N° d'ordre : 3490

THÈSE

PRÉSENTÉE À

L'UNIVERSITÉ BORDEAUX I

ÉCOLE DOCTORALE DE MATHÉMATIQUES ET D'INFORMATIQUE

Par **Arnaud LABOUREL**

POUR OBTENIR LE GRADE DE

DOCTEUR

SPÉCIALITÉ : INFORMATIQUE

Partition d'arêtes et représentation implicite de graphes

Soutenue le : 10 décembre

Après avis des rapporteurs :

Dieter Kratsch . Professeur
Ioan Todinca ... Professeur

Devant la commission d'examen composée de :

Bruno Courcelle	Professeur	Président
Dieter Kratsch .	Professeur	Examinateur
Ioan Todinca ...	Professeur	Examinateur
Cyril Gavoille ..	Professeur	Examinateur
Michel Habib ..	Professeur	Examinateur
André Raspaud	Professeur	Rapporteur

– 2007 –

Remerciements

Je voudrais tout d'abord remercier Cyril Gavoille mon directeur de thèse. Il a été dans un premier temps mon directeur de stage de Master deuxième année puis m'as proposé ce sujet de thèse. Il a toujours su être disponible, compréhensif et pédagogue. J'ai vraiment appris beaucoup à ses cotés et je lui témoigne ici de toute ma gratitude.

Je tiens à vivement remercié ici :

Ioan Todinca pour avoir accepté de rapporter ce travail malgré ses responsabilités aux LIFO. J'ai particulièrement apprécié les discussions que j'ai pu avoir avec lui durant mon mon bref passage au LIFO quelques semaines avant la soutenance ;

Dieter Kratsch pour avoir accepté de s'astreindre à la même tache, pour la précision de sa lecture et la pertinence de ses remarques ;

Bruno Courcelle qui avait dirigé mes tout premiers pas dans la recherche lors du premier semestre de Master 2. Qu'il ait accepté de faire partie de mon jury est pour moi un grand honneur ;

Michel Habib pour avoir pris la peine de participer à mon jury de thèse.

André Raspaud pour avoir participer à mon jury de thèse et accepter d'en être le rapporteur.

Je tiens à remercier également les personnes qui ont contribué à rendre cette période de thèse agréable.

Je pense en particulier à Nicolas Bonichon avec qui j'ai eu l'occasion de travailler. Il a toujours été de bons conseils et son aide me fut précieuse tout au long de ma thèse.

Je tiens aussi à remercier tous ceux avec qui j'ai partagé un bureau durant ces trois années. Je pense tout d'abord à Daniel Gonçalves avec qui j'ai partagé le grand bureau de la salle 122. J'ai toujours admiré sa capacité à trouver des solutions innovantes à des problèmes difficiles.

Je tiens aussi à remercier Fabrice Bazzaro et Mikaël Montassier pour l'année durant laquelle on a partagé le même bureau. Leur enthousiasme pour la recherche est vraiment communicatif.

Finalement, je tiens à remercier The Quang Tran et plus particulièrement Louis Esperet avec qui j'ai partagé le même bureau durant la dernière année de ma thèse. Leur aide m'as été précieuse durant cette période difficile.

Et merci à tous les autres, Mamadou Kanté, Pascal Ochem, Charles Lales, Aida Ouangraoua, qui ont contribué à la mise en place d'une ambiance de travail conviviale.

Je n'oublie pas non plus le personnel administratif du LaBRI, en particulier Phillipe Biais, Cathy Roubineau et Fabienne Clairand qui m'ont été d'une aide précieuse.

La thèse ayant été aussi pour moi l'occasion de faire mes débuts en tant qu'enseignant, je tiens aussi à remercier l'équipe pédagogique du département informatique l'UFR Bordeaux 1 et plus particulièrement Giuliana Bianchi et Christian Retoré qui m'ont beaucoup aidé pour ma première année d'enseignement.

Je conclurai en remerciant de tout cœur ma famille et plus particulièrement mes parents qui m'ont toujours soutenu. Sans leur aide et leur soutien indéfectible, cette thèse n'aurait sans doute jamais vu le jour.

Partition d'arêtes et représentation implicite de graphes

Résumé : La représentation implicite de graphes a été introduite en 1966 par Breuer afin de calculer l'adjacence dans les graphes à partir de données locales encodées par chaque sommet. Dans ce document, on s'est intéressé à cette notion pour de nombreuses familles de graphes et en particulier pour les graphes sur surfaces.

Afin de pouvoir utiliser les représentations implicites existantes sur les forêts, on montre tout d'abord l'existence d'une partition des arêtes des graphes de genre d'Euler g en trois forêts plus un ensemble d'au plus $3g-3$ arêtes. De plus, on décrit pour le cas des graphes toriques, un algorithme réalisant cette partition en temps linéaire.

Ensuite, on donne une représentation implicite pour les graphes de degré borné par k en $k/2 \log n + O(1)$ bits par sommet pour k pair et en $(\lceil k/2 \rceil - 1/k) \log n + o(\log n)$ bits par sommet pour k impair.

On présente une représentation implicite pour les arbres de degré interne borné par $\hat{d}$ ayant n sommets avec des étiquettes de $\log n + O(\log \hat{d})$ bits.

Enfin, on décrit une représentation implicite pour les graphes planaires et les graphes de genre borné en $(2 + o(1)) \log n$ bits déduite d'une représentation implicite pour les graphes de largeur arborescente bornée avec des étiquettes de $(1 + o(1)) \log n$ bits. On utilise une technique similaire afin d'obtenir un schéma k-relationnel sur les arbres.

Mots clés : Représentation distribuée, graphes universels, partition d'arêtes, graphes sur surface, largeur arborescente, arbres de degré interne borné, schéma k-relationnel

Discipline : Informatique

LaBRI,
Universitée Bordeaux 1,
351, cours de la libération
33405 Talence Cedex (FRANCE)

Edge partition and implicit representation of graphs

Abstract : Implicit representation of graphs was introduced by Breuer in 1966 in order to compute adjacency in graphs from local information stored on vertices. In this document, we are interested into implicit representation for several graph familys and more particularly for graphs on surfaces.

In order to use known implicit representation on forests, we show that there exists an edge partition of graphs of Euler genus g into three forests plus a set of at most $3g-3$ edges. For the particular case of toroidal graphs, we describe an algorithm computing this partition in linear time.

Then, we give an implicit representation of graphs of degree at most k using $k/2\log n + O(1)$ bits per vertex for even k and $(\lceil k/2\rceil - 1/k)\log n + o(\log n)$ bits per vertex for odd k.

We give an implicit representation for trees of n vertices with inner degree at most $\hat{d}$ with labels of $\log n + O(\log \hat{d})$ bits.

Finally, we describe an implicit representation for planar graphs and bounded genus graphs using $(2 + o(1))\log n$ bits deduced from an implicit representation of graphs of bounded treewidth with labels of $(1 + o(1))\log n$ bits. We use a similar technique to obtain a k-relationship scheme for trees.

Keywords : Distributed representation, universal graphs, edge partition, graphs on surface, trewidth, bounded inner degree trees, k-relationship scheme

Discipline : Computer-Science

LaBRI,
Université Bordeaux 1,
351, cours de la libération
33405 Talence Cedex (FRANCE)

Table des matières

Table des figures

Introduction

Introduction générale

Les graphes permettent de modéliser de nombreux problèmes comme par exemple la communication et le calcul distribué dans les réseaux. Afin de résoudre ces problèmes, il est souvent nécessaire de pouvoir représenter en mémoire des graphes. Il existe deux manières élémentaires d'accomplir cette tache, utiliser une matrice d'adjacence ou bien des listes d'adjacence. Dans certains cas, ces deux représentations ne sont pas adaptées. Par exemple, pour certains algorithmes de calcul distribué, il n'est pas toujours utile que chaque sommet du réseau connaisse la structure globale de celui-ci [50, 66]. Il est donc plus efficace de distribuer localement la représentation globale du réseau parmi ses sommets. Le but est de stocker localement une information utile pour l'algorithme de calcul distribué et de la rendre commodément accessible.

Pour réaliser cela, il existe une troisième technique, connue sous les noms de *représentation distribuée* et de *schéma d'étiquetage*. Cette technique consiste à fixer une requête portant sur un ou plusieurs sommets du graphe à laquelle la représentation distribuée doit permettre de répondre localement. La requête choisie dépend évidemment de l'application considérée. Dans le cas d'un algorithme distribué, la requête doit par exemple recouvrir toutes les informations utiles au bon déroulement de l'algorithme. Concrètement, une information sous la forme d'une étiquette binaire est attribuée à chaque sommet du graphe telle que la requête choisie puisse être calculée en utilisant seulement les étiquettes des sommets concernés. Le problème est de minimiser la longueur des étiquettes tout en gardant un calcul des requêtes rapide. En effet, l'ensemble des étiquettes attribuées aux sommets va généralement coder une représentation du graphe. Minimiser cette information semble donc essentiel surtout pour la représentation en mémoire de grands graphes dans le cadre d'applications pratiques. De plus, comme nous le verrons plus tard, ce problème d'optimisation est aussi intéressant au niveau théorique (Conjecture de représentation implicite).

Représentation distribuée

Dans cette partie, nous détaillons les différents travaux existants dans le domaine de la représentation distribuée.

