$, a_i se trouve entre a_h et a_ℓ sur C_n, et $X_v = \{a_h, \cdots, a_i, \cdots, a_\ell\}$, voir Figure 1. Donc il existe $0 \leq h, \ell \leq n - 1$ tels que $\bigcup_{t \in T_A} V_t = \{a_h, a_{h+1}, \cdots, a_i, \cdots, a_{\ell-1}, a_\ell\}$.

On peut alors comprimer T_A en (v_1, v_2) avec $X_{v_1} = \{a_h, a_{h+1}, \cdots, a_i\}$ et $X_{v_2} = \{a_i, a_{i+1}, \cdots, a_\ell\}$, c'est-à-dire remplacer le sous-arbre T_A par les seuls sommets v_1 et v_2. Clairement, T' est une décomposition arborescente de C_n. De plus, cette transformation n'augmente pas la largeur de la

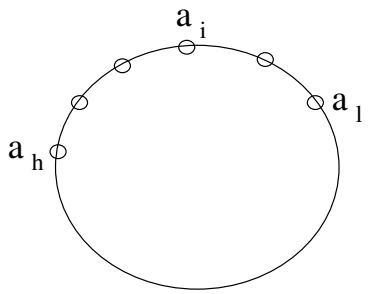

FIG. 1 – Représentation de la position de a_i dans C_n

décomposition. En particulier, T' est un arbre. En itérant la transformation précédente, on obtient donc une décomposition arborescente (T', X') telle que, pour tout i, il existe au plus deux sommets $v \in V(T')$ tels que $a_i \in X'_v$. La largeur de (T', X') est $\leq k$, donc $< \lceil n/2 \rceil$. Il existe donc une arête (v_j, v_{j+1}) pour tout $1 \leq j < r$ et une arête (v_r, v_1). Donc T' contient un cycle et ne peut pas être un arbre. Donc T n'est pas un arbre, d'où la contradiction. ■

La proposition 1 montre qu'imposer la connexité des sous graphes induits par chacun des ensembles X_v est une contrainte très forte qui ne permet pas de développement intéressant. Dans la suite, nous allons plutôt nous inspirer de la définition de la décomposition en branches connexe proposée par Fomin, Fraigniaud et Thilikos dans [12]. Considérons une décomposition arborescente (T, X) d'un graphe G. Etant donnée une arête e de T, il résulte de la suppression de e dans T deux arbres $T_1^{(e)}$ et $T_2^{(e)}$. On définit alors une *e-coupe* comme la paire $(G_1^{(e)}, G_2^{(e)})$ où $G_i^{(e)}$ est le sous-graphe de G induit par $\bigcup_{v \in T_i^{(e)}} X_v$, $i = 1, 2$. Dans la suite de l'exposé, on trouvera aussi la notation $G[T_i]$ pour faire référence à un tel graphe induit, c'est-à-dire $G[T_i] = \bigcup_{v \in T_i^{(e)}} X_v$. On dit qu'une e-coupe est *connexe* si les deux sous-graphes induits par $G_i^{(e)}$, $i = 1, 2$ sont chacun connexes. De cette notion d'e-coupe connexe découle la définition suivante.

Definition 4 *Une décomposition arborescente (T, X) d'un graphe G est connexe si : pour toute arête e de T, la e-coupe correspondante est connexe.*

La largeur d'arborescence connexe de G, notée $\mathbf{ctw}(G)$, est la largeur minimale de toutes les largeurs des décompositions arborescentes connexes de G. Une décomposition arborescente connexe de G ne peut évidemment exister que si G est connexe. Nous en arrivons ainsi au premier résultat important de ce rapport.

Theoreme 1 *Pour tout graphe connexe G, $\mathbf{tw}(G) = \mathbf{ctw}(G)$. De plus, il existe un algorithme polynomial qui, étant donnée une décomposition arborescente de largeur k de G, renvoie une décomposition arborescente connexe de G, de largeur $\leq k$.*

La suite de la section 2.2 décrit une preuve constructive du théorème 1 en décrivant un algorithme polynomial qui, à partir d'une décomposition arborescente (T, X) de largeur k d'un graphe connexe G, renvoie une décomposition arborescente connexe (T', X') de G, de largeur $\leq k$. Cet algorithme se décompose en deux parties utilisant chacune une même transformation, appelée *éclate* dans la suite. On considère la décomposition (T, X) comme un arbre enraciné en un sommet arbitraire u. La première partie de l'algorithme part des feuilles et remonte vers la racine u en s'assurant que

chaque sommet satisfait une propriété de sous-connexité définie ci-dessous. La seconde partie de l'algorithme parcourt l'arbre de la racine aux feuilles afin de rendre connexe la décomposition.

On note (T_u, X) la décomposition arborescente (T, X) avec T enraciné en u. Soit $t \in V(T)$, on note $T^{(t)}$ le sous arbre de T_u enraciné en t.

Definition 5 *Soit G un graphe et (T_u, X) une décomposition arborescente de G. On dit que la décomposition (T_u, X) est sous-connexe en $v \in V(T)$ si, pour tout $w \in T^{(v)}$, $G[T^{(w)}]$ est connexe. (T_u, X) est dite sous-connexe si elle est sous-connexe en u.*

Notons qu'on peut alternativement définir la sous-connexité de (T, X) en v comme : (1) $G[T^{(v)}]$ est connexe, et (2) (T, X) est sous-connexe en w, pour tout fils w de v dans T_u.

Nous décrivons ci-dessous une procédure élémentaire, nommée *éclate*, qui, à partir d'une décomposition arborescente (T_u, X) de largeur k d'un graphe connexe G, et d'un sommet $v \in V(T_u)$ tel que : (1) (T_u, X) n'est pas sous-connexe en v et (2) pour tout fils w de v, (T_u, X) est sous-connexe en w, construit une décomposition arborescente (T'_u, X') de G, de largeur $\leq k$, en remplaçant le sommet v par un ensemble de nouveaux sommets tels que, pour tout sommet créé x, (S_u, Y) est sous-connexe en x. De plus, la procédure *eclate* vérifie que pour tout $w \in T_u \setminus \{v\}$, si (T_u, X) ést sous-connexe en w, alors (S_u, Y) est encore sous-connexe en w.

Transformation *eclate*. Soit (T_u, X) une décomposition arborescente de largeur k d'un graphe G. Soit $v \in V(T)$ tel que

- (T_u, X) n'est pas sous-connexe en v ;
- pour tout fils w de v, (T_u, X) est sous-connexe en w.

X_v n'induit donc pas un sous-graphe connexe de G. Supposons que $X_v = \bigcup_{i=0,\cdots,r} X_v^{(i)}$, où $X_v^{(i)}$ sont les composantes connexes de X_v. Notons que $r > 1$. Soit v' le père de v dans T_u. La transformation *eclate* consiste en :

Cas 1 : v est une feuille de T. Soit S l'arbre obtenu à partir de T privé du sommet v auquel on ajoute r sommets $v_1, \cdots, v_r$ comme fils de v'. On remplace alors X par $Y = \{Y_t, t \in S\}$ où pour tout $t \in V(T) \bigcap V(S)$, $Y_t = X_t$ et, pour tout $i \in \{1, \cdots, r\}$, $Y_{v_i} = X_v^{(i)}$.

Cas 2 : v possède $s \neq 0$ fils. Soient $w_1, \cdots, w_s$ les fils de v. Comme $G[T^{(v)}]$ n'est pas connexe, on pose $\{Z_i, i = 1, \cdots, \ell\}$ ses composantes connexes. Alors, il existe une partition $\{I_i, i = 1, \cdots, \ell\}$ de $\{1, \cdots, r\}$ et une partition $\{J_i, i = 1, \cdots, \ell\}$ de $\{1, \cdots, s\}$ telles que, pour tout $i = 1, \cdots, \ell$,

$$Z_i = \left(\cup_{j \in J_i} V(G[T^{(w_j)}]) \right) \bigcup \left(\cup_{j \in I_i} X_v^{(j)} \right)$$

Dans S, v' garde ses fils, excepté v qui est remplacé par ℓ sommets v_i, $i = 1, \cdots, \ell$, où v_i est le père de chaque w_j, $j \in J_i$. Finalement, Y est défini par $Y = \{Y_t, t \in V(S)\}$ tel que pour tout $t \in V(T) \bigcap V(S)$, $Y_t = X_t$ et pour tout $i \in \{1, \cdots, \ell\}$, $Y_{v_i} = \bigcup_{j \in I_i} X_v^{(j)}$.

Remarque Supposons que pour tout fils w de u, (T_u, X) est sous-connexe en w. Comme $G = G[T_u]$ est connexe, alors (T_u, X) est sous-connexe.

Lemme 1 *Soit (T_u, V) une décomposition arborescente de largeur k d'un graphe G. Soit $v \in T \setminus \{u\}$ tel que* (1) *(T_u, X) n'est pas sous-connexe en v, et* (2)*, pour tout fils w de v, (T_u, X) est sous-connexe en w. Alors la transformation* eclate *appliquée au sommet v de T_u, retourne une décomposition arborescente de G de largeur $\leq k$.*

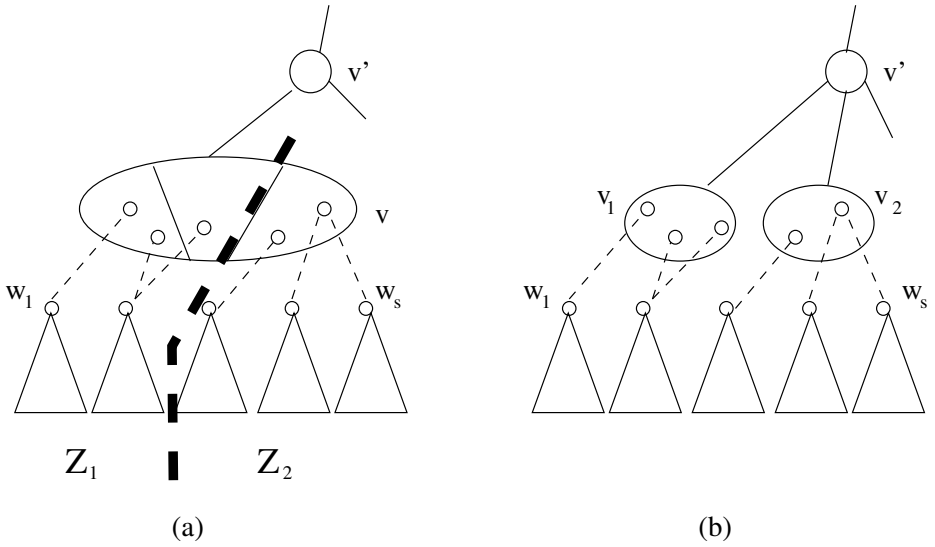

FIG. 2 – Illustration de la procédure *eclate* dans le cas 2. La figure (a) représente T_u avant transformation, et la figure (b) représente S_u après application de la procédure

Preuve. On se place dans le deuxième cas de la transformation, montrons que (S, Y) satisfait les trois conditions d'une décomposition arborescente :

1. $\bigcup_{t\in V(S)} Y_t = (\cup_{t\in V(T)\setminus\{v\}} Y_t) \bigcup (\cup_{i=1,\cdots,r} Y_{v_i}) = (\cup_{t\in V(T)\setminus\{v\}} X_t) \bigcup X_v = \cup_{t\in V(T)} X_t = V(G)$.
2. Soit $e = (x, y) \in E(G)$. S'il existe $t \in V(T) \setminus \{v\}$ tel que $\{x, y\} \subset X_t$ alors, cela reste vrai dans (S, Y). Sinon, $\{x, y\} \subset X_v$, plaçons nous dans ce cas. Puisque $\{x, y\}$ induit un sous-graphe connexe, x et y appartiennent à une même composante connexe de X_v, donc il existe $1 \leq i \leq r$ tel que $\{x, y\} \subset X_v^{(i)}$. On en déduit que $\{x, y\} \subset Y_{v_i}$. Ainsi pour toute arête $e \in E(G)$, il existe $t \in V(S)$ tel que les deux extremités de e sont dans Y_t
3. Soit u_1, u_2, u_3 trois sommets distincts de S avec u_2 sur le chemin entre u_1 et u_3.
   - Si $v_i \notin \{u_1, u_2, u_3\}$ pour tout $i = 1, \cdots, r$, alors on a la même structure que dans T, c'est-à-dire $Y_{u_1} \bigcap Y_{u_3} = X_{u_1} \bigcap X_{u_3} \subseteq X_{u_2} = Y_{u_2}$;
   - S'il existe $i \in \{1, \cdots, r\}$ tel que $u_2 = v_i$ et , $v_i \notin \{u_1, u_3\}$ pour tout $i = 1, \cdots, r$, alors, sans perte de généralité, supposons que $u_1 \in V(S^{(v_i)})$. Comme $u_2 = v_i$ est sur le chemin entre u_1 et u_3 dans S, alors le sommet v supprimé est sur le chemin entre u_1 et u_3 dans T. Soit $x \in Y_{u_1} \bigcap Y_{u_3}$. Comme $Y_{u_1} \bigcap Y_{u_3} = X_{u_1} \bigcap X_{u_3} \subseteq X_v$, alors $x \in X_v$. Donc x appartient dans (T, X) à la composante de X_v connexe avec $G[A]$, où A est le sous-arbre de T contenant u_1 et enraciné en un fils de v. Donc $x \in Y_{v_i}$.
   - Supposons qu'il existe $i, j \in \{1, \cdots, r\}$ tel que $u_1 = v_i$ et $u_3 = v_j$, nous allons montrer que cela aboutit à une contradiction. En effet, $X_v^{(i)} \bigcap X_v^{(j)} \neq \emptyset$. Donc $i = j$, ce qui contredit le fait que u_1 et u_3 sont disjoints ;
   - S'il existe $i \in \{1, \cdots, r\}$ tel que $u_1 = v_i$, et $u_3 \neq v_j$ pour tout $j \in \{1, \cdots, r\}$, alors $X_v^{(i)} \bigcap X_{u_3} \subseteq X_v \bigcap X_{u_3} \subseteq X_{u_2}$ puisque u_2 est sur le chemin entre u_1 et u_3 dans S, donc entre v et u_3 dans T.

Donc (S, Y) est une décomposition arborescente de G. La largeur de (S, Y) est au plus k car les ensembles Y_t sont des sous-ensembles des X_t. La démonstration du cas où v est une feuille est similaire. ■

La première phase de l'algorithme anoncé dans le théorème 1 est décrite dans l'algorithme 1.

Lemme 2 *Etant donnée une décomposition arborescente (T_u, X) d'un graphe connexe G, de largeur k, l'algorithme 1 retourne une décomposition arborescente sous-connexe (S_u, Y) de G, de largeur $\leq k$.*

Algorithm 1

ENTRÉES: G un graphe connexe, (T_u, X) une décomposition arborescente de G, de largeur k.
SORTIES: une décomposition arborescente (S_u, Y) de G, de largeur $\leq k$ sous-connexe.

Début
$(S_u, Y) \leftarrow (T_u, X)$.
$W = \{t \in V(T) \setminus \{u\} \mid (T_u, X)$ n'est pas sous-connexe en $t\}$.
tantque $W \neq \emptyset$ **faire**
Soit $t \in W$ tel que, t est une feuille, ou pour tout fils t' de t dans $T_u, t' \notin W$;
$(S_u, Y) \leftarrow eclate(G, (S_u, Y), t)$;
$W \leftarrow W \setminus \{t\}$;
fin tantque
Retourner (S_u, Y)
Fin

Preuve. D'après le lemme 1, la procédure *eclate* retourne une décomposition arborescente (S_u, Y) de G de largeur $\leq k$. A la fin de chaque passage dans la boucle *tant que*, W est exactement l'ensemble des sommets $t \in V(S) \setminus \{u\}$ tels que (S_u, Y) n'est pas sous-connexe en t. En effet, *eclate* supprime t qui était dans W et n'ajoute aucun sommet $x \in V(S)$ tel que (S_u, Y) n'est pas sous-connexe en x. Puisque $W = \emptyset$ à la fin de l'algorithme, (S_u, Y) est sous-connexe en chaque fils de u et comme G est connexe, (S_u, Y) est sous connexe. ■

La seconde phase de l'algorithme annoncé dans le théorème 1 est décrite dans l'algorithme 2.

Lemme 3 *Etant donnée une décomposition arborescente sous-connexe (T_u, X) d'un graphe connexe G, de largeur k, l'algorithme 2 retourne une décomposition arborescente connexe de G, de largeur $\leq k$.*

Preuve. Le même type de preuve que celle proposée pour le lemme 1 permet de montrer que le résultat (S_r, Y) renvoyé par l'algorithme 2 est une décomposition arborescente de largeur $\leq k$ de G. Montrons que pour tout e choisie dans la boucle tant que, la coupe correspondante est rendue connexe. Plus exactement, on va montrer qu'à la fin de chaque boucle de l'algorithme 2, on a (1) (S_r, Y) est une décomposition arborescente sous-connexe et (2) l'ensemble E est exactement l'ensemble des arêtes $e \in E(S)$ qui induisent une e-coupe qui n'est pas connexe dans S.

Supposons qu'à une étape de l'algorithme, au moment de l'entrée dans la boucle, (S_r, Y) soit une décomposition arborescente sous-connexe de G, telle que E est l'ensemble des arêtes $e \in E(S)$ pour lesquelles la coupe n'est pas connexe dans S. Soit $e = (v, w) \in E$ telle que v le père de w dans S_r et, pour toute arête e' sur le chemin entre r et v, e' n'est pas dans E. On suppose d'abord que $r \neq v$.

On introduit ici quelques notations utilisées dans la suite de la preuve :
- Soit v' le père de v dans S_r (éventuellement, on peut avoir $r = v'$) ;
- Soit $T^{(v')}$ le sous-arbre contenant v' obtenu en supprimant l'arête (v', v) de S. Le choix de e implique que $(v', v) \notin E$, donc $G[T^{(v')}]$ est un sous graphe connexe de G.

Algorithm 2

ENTRÉES: G un graphe connexe et (T_u, X) une décomposition arborescente sous-connexe de G, de largeur k.

SORTIES: une décomposition arborescente connexe (S_r, Y) de G, de largeur $\leq k$.

Début

$(S_r, Y) \leftarrow (T_u, X)$.

$E = \{e \in E(T) \mid$ la e-coupe de S_r n'est pas connexe$\}$

tantque $E \neq \emptyset$ **faire**

Choisir $e = (v, w) \in E$ telle que v est le père de w dans S_r, et, pour toute arête e' sur le chemin entre r et v, e' n'est pas dans E ;

$r \leftarrow w$; /* "rotation" de l'arbre, voir */

$I \leftarrow \{i \in \{1, \cdots, s\}$ *tel que* w_i *est le fils de* v *et* $\{w_i, v\} \in E\}$

$(S_r, Y) = eclate(G, (S_r, Y), v)$;

/* mise à jour des arêtes de E modifiées lors de l'application de la procédure *eclate* */

$E \leftarrow E \setminus \{e\}$;

pour tout $i \in I$ **faire**

$E \leftarrow (E \setminus \{w_i, v\}) \cup \{w_i, v_j\}$, où v_j est le père de w_i après application de la procédure *eclate* ;

fin pour

fin tantque

Retourner (S_r, Y)

Fin

– on pose $v_1, \cdots, v_\ell$ les fils de v excepté w et $T^{(1)}, \cdots, T^{(\ell)}$ les sous-arbres de S_r enracinés en ces sommets. Comme (S_r, Y) est sous-connexe, pour tout $\leq j \leq \ell$, $G[T^{(j)}]$ est un sous-graphe connexe de G.

Après la rotation, S est enraciné en w. Remarquons que $e \in E$ signifie que (S_w, Y) n'est pas sous-connexe en v. Par contre pour tout $t \neq v$, (S_w, Y) est sous-connexe en t. En effet, soit $t \in S \setminus \{v\}$. Si t n'est pas sur le chemin entre v et r, alors (S_w, Y) sous-connexe en t équivaut à (S_r, Y) sous-connexe en t. Donc (S_w, Y) est sous-connexe en t puisque (S_r, Y) est sous-connexe en t. Si t est sur le chemin entre v et r, soit alors t' le père de t dans S_r. Le choix de v implique que la $\{t, t'\}$-coupe est connexe, et donc (S_w, Y) est sous-connexe en t.

L'algorithme 2 applique la procédure $eclate(G, (S_r, Y), v)$. Comme, pour tout $1 \leq j \leq \ell$, $G[T^{(j)}]$ est un sous-graphe connexe de G, et comme $G[T^{(v')}]$ est un sous graphe connexe de G, on est bien dans les conditions d'utilisation de la procédure *eclate*. Donc en sortie de boucle, (S_r, Y) est une décomposition arborescente sous-connexe de G. ((S_u, Y) est sous-connexe en w puisque (S_u, Y) est sous-connexe en chaque des fils de w et que G est connexe.)

On montre tout d'abord que pour toute arête $e' \neq e$ de S, si e' induit une coupe connexe avant l'application de la procédure éclate, alors, e' induit une coupe connexe après application de la procédure. Soit $e' \neq e$ une arête de S. Soit T_1 et T_2 les sous-arbres résultant de la suppression de e' dans S. Comme $e' \neq e$, après application de la procédure *eclate*, e' est toujours une arête de S. Soit S_1 et S_2 les sous-arbres résultant de la suppression de e' dans S après l'application de la procédure *eclate*. Alors, pour $i \in \{1, 2\}$, $G[T_i] = G[S_i]$. Donc, si $e' \neq e$ induit une coupe connexe avant l'application de la procédure *eclate*, cette coupe reste connexe après.

Il reste à montrer que les arêtes créées forment des coupes connexes, ce qui permettra de prouver que l'on n'augmente pas le nombre des arêtes induisant des e-coupes non connexes de (S_r, Y) en sortie de boucle. Soit $T^{(w)}$ le sous-arbre de S_r enraciné en w. Comme (S_r, Y) est sous-connexe,

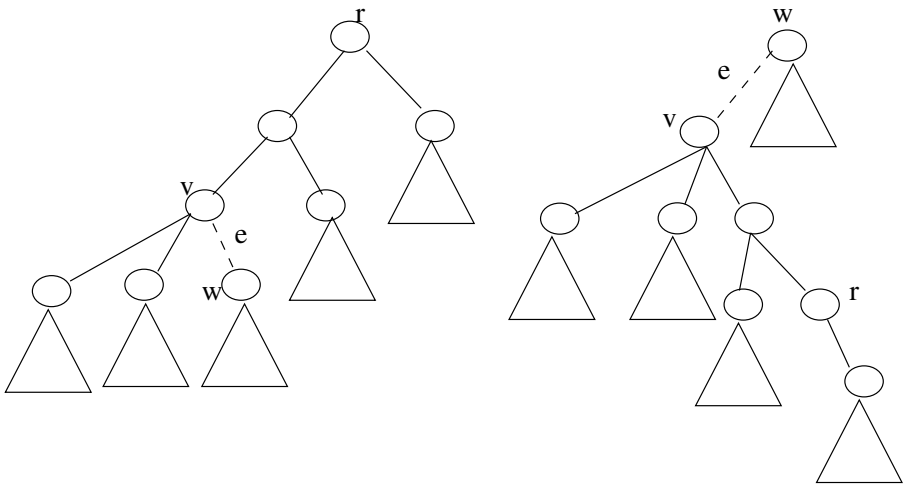

FIG. 3 – "Rotation" de l'arbre pendant l'application de l'algorithme 2. A gauche, l'arbre enraciné en r avant rotation. A droite, l'arbre enraciné en w, après rotation.

$G[T^{(w)}]$ est un sous-graphe connexe de G. Donc, toutes les arêtes créées par la procédure *eclate* induisent des coupes connexes. En effet, soit une arête e' créée lors de l'application de la procédure *eclate*. Par construction, une de ses extrémités est w. Soit t tel que $e' = \{w, t\}$ et soit $T^{(t)}$ le sous arbre enraciné en t. Montrons que $G[T^{(t)} \cup T^{(w)}]$ est connexe. D'une part $G[T^{(w)}]$ est un sous-graphe connexe de G. D'autre part, par construction, $G[T^{(t)}]$ est un sous-graphe connexe de G. Par ailleurs, comme G est connexe, $X_w \cap V(G[T^{(t)}]) \neq \emptyset$. Donc $G[T^{(t)} \cup T^{(w)}] = G[T^{(t)}] \cup G[T^{(w)}]$ est un sous graphe connexe de G. On en déduit que les arêtes créées correspondent à des coupes connexes. Donc, la procédure *eclate* supprime de S une arête e qui induit une coupe non connexe, pour la remplacer par des arêtes qui induisent des coupes connexes. Donc, l'instruction $E \leftarrow E \setminus \{e\}$ fait que E est bien l'ensemble des arêtes correspondant aux coupes non connexes de (S_r, Y) en sortie de boucle.