Requêtes d'adjacence

Le premier type de requête qui fut considéré pour le problème de représentation distribuée est la requête d'adjacence. Un des premiers travaux portant sur le sujet est celui de Breuer [23] démontrant que le le test d'adjacence entre sommets de n'importe quel graphe peut se ramener à un simple test de distance de Hamming sur des étiquettes binaires préalablement attribuées. Cette notion a ensuite fait l'objet d'autres travaux [71, 58] sous le nom de *représentation implicite* ou *schéma (d'étiquetage) d'adjacence*. Formellement, une représentation implicite d'une famille de graphe $\mathcal{F}$ donnée consiste en deux fonctions : une première fonction dite d'étiquetage attribuant à chaque sommet des graphes d'une famille de graphe $\mathcal{F}$ donnée une étiquette binaire distincte des autres sommets du graphe et une deuxième fonction dite d'adjacence déterminant l'adjacence entre deux sommets en utilisant seulement les étiquettes des sommets concernés. De plus, les définitions de [58] impliquent que les étiquettes soit de taille logarithmique par rapport à l'ordre du graphe (en $O(\log n)$ pour un graphe de n sommets) et que le temps de calcul de calcul de la requête soit polynomial en la taille des étiquettes et donc polylogarithmique par rapport à l'ordre du graphe. Dans leur papier [58], Kannan, Naor et Rudich montre que de nombreuses familles de graphes admettent une représentation implicite en $O(\log n)$:

- les graphes ayant une arboricité borné,
- les graphes ayant un genre borné,
- les graphes ayant une largeur arborescente bornée,
- les graphes d'intervalles,
- les graphes de permutation.

Ils montrent aussi dans le même papier que la borne supérieure imposée sur la taille des étiquettes restreint l'ensemble des familles admettant une représentation implicite aux familles de graphes ayant au plus $2^{O(n \log n)}$ graphes étiquetés (avec des entiers de 1 à n) de n sommets. Considérant ce résultat et le fait que toutes les familles pour lesquelles ils ont prouvé l'existence d'une représentation implicite sont héréditaires c'est-à-dire closes par sous-graphe induit, ils ont proposé la conjecture suivante.

Problème ouvert 1 (Conjecture de représentation implicite)**.** *Est-ce que toutes les familles héréditaires ayant $2^{O(n \log n)}$ graphes étiquetés de n sommets admettent une représentation implicite ?*

Cette conjecture est toujours ouverte à l'heure actuelle et semble être l'un des problèmes les plus intéressants et difficiles du domaine. Grossièrement, prouver cette conjecture revient à prouver que l'on peut répartir de manière asymptotiquement optimale l'information du graphe sur chacun de ses sommets de manière à pouvoir calculer l'adjacence localement. En effet, les graphes à n sommets de ces familles peuvent être codés globalement avec $O(n \log n)$ bits puisqu'il en existe $2^{O(n \log n)}$ et donc une représentation asymptotiquement optimale utiliserait bien $O(n \log n/n) = O(\log n)$ bits par sommets.

Une généralisation de la notion de représentation implicite a été proposée par Spinrad dans [88]. Il est en effet possible de passer outre la restriction sur le nombre de graphes de la famille. Pour cela, il suffit de ne garder que la propriété d'optimalité en espace mémoire de la

représentation. Soit $\mathcal{F}$ une famille de graphe ayant au plus $2^{f(n)}$ graphes de n sommets. Les graphes de $\mathcal{F}$ ayant n sommets peuvent être codés globalement en utilisant $O(f(n))$ bits. Une *représentation implicite généralisée* de $\mathcal{F}$ autorise donc d'attribuer jusqu'à $O(f(n)/n)$ bits par sommets. Cette généralisation de la notion de représentation implicite aboutit naturellement à la généralisation de la conjecture de représentation implicite.

Problème ouvert 2 (Conjecture de représentation implicite généralisée)**.** *Est-ce que toutes les familles héréditaires admettent une représentation implicite généralisée ?*

Toujours dans le même papier, les auteurs montre une relation avec un problème ancien en théorie des graphes connu sous le nom de graphe universel induit. Soit $\mathcal{F}$ une famille de graphes. Un graphe est dit *$\mathcal{F}$-universel induit* (notion introduite pour la première fois par [41]) s'il contient tous les graphes de la famille $\mathcal{F}$ en tant que sous-graphes induits. Si la famille $\mathcal{F}$ admet une représentation implicite utilisant $k \log n$ bits alors elle admet un graphe universel induit de n^k sommets [58]. Inversement, si on relache la contrainte sur le temps d'exécution de la requête d'adjacence l'existence d'un graphe $\mathcal{F}$-universel induit de n^k sommets implique l'existence d'une représentation implicite de $\mathcal{F}$ utilisant $k \log n$ bits sommets. Il y donc équivalence entre les deux problèmes. Clairement cette relation rends très importante la valeur de la constante k multiplicative de la représentation implicite et beaucoup de travaux se sont intéressés à l'optimisation de cette constante pour des familles de graphes spécifiques.

L'une des familles les plus étudiées pour ce problème d'optimisation est la famille des forêts et par extension la famille des graphes d'arboricité bornée. Kannan, Naor et Rudich [58] ont décrit le premier schéma connu pour la famille des graphes d'arboricité k qui utilise des étiquettes de $(k+1) \log n$ bits. Ce schéma très simple se résume en trois étapes :

- enraciner les composantes connexes des arbres formant les k forêts de la partition,
- attribuer des identifiants (étiquettes distinctes) de $\log n$ à chaque sommmet,
- encoder dans chaque sommet, en plus de son identifiant, les identifiants de ses pères dans les k arbres auxquels il appartient.

Le test d'adjacence est lui aussi très simple car il se résume à tester s'il l'un des sommets est le père de l'autre dans l'un des k arbres, cette opération s'eefectuant très facilement car chaque sommet possède l'identifiant de ses pères.

Ce schéma a été ensuite amélioré de manière non-triviale par Chung [27] en un schéma utilisant $k \log n + k \log \log n + O(1)$ bits. En fait, ce schéma est déduit d'un graphe universel induit de $O((n \log n)^k)$ sommets. La construction de ce graphe universel induit utilise un graphe *universel*, c'est-à-dire un graphe contenant tous les graphes d'une famille de graphe en tant que sous-graphe. Le principal résultat du papier étant une transformation permettant de passer d'un graphe universel à un graphe universel induit en fonction de l'arboricité de la famille.

Finalement, le meilleur schéma existant pour les arbres utilise $k \log n + O(k \log^* n)$ [9]. La technique utilisée est cette fois ci beaucoup plus complexe que dans le cas du schéma de [58]. Les auteurs décrivent d'abord un schéma d'adjacence de $\log n + O(\log \log n)$ bits. Le schéma repose sur un parcours préfixe de l'arbre orienté. Chaque sommet va coder

son numéro ($\log n$ bits) dans le parcours et ainsi que d'autres informations ($\log \log n$ bits) comme le logarithme de la différence avec numéro avec son père et celui d'un de ses fils appellé fils lourd. La force de ce schéma est de combiner astucieusement plusieurs conditions nécessaires à l'adjacence afin d'obtenir une condition suffisante. Le représentation implicite ainsi obtenue utilise $\log n + \log \log n$ bits par sommets. Afin d'obtenir une représentation en $\log n + \log^* n$ bits, les auteurs utilise une partition spéciale des arbres et le fait qu'il est possible de construire une représentation implicite utilisant peu de bits sur les feuilles. Leur méthode repose sur la combinaison de deux schémas : un schéma dont les sommets internes ont besoin de coder $\log n + \log \log n$ bits mais dont les feuilles utilisent seulement $\log n + O(1)$ bits et un autre schéma dont la taille maximum des étiquettes est plus petite. En combinant ces deux schémas grâce à leur partition spéciale des arbres, ils obtiennent un schéma dont la taille maximum des étiquettes est plus petite. Le schéma final est obtenue par une réapplication récursive de ce procédé $O(\log^* n)$ fois.

Pour de nombreuses familles, le schéma dépendant de l'arboricité est le meilleur connu. Par exemple, pour les graphes planaires, le schéma le plus compact est en $3 \log n + O(\log^* n)$ et est déduit du fait que les graphes planaires ont une arboricité 3. Il en est de même pour les graphes de genre borné et les graphes excluant un mineur fixé.

D'autres familles ont bien sûr été étudiées. Parmi celles-ci, il est intéressant de citer les familles issues d'intersection d'objets géométriques comme les graphes d'intervalles et les graphes de permutation. La particularité de ces familles est qu'une représentation implicite peut facilement être déduite de la définition même de la famille. Dans les graphes d'intervalles, chaque sommet correspond à un segment dans la droite réelle et deux sommets sont adjacents si leur segment s'intersectent. Pour obtenir une représentation implicite de ces graphes, il suffit de coder les deux extrémités des intervalles. Celles-ci peuvent être codée avec $2 \log n + O(1)$ bits pour un graphe de n sommets car on peut facilement se réduire au cas où les extrémités sont des entiers compris entre 1 et $2n$. Avec cette information, il est possible de tester l'intersection de deux segments et donc l'adjacence entre deux sommets.

Les graphes de permutation sont aussi définies par des intersections de segments. La différence est que l'on utilise cette fois-ci deux droites parallèles. Chaque segment a une extrémité dans chacune des deux droites. La aussi il est possible d'obtenir une représentation implicite de $2 \log n + O(1)$ bits en codant chaque segment et donc chaque sommet avec deux entiers, un premier entier représentant la position de l'extrémité dans la première droite et le deuxième la position de l'autre extrémité dans la deuxième droite. Là aussi, grâce à ces informations il est possible de déterminer si deux segment s'intersectent et donc si deux sommets sont adjacents.

Pour ces deux familles de graphes, la problèmatique d'optimisation de la représentation implicite n'est pas dans la description d'une représentation implicite optimale mais bien de prouver que celle obtenue à partir de la définition de la famille est optimale. En effet, s'il est évident que pour les deux modèles, qu'il faut $2 \log n$ bits pour coder les segments et tester leur intersection, il n'est pas clair que ce nombre de bits soit nécessaire pour le test d'adjacence. En effet, plusieurs arrangements différents de segments peuvent correspondre au même graphe et donc il n'est pas utile de considérer tous les arrangements de segments

possible pour l'adjacence. Némanmoins, il a été prouvé qu'il est nécessaire d'avoir $\log n - \Omega(\log \log n)$ bits par sommets afin de tester l'adjacence de manière locale dans ces deux familles de grahes [12, 49]. Les preuves de ces deux résultats reposent sur des techniques d'énumération afin d'obtenir le nombre de graphes étiquetés des familles.

Autres requêtes

Récemment, des requêtes plus complexes que la simple adjacence entre sommets ont été considérées pour la représentation distribuée. Cette extension se justifie par le fait que pour de nombreux problèmes, comme par exemple le routage dans les réseaux, la seule connaissance de l'adjacence entre sommets ne permet pas une résolution du problème à partir de données locales. En effet, retrouver un chemin entre deux sommets nécessiterait avec une représentation implicite de connaître au minimum toutes les informations relatives aux sommets du chemin.

Une des requêtes les plus étudiées de part le fait de ses applications pratiques évidentes pour les communications dans les réseaux est celle du routage qui consiste à trouver un chemin entre deux sommets. Plus précisément, étant donné deux sommets, ce type de représentation permet de déterminer de manière locale, c'est-à-dire à partir des informations attribuées par le schéma aux deux sommets, le numéro de port menant à la première arête d'un chemin allant d'un sommet à un autre. Évidemment, le chemin donné par la représentation n'est pas quelconque. Il existe deux variantes possibles : le routage exact dans lequel le chemin est un plus court chemin et le routage approximatif dans lequel le chemin a une longueur proche d'un plus court chemin. Dans ce dernier cas, le chemin obtenu a souvent pour longueur maximale un multiple de la distance entre les deux sommets. Le facteur multiplicatif entre la distance et la longueur du chemin est appelé facteur d'étirement. Ce type de représentation a fait l'objet d'étude pour de nombreuse familles de graphes comme les graphes d'intervalles et de permutation [37, 38], les arbres [44, 67] et les graphes pondérés [92].

Un autre type de requêtes étudiées pour la représentation distribuée est la requête de distance qui permet de déterminer la distance entre deux sommets du graphes de manière locale. Ce type de requête est strictement plus puissant que la requête d'adjacence et possède de nombreuses applications notamment pour la propagation de messages dans les réseaux. Le cas des familles de graphes d'intervalles [49], de permutation [12], des graphes planaires, des arbres, des graphes de degré borné et des graphes quelconques [32] ont déjà été traité. Il existe deux variantes à ce problème. Dans la première connue sous le nom de requête de distance approximative [45], au lieu de calculer la distance exacte entre les sommets, le schéma en donne seulement une approximation, généralement une valeur étant au plus une constante fois la distance. De manière similaire au cas du routage, ce facteur multiplicatif entre la véritable distance et la valeur donnée par le schéma est appelé facteur d'étirement. La seconde variante connues sous le nom de faible distance [59, 6] consiste à se fixer un paramètre qui va borner la distance maximale qui peut calculée par le schéma. Le schéma ne pourra ainsi répondre qu'aux requêtes concernant deux sommets proches.

Comme il a été dit précédemment, la famille des arbres est l'une des plus étudiée pour le problème de représentation implicite. C'est assez naturellement qu'elle fut l'une des premières étudiées pour des représentation distribuée répondant à des requêtes autres que l'adjacence. Ce sont surtout les arbres enracinés et les relation d'ancêtres entre sommets qui ont été étudiées. L'un des premier type de requête qui fut étudié est la relation d'ancêtre qui permet de déterminer si un sommet est ancêtre d'un autre [1]. Une variante de ce type de schéma est le schéma k-relationnel [7]. Un tel schéma permet de déterminer la distance de deux sommets par rapport à leur plus petit ancêtre commun si celui-ci est suffisamment proche, c'est-à-dire à une distance inférieure à k. Cela permet pour des sommets proche de connaître entièrement leur positions respectives par rapport à la racine. Un autre type de schéma existant est celui du plus petit ancêtre commun qui permet étant donné deux sommets de retrouver un identifiant de leur plus petit ancêtre commun [8].

Un autre type de requêtes étudiées est celles reliées à la connectivité. Dans ces représentations, le schéma détermine si deux sommets sont dans une même composante k-connexes [2, 1]. Un autre problème lié à la connectivité consiste à se poser la question de savoir si deux sommets sont dans la même composante connexe sachant que l'on a enlevé un certain ensemble de sommets au graphes [30].

Finalement, certains travaux se sont intéressé non pas à une représentation distribuée pour une requête précise mais plutôt à un ensemble de requête exprimable par une certaine logique et notamment en logique monadique du second ordre[31]. Une telle approche possède l'avantage de traiter de manière simultanée de nombreuses requêtes intéressantes comme l'adjacence, la distance et la connectivité. Évidemment, la puissance et la généralité de tels schémas rends difficile l'optimisation de la quantité d'informations attribuées à chaque sommet.

Plan du document

Partie I : Partition des arêtes des graphes de genre borné

Chapitre 1 : Présentation des graphes sur surfaces

Dans un premier temps, on rappelle quelques notions de théorie des graphes.

Ensuite, on présente de manière générale les graphes sur surfaces. On définit les notions de surfaces, plongements et cartes ainsi que les différentes variantes du genre d'un graphe : orientable, non-orientable et d'Euler. On rappelle aussi les résultats de base sur les graphes sur surface et notamment la formule d'Euler qui nous sera très utile dans le chapitre suivant.

Chapitre 2 : Partition des arêtes des graphes de genre borné

Dans ce chapitre, on s'intéresse aux partitions des arêtes des graphes sur surfaces. En résultat préliminaire, on donne un majorant en $O(\sqrt{g})$ pour l'arboricité des graphes de

genre d'Euler g. Ensuite, on minore par trois la densité arborescente des triangulations de surfaces de genre d'Euler $g \geqslant 1$ généralisant ainsi un résultat de Kundu [68]. Ce résultat nous permet de partitionner en trois forêts plus un ensemble d'au plus $3g - 3$ arêtes, les graphes de genre d'Euler g. On décrit ensuite un algorithme linéaire utilisant la notion de réaliseur de Schnyder [85] qui construit cette partition en temps linéaire pour les graphes toriques. On termine le chapitre par des propositions de généralisations des partitions connues pour les graphes planaires afin de les adapter aux graphes sur surfaces.

Ces résultats ont fait l'objet d'une présentation à une conférence internationale [19].

Partie II : Représentation implicite de graphes

Chapitre 3 : Graphes universels et variantes

Dans ce chapitre, on s'intéresse particulièrement aux graphes universels induits pour la famille des graphes de degré borné. On décrit la construction d'un graphe universel induit $U_{n,k}$ pour la famille des graphes de degré maximum k ayant n sommets. Pour k pair, le graphe $U_{n,k}$ a $O(n^{k/2})$ sommets et pour k impair, $U_{n,k}$ a $O(n^{\lceil k/2 \rceil - 1/k} \log^{2+2/k} n)$ sommets. On améliore ainsi un résultat de Butler [24].

Dans la deuxième partie du chapitre on s'intéresse au cas des graphes universels induits pour des graphes orientés. On donne d'abord une construction pour un graphe universel induit de $O(n^k)$ sommets pour la famille des graphes de degré entrant et sortant au plus k. Ensuite, on décrit une construction générale permettant de passer du cas orienté au cas non-orienté et on conclue par quelques problèmes ouverts.

L'ensemble de ce travail a fait l'objet d'un rapport de recherche [42].

Chapitre 4 : Schémas d'adjacence utilisant le parcours bondissant

Dans ce chapitre, on s'intéresse à un schéma d'adjacence pour les arbres enracinés de degré interne borné par $\hat{d}$. On présente un schéma d'adjacence pour les arbres de n sommets avec des étiquettes de $\log n + O(\log \hat{d})$ bits, ce qui nous donne un schéma en $\log n + O(1)$ bits pour les chenilles et les arbres de degré borné. Bien que ce résultat soit en lui-même intéressant car les schémas connus pour les arbres ne se simplifient pas pour ces sous-familles, c'est surtout la technique des schémas qui est la plus importante. En effet, cette technique dite de "parcours bondissant" diffère grandement des techniques d'étiquetage précédemment connues et on peut espérer pouvoir l'utiliser sur d'autres famille de graphes.

Ce travail a fait l'objet de présentations à des conférences internationales [20, 21].

Chapitre 5 : Schéma d'adjacence pour les graphes planaires

On décrit dans ce chapitre un schéma d'adjacence pour les graphes planaires en $(2 + o(1)) \log n$ bits. Ce schéma est une amélioration notable du précédent meilleur schéma

connu qui est en $(3 + o(1)) \log n$ bits [9]. En fait, ce schéma est déduit d'un schéma pour les graphes de largeur arborescente k avec des étiquettes de $\log n + O(k \log \log(n/k))$ bits.

La technique utilisée pour obtenir le schéma pour les graphes de largeur arborescente k peut aussi être utilisée pour obtenir un schéma k-relationnel sur les arbres. Le schéma obtenu, qui permet entre autre le calcul exact de distance entre sommets à distance au plus k, améliore un résultat récent de [6, 7] de manière non-triviale.