Le cas $r = v$ est similaire, si ce n'est qu'on n'a pas besoin de vérifier que les arêtes sur le chemin entre le père de v et r restent des e-coupes connexes après application de la procédure *eclate* puisque ni le père de v ni le chemin entre v et r ne sont définis dans ce cas.

Comme $|E|$ décroit strictement à chaque étape, l'algorithme se termine. ■

La preuve du théorème 1 découle directement des lemmes 2 et 3.

La section suivante présente l'étude de la complexité de l'algorithme précédent. Nous présentons également un algorithme approché qui à partir d'un graphe connexe renvoie une décomposition arborescente connexe, avec un rapport d'approximation au plus $\log \mathbf{tw}(G)$.

2.3 Complexité de l'algorithme et approximation

En prenant comme entrées un graphe de n sommets et une décomposition arborescente de ce graphe de largeur k, et comportant n sommets, nous montrons que l'algorithme a un temps d'exécution en $O(n\, k^2)$. Donc appliqué à une classe de graphes de largeur d'arborescence bornée, cet algorithme est linéaire en le nombre de sommets du graphe. Dans la deuxième partie de cette section, nous donnons un algorithme polynomial qui retourne une décomposition arborescente *connexe* avec un rapport d'approximation $O(\log \mathbf{tw}(G))$ par rapport à l'optimal.

Proposition 2 *Etant donnée une décomposition arborescente (T, X) d'un graphe connexe G, de largeur k, et telle que $|V(T)| = N$, l'algorithme 1 a un temps d'exécution en $O(N \cdot k^2)$.*

Lemme 4 *Soit (T_u, X) une décomposition arborescente de largeur k d'un graphe G. Soit $v \in V(T)$ vérifiant les conditions d'application de la procédure* eclate. *L'application de la transformation* eclate *au sommet v prend un temps :*

- *$O(k)$ si v est une feuille de T ;*
- *$O(k \cdot \sum_{v' \ fils \ de \ v} |X_{v'}|)$ sinon.*

Preuve. Soit v le sommet auquel on veut appliquer la procédure *eclate*, on suppose déterminées les composantes connexes de X_v. Si v est une feuille de T, le remplacement du sommet initial par autant de feuilles que de composantes connexes (dont le nombre est au plus k) prends un temps en $O(k)$. Supposons que v possède $r > 0$ fils. Pour tout $x \in X_v$, la procédure *eclate* teste alors l'ensemble des $y \in \bigcup_{w \ fils \ de \ v} X_w$. Comme $|X_v| \leq k$, cette opération demande un temps $O(k \cdot \sum_{w \ fils \ de \ v} |X_w|)$. ■

Preuve. de la proposition 2 Initialement, (T, X) est une décomposition arborescente de largeur k, telle que $|V(T)| = N$. Donc $\sum_{v \in V(T)} |X_v| \leq N \cdot k$. Avant de débuter l'exécution de l'algorithme, on peut déterminer les composantes connexes de chacun des X_v pour $v \in V(T)$. Etant donné $v \in V(T)$, déterminer les composantes connexes de X_v prend un temps $O(|X_v|^2)$ donc un temps $O(k^2)$. Déterminer les composantes connexes de tous les X_v prend donc un temps $O(k^2 \cdot N)$. L'algorithme 1 commence par traiter les feuilles de T. D'après le lemme 4, le traitement d'une feuille prend un temps $O(k)$. Donc, le traitement des feuilles nécessite un temps total en $O(k \cdot nombre\ de\ feuilles\ de\ T)$, donc $O(k \cdot N)$. L'algorithme traite ensuite les sommets internes de la décomposition initiale. Soit un sommet interne $v \in V(T)$. Notons H_v l'ensemble des fils de v dans la décomposition initiale. Lorsque que l'algorithme 1 traite le sommet v, ses descendants ont été traités, c'est-à-dire que ses fils peuvent avoir été divisés en de nouveaux fils. Notons H'_v l'ensemble des fils de v juste avant que l'algorithme ne traite ce sommet ; tous les descendant de v ont donc été traités. La procédure *eclate* est telle que $\sum_{w \in H_v} |X_w| = \sum_{w \in H'_v} |X_{w'}|$. D'après le lemme 4, le traitement du sommet v nécessite un temps $O(k \cdot \sum_{w \in H'_v} |X_w|)$, ce qui est donc égal à $O(k \cdot \sum_{w \in H_v} |X_w|)$. Pour traiter tous les sommets internes, le temps nécessaire est donc en $O(\sum_{v \ sommet\ interne\ de\ T}(k \cdot \sum_{w \in H_v} |X_w|))$. En effet, seuls les sommets initialement présents dans T sont traités par l'algorithme **??** et ils ne le sont qu'une fois. $\sum_{v \ sommet\ interne\ de\ T}(\sum_{w \in H_v} |X_w|) \leq \sum_{v \in V(T)} |X_v| \leq N \cdot k$. Donc le traitement des sommets internes prends un temps total $\leq O(k^2 \cdot N)$.

L'algorithme 1 nécessite donc un temps d'exécution en $O(k^2 \cdot N)$. ■

Proposition 3 *Etant donnée une décomposition arborescente (T, X) d'un graphe connexe G, de largeur k, et telle que $|V(T)| = N$, soit (T', X') la décomposition arborescente sous-connexe calculée par l'algorithme 1 à partir de (T, X). En prenant en entrée (T', X'), l'algorithme 2 a un temps d'exécution en $O(N \cdot k^3)$.*

Preuve. Cet algorithme se différencie de l'algorithme 1 dans la mesure où l'application de la procédure *eclate* n'est pas seulement effectuée sur les sommets présents dans l'arbre T'. En effet, étant donné un sommet $v \in V(T')$, il peut exister plusieurs arêtes *non-connexes* incidentes à v. L'application de la procédure *eclate* au sommet v supprime au moins une de ces arêtes, mais il peut subsister une ou plusieurs arêtes *non-connexes*. Dans l'arbre résultant, celles-ci sont incidentes à des sommets résultants de "l'éclatement" de v. L'algorithme 2 prévoit alors l'application de la procédure *eclate* à ces sommets et ce processus peut devoir être répété récursivement sur des sommets issus de

l'éclatement des sommets issus de v. Dans la suite, on dira qu'un sommet w est issu d'un sommet $v \in V(T)$, si il existe un sommet $w' \in v(T')$ tel que :

- $w' = v$, ou w' résulte de "l'éclatement" de v lors de l'application de l'algorithme 1 à T.
- w résulte de "l'éclatement" de w', ou, de manière récursive, de "l'éclatement" d'un sommet issu de w'.

Calculons tout d'abord la complexité de l'application de la procédure *eclate* à v ou à l'un des sommets issu de v. Soit w un tel sommet. Le lemme 4 assure que cette complexité est en $O(k \cdot \sum_{u \ fils \ de \ w} |X_u|)$, ce qui est majoré par $O(k \cdot \sum_{u \ voisins \ de \ w} |X_u|)$. Or tout voisin de w est issu d'un voisin de v dans l'arbre T (par construction de la procédure *eclate*). Donc $\sum_{u \ voisins \ de \ w} |X_u| \leq \sum_{u \ voisins \ de \ v} |X_u|$. Donc la complexité de l'application de la procédure *eclate* à v ou à l'un des sommets issu de v est en $O(k \cdot \sum_{u \ voisins \ de \ v} |X_u|)$, les voisins considérés étant ceux dans l'arbre initial.

Notons alors M_v le nombre de fois que l'on applique la procédure *eclate* à v ou à l'un des sommets issu de v. Initialement, $|X_v| \leq k$, donc $M_v \leq k$. On déduit de ce qui précède que la complexité de l'algorithme 2 est bornée par :

$$\sum_{v \in V(T)} (M_v \cdot (k \cdot \sum_{u \ voisins \ de \ v} |X_u|)) \leq k^2 \cdot \sum_{v \in V(T)} (\sum_{u \ voisins \ de \ v} |X_u|) \leq 2 \cdot k^3 \cdot N.$$

■

Remarque. Fulkerson et Gross [14] ont prouvé qu'un graphe triangulé de n sommets a au plus n cliques maximales. De ce résultat découle que pour tout graphe G de n sommets, il existe une décomposition arborescente optimale (T, X) telle que $|V(T)| \leq n$. A partir d'une telle décomposition optimale, les algorithmes 1 et 2 permettent de calculer une décomposition arborescente connexe d'un graphe G de n sommets en temps $O(k^3 \cdot n)$.

Il est bien connu que déterminer une décomposition arborescente optimale d'un graphe est un problème NP-complet. Donc, déterminer une décomposition arborescente connexe optimale d'un graphe est aussi NP-complet. Nous rappelons ici un théorème démontré par Bouchitté et co [6] selon lequel il existe un algorithme polynomial pour construire une décomposition arborescente connexe d'un graphe connexe G avec un rapport d'approximation au plus $\log \mathbf{tw}(G)$. Tout d'abord, nous rappelons ce qu'est un α-séparateur des sommets d'un graphe :

Definition 6 *Soit un graphe $G = (V, E)$. Soient $0 < \alpha < 1$. Un α-séparateur-sommet de V dans G, est un ensemble $S \subset V$ tel que chaque composante connexe de $G[V \setminus S]$ contient au plus $\alpha |V|$ sommets.*

Even et co [11] ont prouvé le résultat suivant : étant donnés un graphe G et un ensemble W de sommets de G, soit τ la taille du plus petit α-séparateur de W dans G, $1/2 \leq \alpha < 1$. Soit $\alpha < \alpha' \leq 1$, alors il existe un algorithme polynomial qui calcule un α'-séparateur de taille au plus $c_{\alpha,\alpha'} \ \tau \ \log \tau$, avec $c_{\alpha,\alpha'}$ ne dépendant que de α et α'.

Ce résultat a permis a Bouchitté et co [6] d'améliorer l'algorithme de Bodlaender et co [5] qui, pour un graphe G, calcule en temps polynomial une décomposition arborescente de largeur au plus $O(\mathbf{tw}(G) \ \log |V(G)|)$. L'algorithme de Bouchitté et co [6] calcule en temps polynomial une décomposition arborescente de largeur au plus $O(\mathbf{tw}(G) \ \log \mathbf{tw}(G))$. De plus, la décomposition arborescente calculée par cet algorithme est sous-connexe. Il suffit donc d'appliquer l'algorithme 2 à la décomposition arborescente obtenue pour obtenir une décomposition arborescente connexe de largeur $\leq tw(G) \ \log tw(G)$ en temps polynomial.

Dans la section suivante, nous montrons que le théorème 1 a une conséquence importante sur l'étude de l'expansion et de l'encerclement.

2.4 Expansion connexe

Cette section montre comment utiliser le résultat du théorème 1 pour obtenir une nouvelle preuve de $\mathbf{cx}(G)/\mathbf{x}(G) \leq 1 + \log n$ pour tout graphe connexe G, originellement montré par Fomin, Fraigniaud et Thilikos [12] en utilisant $\mathbf{cx}(G) \leq \mathbf{bw}(G) \cdot \log n$ découlant de l'égalité $\mathbf{cbw}(G) = \mathbf{bw}(G)$ pour tout graphe G 2-connexe. En fait, nous allons montrer sans utiliser la décomposition en branche, que $\mathbf{cx}(G) \leq \mathbf{tw}(G) \cdot \log n$, dont il découle le résultat annoncé.

Theoreme 2 *Pour tout graphe connexe G de n sommets,* $\mathbf{cx}(G) \leq \mathbf{tw}(G) \cdot \log n$.