Ces travaux ont fait l'objet de trois présentations dans des conférences internationales [46, 48, 47].

Première partie

Partition des arêtes des graphes de genre borné

Chapitre 1

Présentation des graphes sur surfaces

Dans ce chapitre, on établit quelques outils de base pour l'étude des graphes plongés sur une surface. Pour plus de détails sur les graphes plongés sur surfaces, on conseille au lecteur de se référer à l'excellent livre de Mohar et Thomassen portant sur le sujet [70]. Ces outils nous seront utiles dans le chapitre suivant dans lequel on présente des partitions d'arêtes pour ces graphes.

1.1 Éléments de théorie des graphes

Un graphe $G = (V(G), E(G))$ est constitué d'un ensemble de *sommets*, noté $V(G)$, et d'un multi-ensemble d'*arêtes* constitué de paires de sommets. Une arête uv est une *boucle* si $u = v$. Une arête qui apparaît plusieurs fois dans l'ensemble $E(G)$ est appelée *arête multiple*. Un graphe qui possède des boucles ou des arêtes multiples est un *graphe multiple*. À l'inverse, un graphe n'ayant ni boucle, ni arête multiple est dit simple.

On dit que G' est un *sous-graphe* de G si $V(G) \subset V(G')$ et $E(G) \subset E(G')$. Le graphe G' est un *sous-graphe induit* si pour tout couple de sommets (x, y) de $V(G')$, $(x, y) \in E(G) \Leftrightarrow (x, y) \in E(G')$. On note $G[D]$ le sous-graphe induit de G' ayant $D \subseteq V(G)$ comme ensemble de sommets. De même on définit un sous-graphe induit par un ensemble $C \subseteq E(G)$ d'arêtes noté $G[C]$ comme le sous-graphe $G[C] = (\{x \mid \exists y \in V(G), xy \in C\}, C)$.

Un sommet u est dit *voisin* d'un sommet v dans G si $(u, v) \in E(G)$. Le *degré* du sommet u est égal au nombre de ses voisins.

Un *graphe orienté* $G = (V(G), A(G))$ est constitué d'un ensemble de sommets $V(G)$ et d'un ensemble d'*arcs* $A(G)$ constitué de couples de sommets. Un arc $e = (u, v)$ est un *arc sortant* de u et un *arc entrant* de v. Le sommet u est appelé la *source* de e et v la *cible* de e. Le *degré entrant* d'un sommet u est le nombre d'arcs ayant u comme cible. De même, le *degré sortant* de u est le nombre d'arcs ayant u comme source.

Un *chemin* dans un graphe G est une suite de sommets $(v_1, v_2, \cdots, v_k)$ distincts de G tel que $\forall i \mid 1 \leqslant i \leqslant k-1, (v_i, v_{i+1}) \in E(G)$. La *longueur* d'un chemin est le nombre d'arêtes du chemin. Un *cycle* est un chemin tel que $v_1 = v_k$. Un graphe sans cycle est dit

acyclique. Deux sommets u et v sont dit *connectés* dans G s'il existe un chemin de u vers v dans G. Un graphe est *connexe* si tous ses sommets sont deux à deux connectés. On peut généraliser la notion de connectivité sur les graphes connexes comme suit. Un graphe G à n sommets ($n \geqslant k+1$) est dit *k-connexe* s'il faut supprimer au moins k sommets afin qu'il ne soit plus connexe. Les graphes 1-connexes sont exactement les graphes connexes. De manière similaire, G est *k-arête-connexe* s'il faut supprimer au moins k arêtes afin qu'il ne soit plus connexe.

Par exemple, le graphe G_1 de la Figure 1 possède sept sommets et dix arêtes. Les sommets $v_1, v_2, v_5, v_7, v_6, v_3$ forment un cycle de longueur six. Ce graphe est 2-connexe puisque la suppression de n'importe lequel de ses sommets le laisse connexe. Il n'est pas 3-connexe car si l'on supprime les sommets v_1 et v_7, le graphe n'est plus connexe. Les voisins du sommet v_7 sont les sommets v_2, v_4, v_5, v_6. Le sommet v_7 a donc un degré égal à quatre. Le graphe G_2 de la Figure 1 est un exemple de graphe orienté. Dans ce graphe, le degré entrant du sommet v_7 est égal à trois et son degré sortant égal à un.

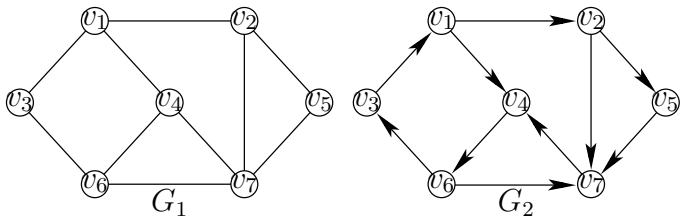


Figure 1 – Un exemple de graphe

1.2 Arbres, k-arbres et largeur arborescente

Un graphe acyclique est une *forêt*. Un *arbre* est une forêt connexe. Les sommets de degré 1 sont appelés *feuilles*, les autres sommets sont les sommets *internes*.

La notion d'arbre peut se généraliser en la notion de k-arbre. Un k-arbre est un graphe construit de faÃ§on récursive comme suit :

- le graphe complet de k sommets est un k-arbre
- le graphe G' obtenu en ajoutant un sommet à k-arbre G relié à tous les sommets d'un sous-graphe complet de k sommets de G est un k-arbre.

Enfin un k-arbre partiel est un sous-graphe d'un k-arbre. Il existe une autre définition possible correspondant à la même famille de graphes que celle des k-arbres partiels. Ce sont les graphes de largeur arborescente au plus k.

Définition 1. *Soit G un graphe, une décomposition arborescente de G est un couple (X, T), où $X = \{X_1, \cdots, X_n\}$ est une famille de sous-ensembles de sommets de $V(G)$, et T est un arbre dont les sommets sont étiquetés par ces sous-ensembles X_i, tels que :*

- *l'union de tous les X_i de X est égale à V,*

- *pour toute arête (v, w) de E, il existe un sommet X_i de l'arbre T qui contient v et w,*
- *si X_i et X_j contiennent un même sommet v, alors tous les sommets X_z de T sur le chemin entre X_i et X_j contiennent v.*

La *largeur* d'une décomposition est égale à $\mathbf{max}\,|X_i| - 1$. La largeur arborescente d'un graphe est le minimum des largeurs des décompositions arborescentes du graphe.

Remarque 1. *La dernière condition de la définition 1 est équivalente au fait que tous les sommets X_i de l'arbre T contenant un sommet v de G induisent un sous-arbre de T.*

Les notions de décomposition arborescente et de largeur arborescente (sous des noms différents) furent introduites pour la première fois par Halin [56]. Ces deux notions furent ensuite réintroduites par Robertson et Seymour [83].

1.3 Plongements de graphes

1.3.1 Espace topologique et plongement de graphe

Afin de définir proprement le plongement d'un graphe sur une surface, il faut définir le plongement d'un graphe dans un espace topologique. L'un des espaces topologiques les plus usités pour les plongements de graphes étant le plan euclidien $\mathbb{R}^2$ qui correspond aux *graphes planaires*, c'est-à-dire les graphes que l'on peut dessiner sur le plan sans croisement d'arêtes.

Un espace topologique est un couple (E, T), où E est un ensemble et T un ensemble de parties de E appelé *ouverts* de (E, T) vérifiant les propriétés suivantes :

- L'ensemble vide et E sont ouverts,
- Toute réunion d'ouverts est un ouvert,
- Toute intersection de deux ouverts est un ouvert.

L'ensemble T est appelée *topologie* de E. Les *fermés* d'une topologie sont les complémentaires de ses ouverts. Un *voisinage* d'un point est un sous-ensemble de X contenant un ouvert contenant ce point. Un espace topologique est dit *séparé* ou de *Hausdorff* si pour deux points distincts x et y quelconques de E, il existe un voisinage de x et un voisinage de y disjoints.

Soit X un espace topologique. Une *courbe* ou *arc* est une fonction continue $f : [0, 1] \to X$. Une courbe $A = f([0, 1])$ est dite comme *joignant* ses deux extrémités $f(0)$ et $f(1)$. Une courbe f est *simple* si f est injective. Une courbe est *fermée* si $f(0) = f(1)$. Un espace topologique est dit *connexe par arcs* si toute paire d'éléments de X peut être jointe par une courbe simple. L'existence d'une courbe simple entre deux éléments de X définit une relation d'équivalence dans X dont les classes d'équivalence sont appelées les *régions* de X. Un ensemble C sépare X si $X \setminus C$ n'est pas connexe par arcs. Les régions de $X \setminus C$ sont les faces de C.

Un graphe G est *plongé* dans un espace topologique X si les sommets de G sont des éléments disjoints de X et chaque arête xy est une courbe simple joignant dans l'espace X les éléments x et y. Un *plongement* d'un graphe G dans un espace topologique X est un graphe isomorphe à G qui est plongé dans X.

1.3.2 Surfaces

Pour les plongements de graphes, on se restreint volontairement à un certain type d'espace topologique. Une *surface* S est un espace topologique d'Hausdorff connexe et compact qui est localement homéomorphe au plan, c'est-à-dire que chaque point de S a un voisinage ouvert homéomorphe au disque ouvert dans $\mathbb{R}^2$.