Preuve. On va montrer par induction sur n que, pour tout graphe connexe G de n sommets et pour tout $e \in E(G)$, on peut construire une k-extension connexe $(X_1, \cdots, X_r)$ de G, avec $k \leq \mathbf{tw}(G) \cdot \log n$ et $X_1 = \{e\}$, en ajoutant successivement toutes les arêtes de G. Pour $n \in \{1, 2\}$, le résultat est trivial. Soit $n \geq 3$ et supposons que pour tout $n' \leq n$, le résultat est acquis, c'est-à-dire qu'on peut construire une $\mathbf{ctw}(G) \cdot \log n'$-expansion connexe de G en commencant par n'importe qu'elle arête. Soit G un graphe connexe, $|V(G)| = n$, et (T, X) une décomposition arborescente connexe de G. Selon le théorème 2.5 de [20], deux cas se présentent.

1. Il existe $u \in V(T)$ tel que si l'on considère T enraciné en u et $T_1, \cdots, T_r$ les sous-arbres enracinés en $u_1, \cdots, u_r$ les fils de u, alors pour tout $i \leq r$, $|V(G[T_i])| \leq n/2$;
2. Il existe $\{u, u'\} \in E(T)$ telle que si T_0 et T'_0 sont les sous-arbres de T obtenus en supprimant $\{u, u'\}$ et enracinés respectivement en u et u', si $u_1, \cdots, u_r$ sont les fils de u dans T_0, et si $T_1, \cdots, T_r$ sont les sous arbre de T_0 enraciné en $u_1, \cdots, u_r$ et si on définit de manière analogue $u'_1, \cdots, u'_{r'}$ et $T'_1, \cdots, T'_{r'}$, alors pour tout sous-arbre $A \in \{T_1, \cdots, T_r, T'_1, \cdots, T'_{r'}\}$, $|V(G[A])| \leq n/2$.

Nous considérons séparemment ces deux cas. Les notations introduites ci-dessus sont conservées dans la suite de la preuve.

Cas 1. Soit $u \in V(T)$ tel que pour tout $1 \leq i \leq r$, $|V(G[T_i])| \leq n/2$. Par définition de la décomposition arborescente connexe, on a de plus, pour tout $1 \leq i \leq r$, $G[T_i]$ est un sous-graphe connexe de G. Soit $e \in E(G)$, et supposons sans perte de généralité qu'il existe $v \in T_1$ tel que $e \in X_v$. En appliquant l'hypothèse d'induction, soit $(X_i)_{1 \leq i \leq p}$ une k-expansion connexe de $G[T_1]$, $k \leq \mathbf{ctw}(G) \cdot \log(n/2)$ telle que $X_1 = \{e\}$. Puisque G est connexe, il existe $f \in E(G) \setminus X_p$ telle que le sous-graphe induit par $X_p \cup f$ soit connexe.

- Si $f \in X_u$, alors on pose $X_{p+1} = X_p \cup \{f\}$. Soit $x \in \delta(X_{p+1})$. Il existe une arête incidente à x dans X_{p+1} et une autre dans $E(G) \setminus X_{p+1}$. Donc $x \in X_u \bigcup (\cup_{t \in T_1} X_t)$ et $x \in \bigcup_{t \notin T_1} X_t$. Donc $x \in X_u$ et $|\delta(X_{p+1})| \leq \mathbf{ctw}(G) \leq \mathbf{ctw}(G) \cdot \log n$.
- Si $f \notin X_u$, alors il existe $i \in \{2, \cdots, r\}$ et $v' \in T_i$ tels que $f \in X_{v'}$. Sans perte de généralité, on suppose $i = 2$. Soit $(X'_i)_{1 \leq i \leq q}$ une k-expansion connexe de $G[T_2]$, $k \leq \mathbf{ctw}(G) \cdot \log(n/2)$, telle que $X'_1 = \{f\}$. On définit alors, pour tout $1 \leq j \leq q$, $X_{p+j} = X_p \cup X'_j$. Pour tout $1 \leq j \leq q$, X_{p+j} induit un sous-graphe connexe de G. Soit $x \in \delta(X_{p+j})$. On a $x \in \delta(X'_j) \cup X_u$ et $|\delta(X_{i_1+j})| \leq \mathbf{ctw}(G) + k \leq \mathbf{ctw}(G) \cdot \log n$.

On itère la construction jusqu'à inclure toutes les arêtes de G.

Cas 2 Soit $u, u' \in V(T)$ tels que, pour tout sous-arbre $A \in \{T_1, \cdots, T_r, T'_1, \cdots, T'_{r'}\}$, $|V(G[A])| \leq n/2$. Par définition de la décomposition arborescente connexe, $G[T_0]$ et $G[T'_0]$ induisent chacun un sous-graphe connexe de G. De plus, pour tout sous-arbre $A \in \{T_1, \cdots, T_r, T'_1, \cdots, T'_{r'}\}$, $G[A]$ induit un sous-graphe connexe de G. Soit $e \in E(G)$. Supposons sans perte de généralité qu'il existe $v \in T_0$ tel que $e \in X_v$. T_0 vérifie toutes les hypothèses du Cas 1. On peut donc construire une k-expansion connexe $(X_i)_{1 \leq i \leq p}$ du sous-graphe induit $G[T_0]$, $k \leq \mathbf{ctw}(G) \cdot \log n$, tel que $X_1 = e$. Comme G est connexe, il existe $f \in E(G) \setminus X_p$ telle que $X_p \cup \{e_2\}$ induise un sous-graphe connexe.

- Si $f \in X_{u'}$, alors on définit $X_{p+1} = X_p \cup \{f\}$. Soit $x \in \delta(X_{p+1})$. Il existe une arête incidente à x dans X_{p+1} et une autre dans $E(G) \setminus X_{p+1}$. Donc $x \in X_{u'} \bigcup (\cup_{t \in T_0} X_t)$ et $x \in \bigcup_{t \in T'_0} X_t$. Donc $x \in X_{u'}$ et $|\delta(X_{p+1})| \leq \mathbf{ctw}(G) \leq \mathbf{ctw}(G) \cdot \log n$.
- Si $f \notin X_{u'}$, alors il existe $1 \leq i \leq r'$ et $v \in T'_i$ tels que $f \in X_v$. Sans perte de généralité supposons que $i = 1$. Soit $(X'_i)_{1 \leq i \leq q}$ une k-expansion connexe de $G[T'_1]$, $k \leq \mathbf{ctw}(G) \cdot \log(n/2)$, telle que $X'_1 = \{f\}$. On définit alors, pour tout $1 \leq j \leq q$, $X_{p+j} = X_p \cup X'_j$. Pour tout $1 \leq j \leq q$, X_{p+j} induit un sous-graphe connexe de G. Soit $x \in \delta(X_{p+j})$, alors $x \in \delta(X'_j) \cup X_{u'}$ et $|\delta(X_{p+j})| \leq \mathbf{ctw}(G) + k \leq \mathbf{ctw}(G) \cdot \log n$.

On itère la construction jusqu'à inclure toutes les arêtes de G. ■

En utilisant le fait que $\mathbf{tw}(G) \leq \mathbf{pw}(G) \leq \mathbf{x}(G) + 1$, ainsi que les inégalités $\mathbf{x}(G) \leq \mathbf{s}(G)$ et $\mathbf{cs}(G) \leq \mathbf{cx}(G) + 1$, on en déduit le corollaire suivant :

Corollaire 1 *Pour tout graphe connexe G de n sommets,* $\mathbf{cx}(G)/\mathbf{x}(G) \leq 1 + \log n$ *et* $\mathbf{cs}(G)/\mathbf{s}(G) \leq 2 + \log n$.

2.5 Décomposition linéaire connexe

Cette section étudie la décomposition linéaire connexe pour prouver notamment que, malheureusement,le rapport largeur linéaire connexe sur largeur linéaire n'est pas borné. Bodlaender [4] a prouvé que $\mathbf{pw}(G) \leq \mathbf{tw}(G) \cdot \log(n)$ pour tout graphe G. Dans l'introduction, il est rappelé qu'il existe d'autres inégalités liant décomposition linéaire et expansion, notamment $\mathbf{pw}(G) - 1 \leq \mathbf{x}(G) \leq \mathbf{pw}(G) + 1$. Il était donc légitime de se demander si on a une relation analogue avec les grandeurs connexes. Nous montrons qu'il n'en est rien. Par ailleurs, cette section contient un algorithme qui, pour tout arbre T, renvoie une décomposition linéaire connexe optimale de T.

2.5.1 Le prix de la connexité

Soit G un graphe connexe, par composition des définitions 2 et 4, une décomposition linéaire connexe (P, X) de G est une décomposition arborescente connexe de G telle que P soit un chemin. Dans la suite, on notera $V(P) = \{1, 2, \cdots, r\}$, avec pour tout $1 \leq i \leq r-1$, $(i, i+1) \in E(P)$. On notera donc $(X_1, \cdots, X_r)$ une décomposition linéaire. On note $\mathbf{cpw}(G)$ la largeur linéaire connexe de G. La définition de la décomposition linéaire connexe est beaucoup plus contraignante que celle de l'expansion connexe dans la mesure où, dans le premier cas, pour tout $1 \leq i \leq r$, $\bigcup_{1 \leq j \leq i} X_j$ et $\bigcup_{i \leq j \leq r} X_j$ doivent chacun induire des sous-graphes connexes, alors que dans le cas de l'expansion, seul $\bigcup_{1 \leq j \leq i} X_j$ doit induire un sous-graphe connexe. L'ordre de grandeur de ces deux quantités satisfait toutefois le théorème suivant :

Theoreme 3 *Pour tout graphe connexe G,* $\mathbf{cx}(G) \leq \mathbf{cpw}(G) + 1$

Preuve. Soit un graphe G et $(X_1, \cdots, X_r)$ une décomposition linéaire connexe optimale de G. Construisons de manière recursive une ($\mathbf{cpw}$(G)+1)-expansion connexe de G.

X_1 induit un sous-graphe connexe de G. Soit $G[X_1]$ ce sous graphe. $|V(G[X_1])| \leq \mathbf{cpw}(G)+1$. Soit $p = |E(G[X_1])|$, et soit $e \in E(G[X_1])$. On pose $Y_1 = \{e\}$. Il existe alors $f \in E(G[X_1]) \setminus \{e\}$ incidente à e. On pose $Y_2 = Y_1 \cup \{f\}$. De manière itérative, on complète l'expansion avec des éléments de $E(G[X_1])$. Pour tout $\leq j \leq p$, $\delta(Y_j) \subseteq X_1$. Donc $\delta(Y_j) \leq \mathbf{cpw}(G)+1$.

Soit $1 \leq \ell < r$. On suppose que l'on a construit l'expansion de façon à couvrir $\bigcup_{j\leq \ell} X_j$. Soit i tel que l'on a construit l'expansion jusqu'à Y_i. Ainsi, $V([G(Y_i)]) = \bigcup_{j\leq\ell} X_j$. Soit $e \in E(G[X_{\ell+1}]) \setminus Y_i$ telle que $Y_i \cup \{e\}$ induit un sous graphe connexe. On pose $Y_{i+1} = Y_i \cup \{e\}$. Montrons qu'à cette étape de la construction, $\delta(Y_{i+1}) \subseteq X_{\ell+1}$. Soit $v \in \delta(Y_{i+1})$. Il existe $s \in V(G)$ tel que $\{v,s\} \in E(G) \setminus Y_{i+1}$. De plus il existe $k > \ell$ tel que $\{v,s\} \subset X_k$. Donc, si $v \in V(G[Y_i])$, alors $v \in (\cup_{j\leq\ell} X_j \bigcap X_k) \subset X_{\ell+1}$. De même, si $v \in V(G[Y_{i+1}]) \setminus V(G[Y_i])$, alors $v \in X_{\ell+1}$. Donc $|\delta(Y_{i+1})| \leq \mathbf{cpw}(G)+1$. On continue de construire l'expansion de la même façon jusqu'à avoir englobé toutes les arêtes de G. ■

Les résultats suivants montrent que les quantités connexes se comportent très différement de la version standard.

Theoreme 4 *Pour tout n, il existe un arbre T de n sommets tel que $\lceil n/3 \rceil \leq \mathbf{cpw}(T)/\mathbf{pw}(T)$. Egalement, pour tout $n > 0$, il existe un arbre T de $3n+1$ sommets tel que $\mathbf{cpw}(T)/\mathbf{ctw}(T) = n$.*

Preuve. du théorème 4 Pour montrer la première partie du théorème 4, on considère le graphe S_n qui est une étoile à trois branches, chacune possédant n sommets. Plus précisemment,

- $|V(S_n)| = 3n+1$, $V(S_n) = \{a_1, \cdots, a_n, b_1, \cdots, b_n, c_1, \cdots, c_n, r\}$;
- $E(S_n) = \bigcup_{i\in\{1,\cdots,n-1\}} \{(a_i, a_{i+1}), (b_i, b_{i+1}), (c_i, c_{i+1})\} \cup \{(a_n, r), (b_n, r), (c_n, r)\}$.

On peut montrer facilement que $\mathbf{pw}(S_n) = 2$. Nous allons montrer que $\mathbf{cpw}(S_n) = n$.

Soit C le chemin tel que $V(C) = \{x, y, z\}$ et $E(C) = \{\{x, y\}, \{y, z\}\}$. Soit les ensembles $X_x = \{a_1, \cdots, a_n, r\}$, $X_y = \{b_1, \cdots, b_n, r\}$, $X_z = \{c_1, \cdots, c_n, r\}$. Alors (C, X) est une décomposition linéaire connexe de S_n de largeur n. Donc $\mathbf{cpw}(S_n) \leq n$.

Supposons qu'il existe une décomposition linéaire connexe (C, X) de S_n de largeur $k < n$. Soit $r = |V(C)|$, on notera C comme le chemin $1, 2, \cdots, r$. Soit m (respectivement k, ℓ) le plus petit indice tel que $a_1 \in X_m$ (respectivement b_1, c_1). Supposons, sans perte de généralité, que $m \leq k \leq \ell$. Soit $X = \bigcup_{i\leq k}(X_i)$, X induit un sous graphe connexe de S_n contenant a_1 et b_1. Donc X contient $b_1, \cdots, b_n, r$. De meme, soit $Y = \bigcup_{k\leq i}(X_i)$. Y induit un sous graphe connexe de S_n contenant b_1 et c_1. Donc Y contient $b_1, \cdots, b_n, r$. Donc $\{b_1, \cdots, b_n, r\} \subset X \cap Y \subset X_k$. Or $n+1 \leq |X_{t_k}|$ d'ou la contradiction. Donc $n \leq \mathbf{cpw}(S_n)$.

Montrons la seconde partie du théorème. Soit $n > 0$, on considère S_n le graphe étoile défini précédemment. On a montré que $cpw(S_n) = n$, de plus S_n est un arbre, donc on a $\mathbf{tw}(S_n) = \mathbf{ctw}(S_n) = 1$. D'ou $\mathbf{cpw}(S_n)/\mathbf{ctw}(S_n) = n$. ■

En conséquence de quoi, comme $\mathbf{cx}(S_n) = \mathbf{pw}(S_n) = 2$, on obtient le corollaire suivant :

Corollaire 2 *Pour tout n, il existe un arbre T de n sommets tel que $\lceil n/3 \rceil \leq \mathbf{cpw}(T)/\mathbf{cx}(T)$.*

2.5.2 Complexité de la largeur linéaire connexe

On sait que déterminer si $\mathbf{pw}(G) \leq k$ est un problème NP-complet [23]. Dans cette section, on prouve qu'il en est de même pour la largeur linéaire connexe.

Theoreme 5 *Le problème de savoir si $\mathbf{cpw}(G) \leq k$ est NP-difficile.*

Preuve. Soit un graphe G et soit $(X_0, \cdots, X_r)$ une décomposition linéaire optimale de G. On construit le graphe connexe H en ajoutant un sommet x à G de la manière suivante : $V(H) = V(G) \cup \{x\}$ et $E(H) = E(G) \bigcup \cup_{y \in V(G)} \{x, y\}$. Alors, $(X_0 \cup \{x\}, \cdots, X_r \cup \{x\})$ est une décomposition linéaire connexe de H. Donc $\mathbf{cpw}(H) \leq \mathbf{pw}(G) + 1$.

Soit $(Y_0, \cdots, Y_r)$ une décomposition linéaire connexe optimale de H. Alors, $(Y_0 \setminus \{x\}, \cdots, Y_r \setminus \{x\})$ est une décomposition linéaire de G. Donc $\mathbf{pw}(G) \leq \mathbf{cpw}(H) - 1$.

Donc $\mathbf{cpw}(H) = \mathbf{pw}(G) + 1$ et déterminer si $\mathbf{cpw}(H) \leq k$ revient à déterminer si $\mathbf{pw}(G) \leq k - 1$, ce qui est NP-complet. ■

2.5.3 Décomposition linéaire connexe des arbres

Nous présentons un algorithme qui, pour tout arbre, construit une décomposition linéaire connexe optimale de cet arbre. On montre au passage que la largeur linéaire connexe d'un arbre est au plus $\lfloor D/2 \rfloor$ où D est le diamètre de l'arbre.

Theoreme 6 *Il existe un algorithme en temps $O(n^3)$ qui, pour tout arbre T de n sommets, retourne une décomposition linéaire connexe optimale de T.*

Preuve. L'algorihme 3 recherche tout d'abord deux sommets x et y à distance égale au diamètre de T. Le chemin de x à y est appelé le *tronc* de l'arbre. L'algorithme construit une décomposition linéaire connexe en ajoutant les différents sommets du *tronc* au fur et à mesure et en ajoutant les branches qui s'y raccordent.

On montre facilement que $\bigcup_{j \in C}(X_j) = V(T)$ et que pour tout $e \in E(T)$, il existe u tel que $e \in X_u$. De même, par construction, pour tout $t \in T$, $\{\text{u} : t \in X_u\}$ est un sous graphe connexe de C. Donc l'algorithme retourne bien une décomposition linéaire.

Soit $\ell \leq |V(C)|$, soit i tel que X_ℓ est créé à l'étape i. Soit T_1 et T_2 les arbres obtenus en supprimant les arêtes $(z_{i-1}, z_i) et (z_{i+1}, z_i)$ de T et tels que $x \in T_1$ et $y \in T_2$. Alors, $X = \bigcup_{j \leq \ell} X_j = (T_1 \cup z_i \cup C_1)$ et $Y = \bigcup_{\ell \leq j} X_j = (T_2 \cup z_i \cup C_2)$ avec C_1 et C_2 des ensembles de chemins dont l'une des extrémités est z_i. Donc X et Y induisent des sous-graphes connexes de T et la décomposition linéaire est connexe.

Soit ℓ tel que $|X_\ell| = r$ maximal et $k = r - 1$ la largeur de la décomposition linéaire obtenue. Si $r = 2$, la décomposition est évidemment optimale. Supposons que $r > 2$. Soit $1 \leq i \leq D - 1$ et $v \in V(T)$ tel que $X_\ell = \{\text{chemin de } z_i \text{ a } v\}$. Dans la suite, on supposera sans perte de généralité que $\text{dist}(x, z_i) \leq \text{dist}(z_i, y)$ et donc $\lceil D/2 \rceil \leq \text{dist}(x, z_i)$. Remarquons tout d'abord que $\text{dist}(v, x) \leq \min(\text{dist}(z_i, x), \text{dist}(z_i, y))$. Donc $k \leq \min(\text{dist}(z_i, x), \text{dist}(z_i, y))$. On note $T' = \{t/t \in \text{chemin de } x \text{ à } z_i\}$ et $T'' = \{t/t \in \text{chemin de } y \text{ à } z_i\}$. Comme on a supposé que $\lceil D/2 \rceil \leq \text{dist}(x, z_i)$, alors $|T'| \geq \lceil D/2 \rceil > k$. De plus $\text{dist}(x, v) = \text{dist}(x, z_i) + k \leq D = \text{dist}(x, y) = \text{dist}(x, z_i) + |T''|$, donc $|T''| \geq k$.

Soit $X_1, \cdots, X_m$, une décomposition linéaire connexe optimale de T. Soit a (respectivement b, c, d) le plus petit indice $j \leq m$ tel que v (respectivement z_i,x,y) $\in X_j$. On suppose sans perte de généralité que $a \leq b$. Dans un premier temps, on montre que $b \leq \min(c, d)$. Supposons que $c < b$, montrons que l'on aboutit à une contradiction. $\bigcup_{j \leq b} X_j$ induit un sous-graphe connexe de T qui contient v et x, donc qui contient z_i. Donc il existe $j \leq c < b$ tel que $z_i \in X_j$, ce qui contredit le fait que b est minimal. Donc $c \geq b$. En donnant à y (resp. à d) le rôle de x (resp. de c), On en déduit de même que $d \geq b$.

Nous allons montrer en considérant deux cas selon l'ordre des indices c et d que $\mathbf{cpw}(T) \geq k$.

Cas 1 : $a \leq b \leq c \leq d$. $\bigcup_{j \leq c} X_j$ induit un sous-graphe connexe de T qui contient v et x, donc qui contient T'. $\bigcup_{j \geq c} X_j$ induit un sous-graphe connexe de T qui contient y et x, donc qui contient T'. Donc $T' \subset X_c$. On en déduit $\mathbf{cpw}(T) \geq |X_c| - 1 \geq |T'| \geq k$.

Algorithm 3 Décomposition linéaire connexe optimale d'un arbre

Entrées: T un arbre

Sorties: (C, X) une décomposition linéaire connexe optimale de T

Début

Rechercher $x, y \in V(T)$ les deux sommets les plus éloignés dans T ;

/* $\text{dist}(x, y) = D$ est le diametre de T */

$C' \leftarrow$ le chemin $\{z_1, \cdots, z_{D+1}\}$ de x à y ;

/* $x = z_1$ et $y = z_{D+1}$ */

$V(C) \leftarrow \{1\}$;

$E(C) \leftarrow \emptyset$;

$X_1 \leftarrow \{x, z_1\}$;

$u \leftarrow 2$;

$i \leftarrow 2$;

tantque $i \leq D$ **faire**

  si $3 \leq \deg(z_i)$ **alors**

    $V \leftarrow \{v \mid \{v, z_i\} \in E(T)$ et $v \neq z_j$ pour tout $j \leq D + 1\}$;

    pour $v \in V$ **faire**

      $T' \leftarrow$ le sous-arbre de T enraciné en v ;

      pour chaque feuille f de T' dans un ordre DFS **faire**

        $V(C) \leftarrow V(C) \bigcup u$;

        $E(C) \leftarrow E(C) \bigcup (u - 1, u)$;

        $X_u \leftarrow \{z_i\} \bigcup \{$chemin de v à $f\}$;

        $u \leftarrow u + 1$;

      fin pour

    fin pour

  finsi

  $V(C) \leftarrow V(C) \bigcup \{u\}$;

  $E(C) \leftarrow E(C) \bigcup \{(u - 1, u)\}$;

  $X_u \leftarrow \{z_i, z_{i+1}\}$;

  $u \leftarrow u + 1$;

  $i \leftarrow i + 1$;

fin tantque

Retourner $(C, (X_j)_{j \in C})$

Fin

Cas 2 : $a \leq b \leq d \leq c$. $\bigcup_{j \leq d} X_j$ induit un sous-graphe connexe de T qui contient v et y, donc qui contient T''. $\bigcup_{j \geq d} X_j$ induit un sous-graphe connexe de T qui contient y et x, donc qui contient T''. Donc $T'' \subset X_d$. On en déduit $\mathbf{cpw}(T) \geq |X_d| - 1 \geq |T''| \geq k$.

Donc l'algorithme renvoie une décomposition arborescente connexe optimale de T.

Nous étudions maintenant la complexité de l'algorithme. L'algorithme 3 détermine tout d'abord le *tronc* de l'arbre, ce qui est une opération linéaire. Pour s'en convaincre, considérons un arbre T enraciné en v, posons $T_1, \ldots, T_r$ les sous arbres enracinés en les fils de v, $d_1, \ldots, d_r$ les diamètres de ces arbres et $h_1, \ldots, h_r$ leur hauteur. Soit $1 \leq i, j \leq r$ tel que $h_i = \max_{1 \leq k \leq r} h_k$ et $h_j = \max_{1 \leq k \leq r, k \neq i} h_k$. Alors le diamètre D de T est : $D = \max(d_1, \ldots, d_r, h_i + h_j + 2)$. On peut déduire de cette formule un algorithme linéaire simple calculant le tronc de l'arbre.