Des exemples de surfaces peuvent être construits comme suit. Soit $\mathcal{F}$ une collection de polygones convexes deux à deux disjoints dans le plan avec des cotés de taille 1. Ces polygones ont au total m cotés $\sigma_1, \sigma_2, \cdots, \sigma_m$. On oriente arbitrairement chacun des cotés en choisissant une des deux extrémités comme point d'origine et on partitionne l'ensemble des cotés par paires. De l'union disjointes des polygones, on obtient un espace topologique S en identifiant les cotés d'une même paire de telle manière à ce que l'orientation soit respectée, c'est-à-dire que les points d'origine des deux cotés d'une paire soient identifiés comme étant un même sommet. On obtient bien de cette manière un espace topologique d'Hausdorff connexe qui est localement homéomorphe au plan. C'est trivial de vérifier que la condition est respectée pour les points à l'intérieur des polygones et à l'intérieur des cotés. Pour les points issus de l'identification des coins des polygones, on observe qu'après l'identification de deux cotés, chacun de ces points a un voisinage homéomorphe au demi-plan fermé ou au plan. Par conséquent, si S est connexe, ce que l'on supposera vrai, alors c'est une surface. Les cotés $\sigma_1, \sigma_2, \cdots, \sigma_m$ et leur extrémités définissent un graphe multiple connexe G plongé sur S. On dit alors que G est plongé de manière *cellulaire* sur S. Les polygones de $\mathcal{F}$ sont les faces de G. Si G est un graphe et toutes les faces sont des triangles alors G est une *triangulation de la surface* S et S est une *surface triangulée*.

Dans l'exemple de la figure 2, quatre triangles sont utilisés afin de créer un tétrahèdre qui est une surface triangulée homéomorphe à la sphère.

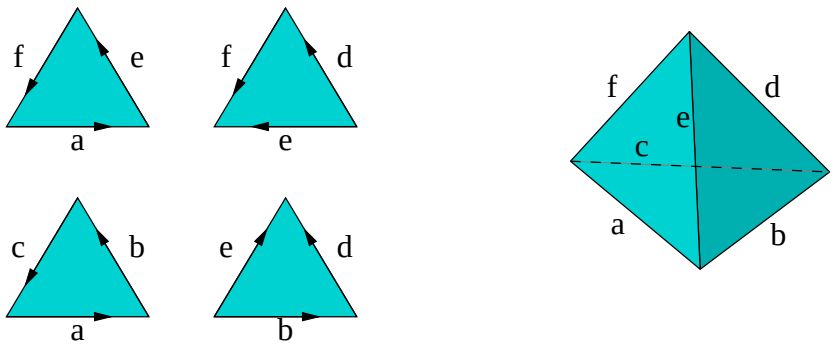


Figure 2 – Le tétraèdre.

On considère maintenant deux triangles disjoints T_1 et T_2 (dont les cotés sont de même longueur), dans une face F d'une surface S ayant un graphe G plongé de manière cellulaire. On construit une nouvelle surface S' en retirant de F l'intérieur de T_1 et T_2 et en identifiant T_1 et T_2. On oriente T_1 et T_2 de telle sorte que leur orientations ne concordent pas dans le plan défini par la face. Dans ce cas, on dit que la surface S' a été obtenue de S par l'ajout à S d'une *anse* (Voir figure 3(a)). Il existe une autre possibilité pour orienter les deux triangles qui consiste à les orienter de telle sorte à de que leurs orientations concordent dans le plan défini par la face (Voir figure 3(b)). On dit dans ce cas que la surface résultante S'' est obtenue par l'ajout à S d'une *anse tordue*.

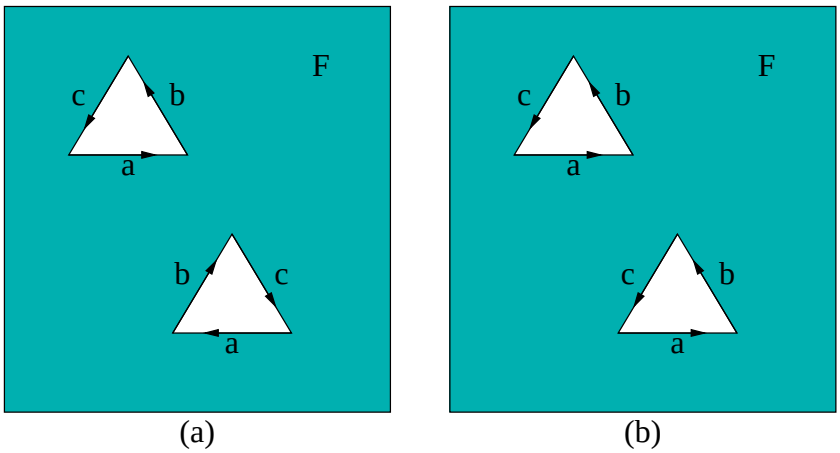


Figure 3 – Ajouter une anse ou une anse tordue.

On considère maintenant toutes les surfaces obtenues de la sphère $\mathbb{S}_0$ (vu comme le tétraèdre) par l'ajout d'anses et d'anses tordues. Par l'ajout de g anses on obtient la surface $\mathbb{S}_g$ appelée *surface orientable de genre* $\hat{g}$ et par l'ajout de $\hat{g}$ anses tordues on obtient la surface $\mathbb{N}_g$ appelée *surface non-orientable de genre* $\hat{g}$. Les surfaces $\mathbb{N}_{\hat{g}}$ sont dite non-orientables car il existe dans ces surfaces des courbes closes simples telles que la droite et la gauche s'interchangent le long de la courbe. Les surfaces orientables n'ont quant à elles aucune courbe ayant cette propriété. Les surfaces $\mathbb{S}_0$, $\mathbb{S}_1$, $\mathbb{S}_2$, $\mathbb{N}_1$ et $\mathbb{N}_2$ sont connues sous les noms respectifs de sphère, de tore, de double tore, de plan projectif et de bouteille de Klein. On appelle triangulations de genre orientable g et triangulations de genre non-orientable $\hat{g}$ les triangulations de $\mathbb{S}_g$ et $\mathbb{N}_{\hat{g}}$.

Un des résultats les plus importants de la théorie des graphes sur surface nous dit qu'il suffit de considérer les surfaces $\mathbb{S}_g$ et $\mathbb{N}_{\hat{g}}$ pour considérer toutes les surfaces à homéomorphisme près.

Théorème 1 ([91]). *(Classification des surfaces) Chaque surface est homéomorphe à exactement une des surfaces* $\mathbb{S}_g$ *avec* $g \geqslant 0$ *ou* $\mathbb{N}_{\hat{g}}$ *avec* $\hat{g} \geqslant 1$.

On définit le *genre orientable* et le *genre non-orientable* d'un graphe G connexe comme

étant respectivement le nombre minimum $g(G)$ et $\hat{g}(G)$ tel que G puisse être plongé respectivement sur $\mathbb{S}_{g(G)}$ et $\mathbb{N}_{\hat{g}(G)}$. On définit le genre d'un graphe non-connexe comme étant la somme des genres de ses composantes connexes.

La *caractéristique d'Euler* $\chi(S)$ d'une surface S est définie comme étant :

$$\chi(S) = \begin{cases} 2-2g & \text{si } S = \mathbb{S}_g \\ 2-\hat{g} & \text{si } S = \mathbb{N}_{\hat{g}} \end{cases}$$

Dans certain cas, il est utile de considérer le *genre d'Euler* d'un graphe G noté $\text{eg}(G)$ qui est égal à $\min_{S\in\mathcal{S}} \{2-\chi(S)\}$ avec $\mathcal{S}$ l'ensemble des surfaces dans lesquelles G peut être plongé.

Une des formules les plus importantes des plongements de graphes sur surfaces est la formule d'Euler qui nous donne une relation entre les nombres de sommets, d'arêtes et de faces du plongement en fonction de la caractéristique d'Euler de la surface. On rappelle qu'un plongement de graphe est dit cellulaire si chaque face du plongement est homéomorphe au disque unité dans le plan.

Propriété 1. *(Formule d'Euler) Soit G un graphe plongé de manière cellulaire dans la surface S. Si G a n sommets, m arêtes et f faces alors*

$$n - m + f = \chi(S).$$

Remarque 2. *Si G est simple alors*

$$m \leqslant 3n - 3\chi(S).$$

Démonstration. Puisque G est simple, chaque face a pour taille au moins 3. De plus, chaque arête est au plus dans deux faces. On en déduit que :

$$\begin{aligned} 2m &\geqslant 3f \\ n - m + \frac{2}{3}m &\geqslant \chi(S) \\ m &\leqslant 3n - 3\chi(S). \end{aligned}$$

□

Soit G une triangulation d'une surface $\mathbb{S}_g$ et soit C un cycle de G. Le cycle C est dit *séparant* s'il sépare la surface $\mathbb{S}_g$ en deux composantes connexes par arcs. Un cycle séparant est dit *contractible* si l'une des deux composantes connexes de $\mathbb{S}_g - C$ est homéomorphe au disque unité dans le plan.

1.3.3 Carte combinatoire

Soit G un graphe multiple connexe dans une surface S orientable. On se donne dans S une orientation locale dans le sens trigonométrique. Pour un sommet v de $V(G)$, la *rotation*

locale en v noté π_v est la relation de succession des arêtes incidentes à v autour du sommet telle que $e_2 = \pi_v(e_1)$ si e_2 est l'arête succédant e_1 dans le sens trigonométrique autour de v.

Définition 2. *Le* système de rotation *du plongement d'un graphe G dans une surface S orientable est l'ensemble $\pi = \{\pi_v \mid v \in V(G)\}$. Une* carte combinatoire *est le couple $< G, \pi >$.*

Le théorème suivant nous permet de considérer les plongements de graphe de manière combinatoire.