Une fois le tronc de l'arbre déterminé, l'algorithme réalise un simple parcours en profondeur d'abord (Depth First Search) ce qui est bien connu pour être linéaire. Une implémentation adéquate de cet algorithme serait de coder pour chaque sommet v de l'arbre, l'intervalle $[i, j]_v$ tel que $v \in X_i, \ldots, X_j$.

■

On déduit de la preuve du théorème 6 le corollaire suivant :

Corollaire 3 *Soit T un arbre de diametre D, alors* $\mathbf{cpw}(T) \leq \lfloor D/2 \rfloor$.

Cette borne est atteinte par exemple par une étoile dont les branches ont même longueur.

3 Expansion q-arête-connexe

Cette section présente une extension des notions d'expansion connexe et de décomposition arborescente connexe à la q-arête-connexité. Dans la suite, lorsque l'on parle de connexité sans préciser, il s'agit d'*arête*-connexité. La section 3.1 généralise le résultat selon lequel pour tout arbre T, $\mathbf{cx}(T) = \mathbf{mcx}(T)$. Dans la section 3.2, on étudie le comportement des grandeurs q-connexes par rapport aux grandeurs standards.

3.1 Quelques expansions q-connexes monotones

On rappelle qu'une expansion $X_1, \cdots, X_r$ est monotone si pour tout $1 \leq i < r, X_i \subset X_{i+1}$. Rappelons également qu'un graphe H est un mineur d'un graphe G, noté $H \preceq G$ si H peut être obtenu à partir de G par une succession de contractions d'arêtes, de suppressions d'arêtes et de suppression de sommets. On dit que G est H-libre si H n'est pas un mineur de G. Etant donnée une propriété P, on dit que H est une obstruction par rapport à P si pour tout graphe G tel que $H \preceq G$, G ne satisfait pas P. Cette notion est très importante dans la théorie élaborée par Robertson et Seymour [10]. En particulier, leur fameux théorème dit que pour toute propriété, la classe des graphes vérifiant cette propriété admet un nombre d'obstructions fini. Pour $k \geq 1$, on note B_k le graphe constitué de deux sommets reliés par k arêtes.

Dans cette partie, on veut étendre la notion de d'expansion connexe à la q-connexité. Plus précisement, on voudrait étendre le résultat selon lequel pour tout arbre T, $\mathbf{cx}(T) = \mathbf{mcx}(T)$. Nous allons définir la k-expansion q-connexe et montrer que pour la classe des graphes q-connexe, B_{q+1}-libre, l'existence d'une k-expansion q-connexe implique celle d'une k-expansion q-connexe monotone.

Les lemmes 6 et 7 prouvent tout d'abord deux propriétés de la classe des graphes q-connexe, B_{q+1}-libre. On en déduit dans le lemme 8 une caractérisation de cette classe. Cette caractérisation nous permet de prouver le théorème 7.

Tout d'abord, nous prouvons un lemme qui nous sera utile dans la suite.

Lemme 5 *Soit G un graphe pour lequel il existe $x, y \in V(G)$ avec k chemins arêtes-disjointes entre x et y. Alors, $B_k \preceq G$.*

Preuve. Soit G un graphe, $\{x, y\} \in V(G)$ et $C_1, \cdots, C_k$, k chemins arêtes-disjointes de x à y.

Si les chemins sont noeuds-disjoints, $G \succeq B_k$ est évident. il suffit de contracter chacun des chemins pour ne plus laisser que x et y liés par k arêtes.

Soit $X = \{z \in V(G) : \text{il existe } (i,j) i \neq j \text{ et } z \in (C_i \cap C_j)\}$ et supposons $X \neq \emptyset$. On raisonne par induction sur $h = |X|$. Si $h = 1$, $X = \{z\}$, pour tout i tel que $z \in C_i$, on contracte x en z le long de C_i. Le graphe obtenu est tel que il existe k chemins noeuds-disjoints de x a y, on est ramené au premier cas. Soit $h \geq 1$ et supposons pour tout $h' \leq h-1$, si $|X| = h'$, alors le lemme est verifié. Soit $z \in X$ tel que $\text{dist}(x, z)$ soit minimale, alors pour tout i tel que $z \in C_i$, on contracte x en z le long de C_i. Le graphe obtenu est tel que il existe k chemins arêtes-disjointes de x a y et ces chemins ne se rencontrent qu'en au plus $h-1$ noeuds, on peut alors appliquer l'hypothèse d'induction et $G \succeq B_k$. ■

A present, on prouve une première propriété de la classe des graphes q-connexe, B_{q+1}-libre.

Lemme 6 *Soit G un graphe sans boucle, q-arête-connexe et B_{q+1}-libre. Alors pour tout $x \in V(G)$, $\deg(x)$ est un multiple de q.*

Preuve. Par l'absurde, supposons qu'il existe $x \in V(G)$, $1 \leq k$ et $1 \leq k' \leq q-1$ tel que $\deg(x) = k * q + k'$. Dans notre démonstration, nous allons supposer de plus que $k = 1$. Soit G un graphe sans boucle, q-connexe et B_{q+1}-libre, et soient $x \in V(G)$ et $1 \leq k \leq q-1$ tels que $\deg(x) = q + k$. Soit y un voisin de x, d'apres le théorème de Menger [10], il existe $(C_1, \cdots, C_q)$ q chemins arêtes-disjointes de x à y. Comme $\deg(x) > q$, il existe une arête (x,z) incidente à x qui n'appartient à aucun de ces chemins. Si $y = z$, on a trouvé $q+1$ chemins arêtes-disjointes reliant x à y, d'après le lemme 5, $B_{q+1} \preceq G$. Supposons donc $z \neq y$, alors il existe $(C'_1, \cdots, C'_q)$ q chemins arêtes-disjointes de z à y. Si pour tout $1 \leq i \leq q$, $x \in C'_i$, on voit facilement que $\deg(x) \geq 2 * q$, donc il existe i tel que $(C'_i \cap x) = \emptyset$. Il suffit de contracter y en z le long de C'_i et le graphe obtenu est tel que il y a $q+1$ chemins arêtes-disjointes de x a y et $B_{q+1} \preceq G$.

La preuve générale est semblable au cas que nous avons traité, nous ne donnons que son principe, sans entrer dans les détails. Supposons que $\deg(x) = k * q + k'$ avec $k \geq 2$. Dans ce cas, il se peut que $x \in C'_i$ pour tout $i \leq q$. Alors, il existe q chemins de x à y et q chemins de x à z tels que ces $2q$ chemins soient arêtes-disjointes. Comme $\deg(x) > 2q$, il existe v un voisin de x qui n'appartient à aucun de ces chemins. On itère la démonstration avec v à la place de z. Il y a au plus k itérations. ■

Le lemme suivant prouve une autre propriété de la classe des graphes q-connexe, B_{q+1}-libre.

Lemme 7 *Soit G un graphe sans boucle, q-arête-connexe et B_{q+1}-libre. Soit $x \in V(G)$ et y un voisin de x. Soit k le nombre d'arêtes liant x à y. Alors,*

- *$k = q$, si q est impair ;*
- *$k \in \{q/2, q\}$ si q est pair.*

Preuve. Soit G un graphe sans boucle, q-arête-connexe et B_{q+1}-libre, soit $x \in V(G)$, et soient $(y_1, \cdots, y_r)$ les voisins de x et $(k_1, \cdots, k_r)$ tels que pour tout $i \leq r$, k_i est le nombre d'arêtes entre x et y_i. Enfin, soit $k = min_{i \leq r}(k_i)$, soit j tel que $k = k_j$ et on pose $y = y_j$.

De manière évidente, pour tout $1 \leq i \leq r$, $k_i \leq q$, sinon B_{q+1} est un sous-graphe de G et $B_{q+1} \preceq G$. Donc $k \leq q$.

Supposons $\lfloor q/2 \rfloor + 1 \leq k \leq q-1$, il existe q chemins arêtes-disjointes entre x et y (théorème de Menger) donc il existe C un chemin de x à y ne passant par aucune arête (x,y). Soit $v \neq y$ le voisin de x appartenant à C. Il existe $k' \geq k \geq \lfloor q/2 \rfloor + 1$ arêtes (x,v) et donc en contractant v en y le long de C, on obtient un graphe dans lequel il y a $k + k' \geq q+1$ arêtes (x,y) et donc $B_{q+1} \preceq G$. Donc si $k < q$, alors $k \leq \lfloor q/2 \rfloor$.

Supposons que $k < \lfloor q/2 \rfloor$. En plus des k arêtes entre x et y, le théorème de Menger assure l'existence de $q-k$ chemins $C_1, \cdots, C_{q-k}$ arêtes-disjointes (et disjoints des arêtes (x,y)) reliant x à y. Soit $z \in V(G)$ un voisin de x tel qu'il existe $1 \leq i \leq q-k, z \in C_i$. Soit $J = \{1 \leq j \leq q-k$ tel que $z \in C_j\}$, et soit k' le nombre d'arêtes (x,z).

Si $|J| < k'$, en contractant y en z le long de tous les C_j, on obtient facilement que $B_{q+1} \preceq G$ (les q chemins qu'il y avait avant la contraction plus l'arête (x,z) supplementaire). Donc $|J| = k'$.

Remarquons tout d'abord que si $q = k + k'$, alors $k' \geq \lceil q/2 \rceil$ or en contractant les k arêtes (x,y), on obtient $2k' > q$ chemins arêtes-disjointes entre x et z, d'apres le lemme 5, cela implique que $B_{q+1} \preceq G$. Donc $q > k + k'$ et $k' \leq \lfloor q/2 \rfloor$.

Donc, il existe $q - k - k'$ chemins de x à z, disjoints des $k + k'$ arêtes (x,y) et (x,z). Soit C_0 un de ces chemins, si C_0 est arêtes-disjointes de $C_1, \cdots, C_{q-k}$, alors en contractant y en z le long d'un chemin $(C_j)_{j \in J}$, on obtient $q+1$ chemins arêtes-disjointes de x à y et $B_{q+1} \preceq G$. Sinon, on considère $C_i \in \{C_1, \cdots, C_{q-k}\}$ tel que C_i et C_0 ont une arête commune (u,v) telle que v soit à distance minimale de z. On remarque que $v \neq x$ et $v \neq y$, donc on peut contracter x en v le long de C_i et y en z le long d'un chemin $(C_j)_{j \in J}$, on obtient $q+1$ chemins arêtes-disjointes de x à y et $B_{q+1} \preceq G$.

Montrons enfin que si q est impair, $k = \lfloor q/2 \rfloor$ aboutit à une contradiction . Supposons que q est impair et qu'il existe $\{x,y\} \in V(G)$ tel que il existe $\lfloor q/2 \rfloor$ arêtes (x,y). D'après le théorème de Menger, il existe au moins $C_1, C_2, \cdots, C_{\lceil q/2 \rceil}$ chemins arêtes-disjointes et disjoints des arêtes (x,y) reliant x à y. On pose $x_1, x_2, \cdots, x_{\lceil q/2 \rceil}$ les voisins de x appartenant respectivement aux chemins $C_i, i = 1, \cdots, \lceil q/2 \rceil$. Si il existe $1 \leq i \leq \lceil q/2 \rceil$ tel qu'il y a q arêtes (x, x_i), il suffit de contracter y le long de C_i et on obtient un graphe tel qu'il y a $q + \lfloor q/2 \rfloor$ arêtes (x,y) et donc $B_{q+1} \preceq G$. Sinon, comme $\deg(x) \geq q$, il existe $i, j \in \{1, \cdots, \lceil q/2 \rceil\}$ tels que il y a $\lfloor q/2 \rfloor$ arêtes (x, x_i) et $\lfloor q/2 \rfloor$ arêtes (x, x_j). Alors, en contractant y en x_i le long de C_i puis en x_j le long de C_j, on obtient un graphe tel qu'il y a $3 * \lfloor q/2 \rfloor$ arêtes (x,y) et donc $B_{q+1} \preceq G$.

Donc si q impair $k = q$, sinon $k = q$ ou $k = q/2$. ■

On definit la classe C_q des graphes obtenus de la manière récursive suivante :

- $B_q \in C_q$
- si q pair, $D_q \subset C_q$ où D_q est la classe des graphes obtenus à partir d'un cycle de cardinalité ≥ 3 pour lequel on multiplie le nombre d'arêtes reliant 2 sommets par $q/2$.
- tout graphe obtenu en identifiant deux sommets de deux graphes de C_q est un graphe de C_q.

On peut remarquer que la classe C_1 est exactement la classe des arbres. La figure 4 représente un graphe de C_2.

Le lemme suivant utilise les deux propriétés vues ci-dessus pour caractériser la classe des graphes q-connexe, B_{q+1}-libre.

Lemme 8 *C_q est la classe des graphes q-connexes, sans boucle et B_{q+1} libres.*

Preuve. Posons C la classe des graphes q-connexes, sans boucles et B_{q+1} libres. Il est facile de montrer que $B_q \in C$ et $D_q \subset C$. De manière inductive, supposons G_1 et $G_2 \in (C \cap C_q)$, construisons G en liant G_1 et G_2 par le sommet x. De manière évidente, G est q-connexe et sans

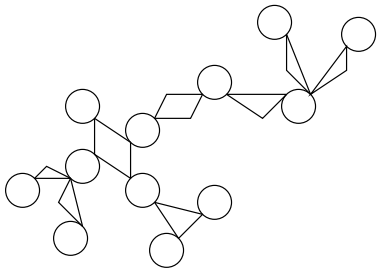

FIG. 4 – Exemple d'un graphe de C_2

boucle. Supposons qu'il n'est pas B_{q+1} libres, soient y et z deux sommets tels qu'en effectuant les opérations de contractions de la théorie des mineurs, on obtiennent le graphe B_{q+1} dont les deux sommets sont y et z : lors de la contraction, on peut supposer sans perte de généralité que $y \in G_1$ se réduit en x, ce qui signifie qu'à partir de x et de z dans G_2 on obtient B_{q+1}, ce qui contredit le fait que G_2 est B_{q+1} libre. Donc $C_q \subset C$.

Soit $G \in C$. Supposons qu'il existe $x, y \in V(G)$ tels qu'il y a q arêtes (x, y). Alors, on raisonne par induction sur $|V(G)|$. Si $|V(G)| = 2$, $G \in C_q$. Soit n, on suppose que pour tout $k \leq n-1$, si $|V(G)| = k$ et $G \in C$, alors $G \in C_q$. Soit $G \in C$, $|V(G)| = n$. Soit $x, y \in V(G)$ tels qu'il y a q arêtes (x, y), alors, G privé de ces q arêtes a deux composantes non reliées G_1 et G_2 chacunes q-connexes et B_{q+1} libre. On leur applique l'hypothèse d'induction et donc $G \in C_q$.

Supposons que q est pair et que pour tout $(x, y) \in E(G)$, il y a $q/2$ arêtes (x, y). On definit alors C'_q la sous-classe de C_q, des graphes obtenus de la manière récursive suivante :

- $D_q \subset C'_q$.
- tout graphe obtenu en identifiant deux sommets de deux graphes de C'_q est un graphe de C'_q.

Montrons que $G \in C'_q$. On montre que pour tout $x \in V(G)$, il existe $G' \in D_q$ tel que G' sous-graphe de G contenant x et que si il existe G_1 et $G_2 \in D_q$ des sous-graphe de G contenant x, alors $(V(G_1) \cap V(G_2)) = \{x\}$ et le sous-graphe induit par $V(G) \setminus \{x\}$ n'est pas connexe.

Soit $x \in V(G)$, soit y tel que $(x, y) \in E(G)$, il existe $r \geq 2$ et $(v_0, v_1, \cdots, v_r)$ un chemin de $x = v_0$ à $y = v_r$ disjoint des $q/2$ arêtes (x, y). Soit G' le sous graphe de G induit par $v_0, v_1, \cdots, v_r$, il est facile de montrer que G' est le cycle $(v_0, v_1, \cdots, v_r)$ tel qu'on a multiplié les arêtes par $q/2$ et donc $G' \in D_q$: en effet si il existait une arête (v_i, v_j) avec $j \neq i+1$ modulo r, on aurait $B_{q+1} \preceq G' \preceq G$. Supposons il existe G_1 et $G_2 \in D_q$ des sous-graphe de G ayant au moins deux sommets communs : on pose $G_1 = v_1, \cdots, v_r$ (on entend par là le cycle $v_1, \cdots, v_r$ augmenté avec $q/2$ arêtes) et $G_2 = u_1, \cdots, u_l$ et sans perte de généralité $u_1 = v_1$ et $u_r = v_l$. Il est immédiat que l'on peut se ramener à un mineur tel qu'il existe $3 * q/2 \geq q+1$ chemins arêtes-disjointes entre u_1 et u_r et donc $B_{q+1} \preceq G$. On obtient le même résultat si on suppose que le sous-graphe induit par $V(G) - x$ est connexe. Donc $C \subset C_q$

De ce qui précède, on déduit que $G \in C'_q$: soit $x \in V(G)$, soit $G_1 \in D_q$ sous-graphe de G contenant x. Si $G = G_1$, la preuve est faite. Sinon, il existe $z \in V(G_1)$ tel qu'il existe $y \in V(G) \setminus V(G_1)$ tel que $(z, y) \in E(G)$, soit $G_2 \in D_q$ sous-graphe de G contenant z et y, d'après ce qui précède, $(V(G_1) \cap V(G_2)) = \{z\}$ donc $V(G_1) \cup V(G_2) \in C'_q$, on continue cette construction récursivement jusqu'à $V(G) = \bigcup V(G_i)$ et par conséquent $G \in C'_q \subset C_q$. ■

On définit maintenant la notion de k-expansion q-connexe. Soit un graphe q-connexe G, la famille $(X_0, \cdots, X_r)$ est une k-expansion q-connexe de G si elle satisfait :

- $X_i \subset E(G)$ pour tout $0 \leq i \leq r$;
- $X_0 = \emptyset$ et $X_r = E(G)$;
- pour tout $0 \leq i \leq r$, $|\delta(X_i)| \leq k$;
- pour tout $0 \leq i \leq r$, le sous graphe $G[X_i]$ de G induit par X_i est q-connexe ;
- pour tout $0 \leq i \leq r$, $X_{i+1} \setminus X_i$ est minimal au sens de l'inclusion, i.e. il n'existe pas $Y \subset X_{i+1} \setminus X_i$ tel que $X_i \cup Y$ q-connexe ;
- pour tout $0 \leq i \leq r$, il existe une k-expansion $(q-1)$-connexe de X_i à X_{i+1} dans G.

On note $\mathbf{cx}_q(G)$ le plus petit k tel qu'il existe une k-expansion q-connexe de G. L'expansion est de plus dite monotone si, pour tout $0 \leq i < r$,

- $X_i \subset X_{i+1}$;
- l'expansion $q-1$-connexe de X_i dans X_{i+1} est monotone.

On note $\mathbf{mcx}_q(G)$ le plus petit k tel qu'il existe une k-expansion q-connexe monotone dans G. On a evidement, $\mathbf{cx}_q(G) \leq \mathbf{mcx}_q(G)$.

Theoreme 7 *Soit G un graphe sans boucle, q-arête-connexe et B_{q+1}-libre. S'il existe une k-expansion q-connexe dans G, alors il existe une k-expansion q-connexe monotone dans G. En conséquence, $\mathbf{cx}_q(G) = \mathbf{mcx}_q(G)$.*

Corollaire 4 *Si $q = 1$, comme C_1 est la classe des arbres, on retrouve que pour tout arbre T, $\mathbf{cx}(T) = \mathbf{mcx}(T)$.*

Preuve. du théorème 7 Soit G un graphe sans boucle, q-arête-connexe et B_{q+1}-libre, et soit $(X_0, \cdots, X_r)$ une k-expansion q-connexe pour G, vérifiant les deux conditions suivantes :

- $\sum_{1 \leq i \leq r}(|\delta(X_i)| + 1)$ minimum (C_1)
- $\sum_{1 \leq i \leq r} |X_i|$ minimum (C_2)

Montrons qu'une telle k-expansion q-connexe de G est monotone.

Supposons qu'il existe $j \leq r-1$ tel que $X_{j+1} \subset X_j$, alors $(X_0, \cdots, X_{j-1}, X_{j+1}, \cdots, X_r)$ met (C_2) en défaut. Donc pour tout $j \leq r-1$, $|X_{j+1} - X_j| \neq 0$.

Pour tout $1 \leq j \leq r$, $|\delta(X_{j-1} \cup X_j)| \geq |\delta(X_j)|$ sinon, $(X_0, \cdots, X_{j-1}, X_{j-1} \cup X_j, \cdots, X_r)$ met (C_1) en défaut. On en déduit que $|\delta(X_{j-1} \cap X_j)| \leq |\delta(X_{j-1})|$ (puisque $|\delta(X_{j-1} \cup X_j)| + |\delta(X_{j-1} \cap X_j)| \leq |\delta(X_j)| + |\delta(X_{j-1})|$).

Montrons maintenant que $X_{j-1} \cap X_j$ est q-connexe pour tout $1 \leq j \leq r$. $G[X_{j-1}]$ et $G[X_j]$ sont des sous-graphes q-connexes de G, d'après le théorème précédent, on montre qu'ils appartiennent à la classe C_q définie plus haut, de même que leur intersection. Donc $X_{j-1} \cap X_j$ est q-connexe.

Supposons qu'il existe $j \leq r$, tel que $|X_{j-1} \cap X_j| < |X_{j-1}|$, alors $(X_1, \cdots, X_{j-2}, X_{j-1} \cap X_j, X_j, \cdots, X_r)$ est une k-expansion q-connexe pour G, ce qui contredit (C_2). Donc pour tout $j \leq r$, $|X_{j-1} \cap X_j| \geq |X_{j-1}|$, et $X_{j-1} \subset X_j$, cette expansion est donc monotone.

D'apres ce qui précède et le théorème précédent, on montre facilement que pour tout $j \leq r$, $X_{j+1} \setminus X_j \in (D_q \cup B_q)$ et donc la k-expansion $q-1$-connexe de X_j dans X_{j+1} est trivialement monotone. ■

3.2 Décomposition arborescente q-connexe

Dans cette partie, on étudie la décomposition arborescente q-connexe. Plus précisement, on montre que si on définit la décomposition arborescente q-connexe de manière analogue au cas $q = 1$ de la définition 4, la largeur d'arborescence q-connexe ne suit pas le comportement de la largeur d'arborescence. dans un deuxième temps, on étudie le lien entre expansion q-connexe et largeur linéaire q-connexe.

On dit qu'une e-coupe d'une décomposition arborescente est *q-connexe* si les sous-graphes résultant de la coupe sont chacun q-connexes. On appelle décomposition arborescente connexe d'un graphe q-connexe G, (T, V) une décomposition arborescente telle que : pour toute arête e de T, la coupe est q-connexe. La largeur d'arborescence q-connexe de G est la largeur minimale sur toute décomposition arborescente q-connexes de G. Elle est notée $\mathbf{ctw}_q(G)$.

Nous avons tout d'abord montré que la largeur linéaire q-connexe reste, comme pour $q = 0$ et $q = 1$, une borne supérieure pour l'expansion q-connexe.

Theoreme 8 *Pour tout graphe connexe G, $\mathbf{cx}_q(G) \leq \mathbf{cpw}_q(G) + 1$;*

Preuve. La preuve est la même que dans le cas $q = 1$ détaillé dans le théorème 2. ■

Nous montrons à présent qu'imposer la q-connexité lors de la stratégie d'encerclement est une contrainte très forte :

Theoreme 9 *Pour tout $q > 1$, pour tout n_0, il existe $n \geq n_0$ et un graphe q-connexe de n sommets tel que $\mathbf{ctw}_q(G) = n - 1$ et $\mathbf{tw}(G) \leq q$.*

Preuve.