Théorème 2. *Chaque plongement cellulaire d'un graphe G dans une surface orientable S est uniquement déterminé, à homéomorphisme près, par son système de rotation.*

Une suite $W = v_0 e_1 v_1 e_1 \cdots e_k v_k$ de sommets v_i et d'arêtes e_i de G tel que $e_i = v_{i-1}v_i$ ($i \leqslant k$) et $v_0 = v_k$ est appelé *cheminement* de G. On appelle un π*-cheminement* d'une carte combinatoire, un cheminement obtenu comme suit : on part d'un sommet arbitraire v et d'une arête uv incidente à v, on parcourt l'arête e et on continue le cheminement en traversant l'arête $e' = \pi_v(e)$. Les π-cheminements délimitent les faces du plongement du graphe correspondant. En effet, chaque partie connexe par arcs de $S \setminus G$ d'un plongement G sur la surface S est délimitée par des arêtes formant un π-cheminement dans la carte correspondante. Chaque arête d'une carte combinatoire est contenue soit une seule fois dans exactement deux π-cheminements, soit deux fois dans un seul π-cheminement. On appelle ce dernier type d'arête des arêtes *singulières*.

Chapitre 2

Partition des arêtes des graphes de genre borné

2.1 Introduction

En algorithmique des graphes, une technique très employée consiste à "diviser pour régner". En effet, il est souvent utile de diviser une instance d'un problème en plusieurs sous-instances plus simples ou de taille moindre, de traiter chaque sous-instance séparément et enfin de combiner les solutions obtenues pour chaque sous-instance afin d'obtenir une solution pour l'instance de départ. Dans cette optique de simplification, il est souvent nécessaire que les sous-graphes induits par les parties aient des propriétés spécifiques. Cette technique est très utile et notamment pour la représentation distribuée de graphes comme nous le verrons dans la deuxième partie.

Les partitions des arêtes de graphes en forêts ou arbres font partie des types de partitions les plus étudiées. Dans ce chapitre, nous nous sommes intéressés à de telles partitions pour les graphes sur surface. L'un des premiers résultats connus sur les partitions en forêts est une caractérisation des graphes pouvant être partitionnés en k forêts dits d'*arboricité* k, suivant le ratio du nombre d'arêtes sur le nombre de sommets de leurs sous-graphes [72]. En utilisant la formule d'Euler, il est possible d'obtenir en tant que résultat préliminaire un majorant en $O(\sqrt{g})$ pour l'arboricité des graphes de genre d'Euler g.

Un autre paramètre intéressant lié aux partitions de graphes en forêts est la *densité arborescente* de graphe, c'est-à-dire le nombre maximum d'arbres couvrants arête-disjoints contenus dans le graphe. En effet, si un graphe de n sommets et m arêtes a une densité arborescente k alors on peut partitionner ses arêtes en k arbres plus un ensemble d'au plus $m - k(n-1)$ arêtes. En plus de son utilisation pour les partitions d'arêtes, le problème de recherche du nombre maximal d'arbres couvrants apparaît dans la construction de schémas de routage efficaces dans les réseaux à routage "wormhole". Dans de tels réseaux [74], le message est d'abord segmenté en unité d'information de taille constante. Ensuite, chaque sommet intermédiaire transmet l'unité d'information à son voisin conformément au chemin

utilisé par la première partie du ver appelée tête du ver. Bien qu'un arbre couvrant soit suffisant pour construire un algorithme de routage sans blocage [69], il est clair que pouvoir obtenir plusieurs arbres couvrants arêtes-disjoints permet de réaliser dans le réseau autant de multicast simultanés que d'arbres. On prouve que les triangulations de genre d'Euler $g > 0$ ont une densité arborescente supérieure ou égale à 3. En fait, ce résultat est une généralisation d'un résultat de Kundu portant sur les graphes toriques [68]. On en déduit une partition des graphes de genre d'Euler g en trois forêts plus un ensemble d'au plus $3g - 3$ arêtes.

Il est possible de calculer cette partition en temps $O(n^2)$ pour un graphe de n sommets en utilisant un algorithme de Roskind et Tarjan [84]. Dans le cas des graphes planaires cette partition en trois forêts peut se faire en temps $O(n)$ grâce au réaliseurs de Schnyder [86]. On peut se demander s'il n'existe pas un algorithme de partition en temps linéaire pour un graphe de genre d'Euler fixé. On décrit un tel algorithme pour le cas des graphes toriques dans la deuxième partie du chapitre.

Dans la troisième partie du chapitre, on présente quelques généralisations possibles des partitions connues pour les graphes planaires. Les partitions que l'on considère sont une partition des graphes de genre g en trois g-forêts (généralisation des forêts au genre g) et une partition des graphes de genre g en deux graphes g-extérieurs (généralisation des graphes planaires extérieurs au genre g). On conclut finalement le chapitre par une généralisation possible des réaliseurs de Schnyder pour les graphes de genre g.

Ces résultats ont fait l'objet d'une présentation à une conférence internationale [19].

2.2 Arboricité des graphes de genre g

Dans cette partie, on s'intéresse à l'arboricité des graphes de genre borné. L'arboricité d'un graphe se définit comme suit :

Définition 3. *L'*arboricité *d'un graphe G notée $a(G)$ est égale à :*

$$a(G) = \max_{H \in A} \left\lceil \frac{|E(H)|}{|V(H)| - 1} \right\rceil$$

où A est l'ensemble de tous les sous-graphes induits de G qui contiennent au moins 2 sommets.

En fait, l'arboricité est plus connue comme étant le nombre de parties minimum dans une partition d'arêtes d'un graphe en forêts. Le théorème de Nash-Williams [72] nous indique que ses deux notions sont en fait équivalentes. L'une des méthodes les plus faciles pour partitionner les arêtes des graphes de genre g consiste donc simplement à majorer l'arboricité de ces graphes.

Fait 1. *Soit G un graphe de genre d'Euler $g > 0$. On a :*

$$a(G) \leqslant \left\lceil \frac{7 + \sqrt{24g + 1}}{4} \right\rceil .$$

Démonstration. Il nous suffit de borner la fraction $\frac{|E(H)|}{|V(H)|-1}$ pour tout graphe H de genre d'Euler g ayant n sommets et m arêtes. On pose $k(g) = \frac{7+\sqrt{24g+1}}{2}$. On cherche à montrer que l'arboricité des graphes de genre d'Euler g est au plus $\lceil k(g)/2 \rceil$. On considère deux cas possibles pour le graphe H :

- **Cas 1 :** $n \leqslant k(g)$

  Clairement, le graphe H est donc un sous-graphe de $K_{k(g)}$. L'arboricité de $K_{k(g)}$ est égale à $\lceil k(g)/2 \rceil$ car pour n'importe entier $r \geqslant 2$, on a :

$$\frac{|E(K_r)|}{|V(K_r)|-1} = \frac{r(r-1)}{2(r-1)} = \frac{r}{2}$$

- **Cas 2 :** $n > k(g)$

  D'après la Remarque 2, H a au plus $3n + 3g - 6$ arêtes. On a :

$$\begin{aligned}
\left\lceil \frac{m}{n-1} \right\rceil &\leqslant \left\lceil \frac{3n+3g-6}{n-1} \right\rceil \\
&\leqslant 3 + \left\lceil \frac{3g-3}{k(g)} \right\rceil \\
&\leqslant 3 + \left\lceil \frac{3g-3}{\frac{7+\sqrt{24g+1}}{2}} \right\rceil \\
&\leqslant \left\lceil \frac{3(5+\sqrt{24g+1})+6g-6}{5+\sqrt{24g+1}} \right\rceil \\
&\leqslant \left\lceil \frac{9+3\sqrt{24g+1}+6g}{5+\sqrt{24g+1}} \right\rceil \\
&\leqslant \left\lceil \frac{(9+3\sqrt{24g+1}+6g)(5-\sqrt{24g+1})}{(5+\sqrt{24g+1})(5-\sqrt{24g+1})} \right\rceil \\
&\leqslant \left\lceil \frac{42-42g+6\sqrt{24g+1}-6g\sqrt{24g+1}}{24(1-g)} \right\rceil \\
&\leqslant \left\lceil \frac{7+\sqrt{24g+1}}{4} \right\rceil
\end{aligned}$$

□

On peut remarquer que ce majorant est presque optimal puisque pour $n = \lfloor k(g) \rfloor$, le graphe K_n a pour genre d'Euler g d'après [82] or K_n a pour arboricité $\lceil \frac{n}{2} \rceil = \left\lceil \frac{1}{2} \left\lfloor \frac{k(g)}{2} \right\rfloor \right\rceil$.

2.3 Densité arborescente des graphes de genre borné

Les graphes de genre d'Euler g ont donc une arboricité en $O(\sqrt{g})$. Néanmoins, pour g fixé, lorsque n croit le ratio du nombre d'arêtes sur le nombre de sommets pour les graphes de genre d'Euler g ayant n sommets tend vers 3 d'après la remarque 2. Il est donc légitime de se demander si l'on peut partitionner les graphes de genre d'Euler g en trois forêts plus un ensemble d'arêtes dont la taille dépendrait de g. La réponse à cette question est positive et la preuve utilise la notion de densité arborescente.

2.3.1 Densité arborescente

La *densité arborescente* d'un graphe G notée $t(G)$ est le nombre maximum d'arbres couvrants arêtes-disjoints contenus dans G. Soit $P = \{P_1, P_2, \cdots, P_r\}$ une partition des sommets d'un graphe G. Le *graphe de contraction* G selon P notée G_P est un graphe multiple dont les sommets sont les parties de la partition P de $V(G)$ et dont l'ensemble d'arêtes est $E(G_P) = \{(P_i, P_j) \mid \exists u \in P_i, v \in P_j, (u, v) \in E(G)\}$.

L'un des premiers résultats connus pour la densité arborescente est du à Tutte [93] et Nash-Williams [73]. Il nous donne un minorant pour la densité arborescente d'un graphe en fonction de ses graphes de contraction.

Théorème 3 ([93, 73])**.** *Un graphe G a une densité arborescente supérieure ou égale à k si et seulement si pour toute partition P de $V(G)$ on a :*

$$|E(G_P)| \geqslant k(|V(G_P)| - 1).$$

Le deuxième résultat important pour la densité arborescente est celui de Kundu [68] qui est relié à la k-arête-connectivité du graphe.

Théorème 4 ([68])**.** *Soit G un graphe k-arête-connexe mais non $(k+1)$-arête-connexe. On a :*

$$\left\lfloor \frac{k}{2} \right\rfloor \leqslant t(G) \leqslant k.$$

Pour le cas des triangulations de genre g, il n'existe que deux résultats. Les triangulations du plan ont une densité arborescente égale à deux et les triangulations du tore ont une densité arborescente égale à trois [68]. Puisque les triangulations de genre supérieur ont une arboricité supérieure, il paraît logique de penser qu'elles ont de même une densité arborescente supérieure.

2.3.2 Densité arborescente pour les triangulations de genre d'Euler $g > 1$

L'extension du résultat de Kundu pour les triangulations genre d'Euler $g > 0$ est possible. Dans ce paragraphe, nous allons montrer le théorème suivant.

Théorème 5. *Soit G une triangulation de genre d'Euler $g > 0$. On a :*

$$t(G) \geqslant 3.$$

Il est important de remarquer que ce minorant ne peut être amélioré. En effet pour tout $g \geqslant 0$, on peut construire une triangulation de genre d'Euler g qui n'est pas 4-arête-connexe et donc a densité arborescente strictement inférieure à 4 d'après le théorème 4. Il suffit pour cela de prendre une triangulation quelconque et de rajouter un sommet à l'intérieur d'un triangle en le reliant aux trois sommets du triangle. On peut aussi remarquer que pour une triangulation G de genre d'Euler g si $|V(G)| > 3g - 2$ alors $t(G) \leqslant 3$. En effet, G a $3|V(G)| + 3g - 6 < 4(|V(G)| - 1)$ arêtes d'après la formule d'Euler. Donc, G ne peut avoir 4 arbres couvrants arêtes-disjoints puisque chaque arbre couvrant requiert $|V(G)| - 1$ arêtes.

D'après le théorème 5 et la formule d'Euler, on peut obtenir une nouvelle partition des triangulations de genre d'Euler g. Cette partition sera utilisée dans le troisième chapitre afin d'obtenir une représentation distribuée des graphes de genre borné plus compacte que celle déduite de l'arboricité.

Corollaire 1. *Soit G un graphe de genre d'Euler g. Il existe une partition d'arêtes de G en 4 parties A_0, A_1, A_2 et A_3 tel que $|A_0| \leqslant 3g - 3$ et $\forall i \in \{1, 2, 3\}$, A_i induit une forêt.*

Démonstration. Il suffit de plonger le graphe G sur la surface appropriée de caractéristique d'Euler g et de le trianguler sans créer d'arêtes multiples ou de boucles obtenant ainsi un graphe G'. On applique le théorème 5 sur G' obtenant trois arbres couvrants. Ces trois arbres couvrants vont nous donner les trois parties A_1, A_2 et A_3. Les arêtes restantes, au nombre d'au plus $3g - 3$ d'après la formule d'Euler, forment la partie A_0. □

On a besoin du lemme suivant pour prouver le théorème 5.

Lemme 1. *Soit G_P un graphe de contraction de G. Soit $H_1, H_2, \cdots, H_r$ les sous-graphes de G induit par les parties $P_1, P_2, \cdots, P_r$ de $V(P)$. On a :*

$$\sum_{i=1}^{r} \mathrm{eg}(H_i) \leqslant \mathrm{eg}(G).$$

Démonstration. On calcule un arbre couvrant T de G_P. On enlève dans le graphe G toutes les arêtes correspondantes aux arêtes de $E(T)$ obtenant ainsi le sous-graphe G'. Le graphe G' peut être reconstruit en partant du sous-graphe H_1 en reconnectant un par un les sous-graphes H_i par le biais des arêtes de $E(T)$. L'union de deux graphes G_1 et G_2 ayant un sommet en commun a pour genre d'Euler $\mathrm{eg}(G_1) + \mathrm{eg}(G_2)$ [11, 89]. Puisque lors de notre reconstruction de G', on fait l'union des graphes H_i par exactement un seul sommet on en déduit que :

$$\begin{aligned}\sum_{i=1}^{r} \mathrm{eg}(H_i) &= \mathrm{eg}(G') \\ &\leqslant \mathrm{eg}(G) \\ &\quad \text{puisque } G' \text{ est un sous-graphe de } G.\end{aligned}$$

□

Démonstration du théorème 5. On considère G une triangulation de caractéristique d'Euler g. Soit $P = \{P_1, P_2, \cdots, P_r\}$ une partition de $V(G)$ telle que $|P_1| \geqslant |P_2| \geqslant \cdots \geqslant |P_r|$ et G_P le graphe de contraction de G correspondant. On note $n = |V(G)|$, $m = |E(G)|$, $g = \mathrm{eg}(G)$, $n_i = |V(H_i)|$, $m_i = |E(H_i)|$ et $g_i = \mathrm{eg}(H_i)$. D'après la formule d'Euler, on a $m = 3n + 3g - 6$. On considère trois type de parties P_i dans P :

- Les parties P_i de taille au moins 3 : $1 \leqslant i \leqslant k$, $n_i \geqslant 3 \Rightarrow m_i \leqslant 3n_i + 6(g_i - 1)$
- Les parties P_i de taille 2 : $k+1 \leqslant i \leqslant k+q$, $n_i = 2 \Rightarrow m_i = 1$
- Les parties P_i de taille 1 : $k+q+1 \leqslant i \leqslant r$, $n_i = 1 \Rightarrow m_i = 0$

Par construction, on a $|E(G_p)| = m - \sum_{i=1}^{r} m_i$. On obtient :

$$\begin{aligned}|E(G_p)| &\geqslant 3n + 3g - 6 - \left(\sum_{i \leqslant k}(3n_i + 3g_i - 6) + q\right) \\ |E(G_P)| - 3(r-1) &\geqslant 2q + 3(k-1) + 3\left\{g - \sum_{i \leqslant k} g_i\right\}\end{aligned}$$

qui est positif ou nul. En effet, dans le cas où $k = 0$, on a $\sum_{i \leqslant k} g_i = 0$ puisque toutes les parts sont de taille inférieure ou égale à deux et donc de genre 0. On obtient donc que $g - \sum_{i \leqslant k} g_i > 0$ et que $|E(G_P)| - 3(r-1) \geqslant 0$. Dans le cas où $k > 0$, on sait d'après le lemme 1 que $g - \sum_{i \leqslant k} g_i \geqslant 0$. Par conséquent, on a bien $|E(G_P)| - 3(r-1) \geqslant 0$. □

2.3.3 Caractérisation de l'ensemble d'arêtes A_0

D'après le corollaire 1, pour toute triangulation de genre g il existe un ensemble A_0 d'arêtes tel que $G - A_0$ soit d'arboricité 3. Si le graphe G a n sommets alors on peut résoudre ce problème en temps $O(n^2)$ en utilisant l'algorithme de Roskind et Tarjan [84]. En effet, cet algorithme calcule k arbres couvrants arête-disjoints en temps $O(k^2n^2)$. Pour le cas particulier des graphes sur surfaces, on peut se demander s'il est possible d'implémenter un algorithme plus performant. Dans ce but, il peut être utile d'avoir une caractérisation de l'ensemble d'arêtes A_0 puisque calculer l'ensemble d'arêtes A_0 semble être la clé du problème de partitionnement. Le théorème 6 nous donne une caractérisation de cet ensemble de sommets.

On rappelle qu'une région est un ensemble connexe par arcs d'une surface. Soit G un graphe plongé sur une surface S et $w_1, \cdots, w_r$ des cheminements de G. Une *région R délimitée* par des cheminements $w_1, \cdots, w_r$ de G est une composante de $S \backslash \{w_1, \cdots, w_r\}$ tel que tous les voisinages des sommets des cheminements $w_1, \cdots, w_r$ contiennent des éléments de R. La figure 4 nous donne un exemple de région délimitée par deux cheminements w_1 et w_2 (partie foncée de la figure) et de région non-délimitée par un cheminement w_1 (partie claire de la figure).

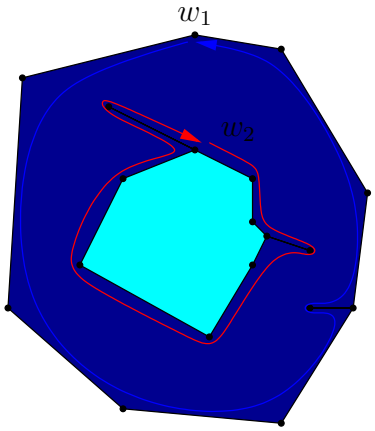


Figure 4 – Exemple de régions délimitées par des cheminements.