Soit $q > 1$, et soit $n_0 > 0$. Si $n_0 \leq q + 1$, le résultat est trivial. On suppose donc que $n_0 > q + 1$. Si $q = 2$, un anneau comportant n_0 sommets convient. Supposons à présent que $q \geq 3$. Nous allons construire un graphe q-connexe G tel que $\mathbf{tw}(G) = q$ et $\mathbf{ctw}_q(G) = |V(G)| - 1$. Soit $k = \lceil (n-2)/(q-1) \rceil$. On considère k répliques du graphe complet de $q + 1$ sommets K_{q+1}. On note ces graphes $Q_0, \cdots, Q_{k-1}$. Et pour tout $j \in \{1, \cdots, k-2\}$, Q_j a deux sommets adjacents, notés a_j et b_j, en commun avec Q_{j-1} et deux sommets adjacents , notés a_{j+1} et b_{j+1},en commun avec Q_{j+1}. Pour tout $j \in \{1, \cdots, k-2\}$, les quatres sommets ainsi définis sont distincts. Pour terminer la construction du graphe G, on supprime les arêtes $\{a_j, b_j\}$ pour tout $j \in \{1, \cdots, k-1\}$. La figure 5 représente le graphe G dans le cas $q = 5$ et $n_0 = 23$.

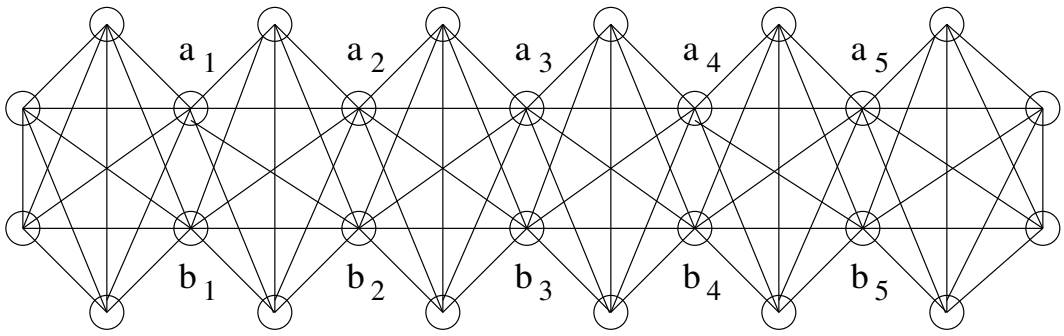

FIG. 5 – Exemple d'un graphe G tel que $tw(G) \leq 5$ et $ctw_5(G) = 25$

$(V(Q_0), \cdots, V(Q_{k-1}))$ est une décomposition linéaire de G, donc $\mathbf{tw}(G) \leq q$. Soit $(X_0, \cdots, X_R)$ une décomposition linéaire q-connexe de G. Par définition, X_0 induit un sous-graphe q-connexe de G. Montrons que le seul sous-graphe de G qui soit q-connexe est G lui-même, et donc que $X_0 = V(G)$. On en déduit alors que $\mathbf{cpw}_q(G) = |V(G)| - 1 \geq n_0$.

Soit H, un sous-graphe q-connexe de G, et on pose j le plus grand indice tel que $V(Q_j) \bigcap V(H) \neq \emptyset$. Supposons $j < k - 1$, montrons que l'on obtient une contradiction. Soit $x \in V(Q_j) \bigcap V(H)$. On considère les cas suivants :

$x = a_j$ ou $x = b_j$. Comme H est q-connexe, $\deg(x) \geq q$. Or, dans G, x est un incident à $q-1$ arêtes de Q_{j-1} et à $q-1$ arêtes de Q_j. Donc, dans $E(H)$, il existe une arête de Q_j. Donc, il existe $y \in V(Q_j) \bigcap V(H) \setminus \{a_j, b_j\}$. On peut donc se ramener à l'un des cas suivants :

$x \notin \{a_j, b_j, a_{j+1}, b_{j+1}\}$. Comme H est q-connexe, $\deg(x) \geq q$. Or, dans G, $\deg(x) = q$. Donc toutes les arêtes incidentes à x dans G appartiennent à $E(H)$. On en déduit que a_{j+1} et b_{j+1} appartiennent à $V(H)$. On se ramène donc au dernier cas :

$x = a_{j+1}$ ou $x = b_{j+1}$. Alors, $V(Q_{j+1}) \bigcap V(H) \neq \emptyset$ ce qui contredit le fait que j était choisi comme le plus grand indice vérifiant cette propriété.

Donc $j = k-1$. On montrerait de même que le plus petit indice tel que $V(Q_j) \bigcap V(H) \neq \emptyset$ est 0. On montre maintenant que pour tout $j \in \{0, \cdots, k-1\}$, $V(H) \bigcap V(Q_j) = V(Q_j)$. On sait qu'il existe $x \in V(Q_0) \bigcap V(H)$ et $y \in V(Q_{k-1}) \bigcap V(H)$. Supposons qu'il existe $0 < j < k-1$ tel que $V(Q_j) \bigcap V(H) = \emptyset$, alors H aurait au moins deux composantes connexes, l'une contenant x et l'autre contenant y, ce qui contredit le fait que H est q-connexe. Donc pour tout $j \in \{0, \cdots, k-1\}$, $V(H) \bigcap V(Q_j) \neq \emptyset$. Supposons qu'il existe $j \in \{0, \cdots, k-1\}$, tel que $V(H) \bigcap V(Q_j) \subset \{a_j, b_j, a_{j+1}, b_{j+1}\}$, alors en supprimant les arêtes $\{a_j, a_{j+1}\}$ et $\{b_j, b_{j+1}\}$, on déconnecte le graphe H. Comme on a supposé que H était q-connexe avec $q > 2$, on aboutit à une contradiction. Donc pour tout $j \in \{0, \cdots, k-1\}$, $V(H) \bigcap V(Q_j) \setminus \{a_j, b_j, a_{j+1}, b_{j+1}\} \neq \emptyset$. Soit $j \in \{0, \cdots, k-1\}$ et soit $x \in V(H) \bigcap V(Q_j) \setminus \{a_j, b_j, a_{j+1}, b_{j+1}\}$. Comme H est q-connexe, $\deg(x) \geq q$. Or, dans G, $\deg(x) = q$. Donc toutes les arêtes incidentes à x dans G appartiennent à $E(H)$, donc $V(H) \bigcap V(Q_j) = V(Q_j)$. On en déduit que $H = G$ et donc que $\mathbf{ctw}_q(G) = n-1$. ■

4 Perspectives

Notre étude ouvre de nombreuses perspectives. En particulier, l'étude des paramètres q-connexes soulèvent de nombreuses questions. Par exemple, a-t-on $\mathbf{mcx}_q(G) = \mathbf{cx}_q(G)$ pour tout graphe q-connexe G, quelque soit $q \geq 1$? C'est une question difficile et on ne sait même pas si cette égalité est satisfaite pour $q = 1$. Considérer la question pour q grand pourrait forcer d'adopter une nouvelle approche et de développer de nouveaux outils. De manière similaire, on sait que $\mathbf{cx}(G)/\mathbf{x}(G) = O(\log n)$, mais la preuve de ce résultat ne semble pas s'étendre au cas q-connexe pour $q > 1$. Ainsi, il est naturelle de chercher une borne pour le rapport $\mathbf{cx}_q(G)/\mathbf{cx}_{q-1}(G)$. En fait, nous conjecturons que pour tout $q \geq 1$, il existe une constante k telle que pour tout graphe q-connexe G, $\mathbf{cx}_q(G) \leq k\, \mathbf{cx}_{q-1}(G)$. Dans le cas $q = 1$, cette constante est conjecturée comme valant 2. Enfin, il serait également intéressant de généraliser à la q-connexité pour $q \geq 2$ le théorème selon lequel $\mathbf{cx}(G) \leq \mathbf{ctw}(G) * \log n$. Plus formellement, est-il vrai que pour tout graphe G de n sommets et possédant une décomposition arborescente q-connexe, $\mathbf{cx}_q(G) \leq \mathbf{ctw}_q(G) * \log n$? La preuve dans le cas $q = 1$ ne semble pas se généraliser aisément.

Par ailleurs, il serait intéressant de chercher à généraliser au cas connexe les algorithmes de construction définis dans le cas non connexe. En particulier, considérant la classe $\mathcal{C}_k$ des graphes de largeur d'arborescence connexe bornée par k, existe-t'il un algorithme exponentiel en k mais polynomial en le nombre de sommet permettant de calculer la largeur d'arborescence connexe de tout graphe dans $\mathcal{C}_k$?

Enfin, une autre voie d'investigation pourrait consister en l'étude des paramètres comme la largeur d'arborescence ou la largeur linéaire (connexe ou pas) dans dans les graphes possédant des propriétés typiques des grands réseaux d'interactions comme le Web ou les réseaux de connaissance. L'impact sur l'encerclement et l'expansion de propriétés "petits mondes" pourrait en effet avoir des conséquences pratiques importantes pour le partitionnement en communautés ou la recherche d'information.

Références

[1] L. Barrière, P. Flocchini, P. Fraigniaud, and N. Santoro. Capture of an intruder by mobile agents. In 14th ACM Symp. on Parallel Algorithms and Architectures (SPAA), pages 200-209, 2002.

[2] L. Barrière, P. Fraigniaud, N. Santoro, and D. Thilikos. Connected and Internal Graph Searching. In 29th Workshop on Graph Theoretic Concepts in Computer Science (WG), Springer-Verlag, LNCS 2880, pages 34–45, 2003.

[3] D. Bienstock and P. Seymour. Monotonicity in graph searching. Journal of Algorithms 12 :239–245, 1991.

[4] H. L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. Theor. Comp. Sc. 209 :1–45, 1998.

[5] H. L. Bodlaender, J. Gilbert, H. Hafsteinsson et T. Kloks. Approximating Treewidth, Pathwidth, Frontsize, and Shortest Elimination Tree. Journal of Algorithms 18 :238-255, 1995.

[6] V. Bouchitté, D. Kratsch, H. Müller et I. Todinca. On treewidth approximations Discrete Applied Mathematics 136 :183-196, 2004.

[7] R. Breisch An intuitive approach to speleotopology. Southwestern Cavers VI(5),72-78, 1967

[8] N. Clarke and R. Nowakowski Cops, Robber and Traps Utilitas Math. 60(2001), 91-98

[9] J. A. Ellis, I.H. Sudborough, J.S. Turner. The Vertex Separation and Search Number of a Graph Information and computation 113 :50-79,1994.

[10] R. Diestel. Graph Theory. Graduate Text in Mathematics, second edition,2000

[11] G. Even, J. Naor, S. Rao et B. Schieber. Fast approximate graph partitioning algorithms. SIAM J. Comput. 28(6) :2187-2214, 1999.

[12] F. Fomin, P. Fraigniaud, D. Thilikos. The Price of Connectedness in Expansions. Rapport Technique No. LSI-04-28-R, UPC Barcelone,2004.

[13] P. Fraigniaud and D. Ilcinkas. Digraphs Exploration with Little Memory. In 21st Symposium on Theoretical Aspects of Computer Science (STACS), LNCS 2296, pages 246-257, 2004.

[14] M. C. Golumbic. Algorithmic graph theory and perfect graphs. Computer Science and Applied Mathematics, 1980.

[15] N. G. Kinnersley. The Vertex Separation number of a graph equals its path-width. Information Processing Letters 42 :345-350,1992.

[16] A. LaPaugh. Recontamination does not help to search a graph. JACM, 40(2) :224-245,1993.

[17] L. Kirousis, C. Papadimitriou. Searching and Pebbling. Theoretical Computer Science 47 :205-218,1986.

[18] N. Megiddo, S. Hakimi, M. Garey, D. Johnson and C. Papadimitriou. The complexity of searching a graph. Journal of the ACM 35(1) :18-44,1988.

[19] T. Parson. Pursuit-evasion in a graph. Theory and Applications of Graphs, Lecture Notes in Mathematics, Springer-Verlag :426-441,1976.

[20] N. Robertson and P. D. Seymour. Graph minors II, Algorithmic Aspects of Tree-Width, J. of Algorithms 7 :309-322,1986.

[21] N. Robertson and P. D. Seymour. Graph minors X, Obstructions to tree-decomposition, J. Combin. Theory Ser. B, 52 :153–190, 1991.

[22] K. Skodinis. Computing optimal linear layouts of trees in linear time. In 8th European Symp. on Algorithms (ESA'00), Springer, LNCS 1879 :403-414,2000.

[23] D. M. Thilikos. Algorithms and obstructions for linear-width and related search parameters. Discrete Applied Mathematics 105 :239-271, 2000.