Une arête e est dite *contenue dans une région R* si la courbe de e moins ses extrémités est contenue dans R. On définit le *sous-graphe $G[R]$ induit par une région R* comme étant le sous-graphe induit par l'ensemble des arêtes contenue dans R.

Théorème 6. *Soit G une triangulation de genre d'Euler g avec $g \geqslant 0$ et A_0 un ensemble de $3g-3$ arêtes de G. L'arboricité de $G - A_0$ est 3 si et seulement si pour chaque ensemble de régions $R = \{R_1, \cdots, R_k\}$ deux à deux disjointes et délimitées par des cheminements $W = \{w_1, \cdots, w_l\}$ de G, $G[R]$ ne contient pas plus de $3(g-\mathrm{eg}(G \backslash G[R]))+6(r-1)+\sum_{i=1}^{i=l}(|w_i|-3)$ arêtes de A_0 où r est le nombre de composantes connexes de $G \setminus G[R]$.*

Démonstration. Soit G un graphe plongé sur S une surface de caractéristique d'Euler g. Premièrement, on montre que si chaque ensemble de régions de S respecte la condition du théorème, alors $a(G \setminus A) = 3$. D'après la définition 3, il nous suffit de montrer que pour chaque sous-graphe H de G, on a $|E(H - A)| \leqslant 3(|V(H - A)| - 1)$. Puisque H est un sous-graphe de G, le plongement de H sur S déduit de celui de G n'est pas nécessairement cellulaire. On considère $F = \{F_1, F_2, \cdots, F_f\}$, l'ensemble des faces de G délimitées par un ensemble de π-cheminements P de G, $P = \{p_1, \cdots, p_q\}$. Pour chaque face F_i qui est non cellulaire dans H, on découpe la surface S le long de la courbe formée par les π-cheminements de F_i, on enlève la région correspondant à l'intérieur de la face et on remplace cette région par une région cellulaire. Par ce processus, on obtient des plongements cellulaires des composantes connexes $H_1, H_2, \cdots, H_r$ de H sur des surfaces

$S_1, S_2, \cdots, S_r$. Pour tout $i = 1, 2, \cdots r$, on considère $F_i = \{F_{i,1}, \cdots, F_{i,f_i}\}$ l'ensemble des faces du plongement de H_i. D'après la formule d'Euler sur le plongement H_i, on obtient :

$$\forall i = 1 \cdots r, |E(H_i)| = |V(H_i)| + f_i + \text{eg}(H_i) - 2 \quad (1)$$

On rappelle que chaque arête d'une carte est soit une fois dans deux π-cheminements, soit deux fois dans un seul π-cheminement. On a donc :

$$\forall i = 1 \cdots r, 2|E(H_i)| = \sum_{j=1}^{f_i} |F_{i,j}| \quad (2)$$

En combinant les Équations 1 et 2, on obtient :

$$\begin{aligned} \forall i = 1 \cdots r, |E(H_i)| &= 3|V(H_i)| + 3f_i - 2|E(H_i)| + 3\text{eg}(H_i) - 6 \\ \forall i = 1 \cdots r, |E(H_i)| &= 3|V(H_i)| - \sum_{j=1}^{f_i}(|F_{i,j}| - 3) + 3\text{eg}(H_i) - 6 \end{aligned}$$

Pour H, il s'en suit que :

$$\begin{aligned} |E(H)| &= \sum_{i=1}^{r} |E(H_i)| \\ |E(H)| &= 3|V(H)| - \sum_{i=1}^{r}\sum_{j=1}^{j=f_i}(|F_{i,j}| - 3) - 6r + 3\sum_{i=1}^{i=r} \text{eg}(H_i) \end{aligned}$$

Pour $i = 1, \cdots, f$, on considère R l'ensemble des régions de S délimitées par les cheminements de P qui contiennent ensembles tous les sommets de $V(G)\backslash V(H)$. D'après la condition du théorème, il ne peut y avoir plus de $3(\text{eg}(G) - \text{eg}(G \backslash G[R])) + 6(r-1) + \sum_{j=1}^{j=q}(|p_j| - 3)$ arêtes de A_0 dans $G[R]$.

On obtient donc :

$$\begin{aligned} |E(H \cap A_0)| &= |E(A)| - |E(A_0 \setminus H)| \\ |E(H \cap A_0)| &\geqslant (3\text{eg}(G) - 3) - 3(\text{eg}(G) - \text{eg}(G \setminus G[R])) + 6(r-1) + \sum_{j=1}^{j=q}(|p_j| - 3) \\ |E(H \cap A_0)| &\geqslant 3\text{eg}(G \setminus G[R]) - 6r + 3 - \sum_{j=1}^{j=q}(|p_j| - 3) \end{aligned}$$

Par construction des plongements cellulaires des H_i, on a :

$$\sum_{i=1}^{i=s}\sum_{j=1}^{j=f_i}(|F_{i,j}|-3) \;=\; \sum_{j=1}^{j=q}(|p_j|-3)$$

De plus, puisque $G \setminus G[R] = \bigcup_{i=1}^{r} H_i$, on a :

$$\bigcup_{i=1}^{i=r}\mathrm{eg}(H_i) \;=\; \mathrm{eg}(G \setminus G[R])$$

On obtient que :

$$|E(H \cap A_0)| \;\geqslant\; 3\bigcup_{i=1}^{i=r}\mathrm{eg}(H_i) - 6r + 3 - \sum_{i=1}^{i=r}\sum_{j=1}^{j=f_i}(|F_{i,j}|-3)$$

Pour $H - A$, cela nous donne :

$$\begin{aligned} |E(H - A_0)| &= |E(H)| - |E(H \cap A_0)| \\ |E(H - A_0)| &\leqslant 3|V(H - A_0)| - 3 \end{aligned}$$

Et donc l'arboricité de G est 3.

Deuxièmement, on montre que si un ensemble de régions $R = \{R_1, \cdots, R_k\}$ deux à deux disjointes et délimitées par des cheminements $W = \{w_1, \cdots, w_l\}$ de G contient plus de $3(\mathrm{eg}(G) - \mathrm{eg}(G \setminus G[R])) + 6(r-1) + \sum_{i=1}^{i=k}(|w_i|-3)$ arêtes de A_0 alors l'arboricité de G est supérieur à 3. On considère le sous-graphe $H = G \setminus G[R]$. On obtient en utilisant les mêmes notations que la première partie de la preuve les relations suivantes.

$$|E(H)| \;=\; 3|V(H)| - \sum_{i=1}^{r}\sum_{j=1}^{j=f_i}(|F_{i,j}|-3) - 6r + 3\sum_{i=1}^{i=r}\mathrm{eg}(H_i)$$

$$|E(H \cap A_0)| \;<\; 3\mathrm{eg}(G \setminus G[R]) - 6r + 3 - \sum_{i=1}^{i=r}\sum_{j=1}^{j=f_i}(|F_{i,j}|-3)$$

$$\begin{aligned} |E(H - A)| &= |E(H)| - |E(H \cap A)| \\ |E(H - A)| &> 3|V(H - A)| - 3 \end{aligned}$$

Finalement, l'arboricité de G est strictement supérieure à 3. □

Cette caractérisation se simplifie énormément dans le cas des graphes toriques.

Corollaire 2. *Soit G une triangulation du tore et A_0 un ensemble de 3 arêtes de G. L'arboricité de $G \setminus A_0$ est 3 si et seulement si pour chaque cheminement contractible w de longueur $k \leqslant 5$ de G, la région cellulaire délimitée par w contient au plus $k-3$ arêtes de A_0.*

Démonstration. Il suffit de remarquer que dans ce cas, $|A_0| = 3$. Par conséquent, les régions séparant le graphe ou diminuant son genre n'imposent aucune contrainte sur les arêtes de A_0 car de telles régions peuvent contenir jusqu'à 6 arêtes de A_0 d'après la condition énoncée par le théorème 6. Les seules régions imposant des contraintes sur A_0 sont les régions contractibles et il est clair que l'on peut considérer que celles délimitées par un seul cheminement. En effet, pour toute région contractible R délimitée par plusieurs cheminements $w_1, \cdots, w_k$, il existe toujours une région contractible délimitée par un seul cheminement w_i qui contient R. □

2.4 Algorithme de partition en forêts pour les triangulations toriques

D'après le corollaire 1, il est possible de partitionner les arêtes d'un graphe torique (de genre orientable 1) de n sommets en 3 forêts plus un ensemble d'au plus 3 arêtes. Comme on l'a indiqué précédemment, il est possible de construire cette partition en temps $O(n^2)$ en utilisant l'algorithme de Roskind et Tarjan [84]. Dans le cas des graphes planaires cette partition en trois forêts peut se faire en temps $O(n)$ grâce au réaliseurs de Schnyder [86]. En combinant les réaliseurs de Schnyder avec une partition particulière des graphes toriques, il est possible d'obtenir un algorithme de partitionnement des graphes toriques en temps $O(n)$.

Théorème 7. *Les arêtes d'un graphe torique de n sommets peuvent être partitionnées en temps $O(n)$ en trois forêts plus un ensemble d'au plus trois arêtes.*

2.4.1 Principe de l'algorithme

On considère un graphe torique G de n sommets. Pour le reste de l'algorithme on a besoin d'une triangulation torique. Le plongement du graphe sur le tore se calcule en temps $O(n)$ [57]. Obtenir une triangulation du tore à partir d'un graphe torique n'est pas aussi évident que dans le cas planaire. Il convient de se montrer prudent afin de ne pas créer d'arêtes multiples car un sommet peut apparaître jusqu'à quatre fois dans une même face. On triangule le plongement obtenu en temps $O(n)$ d'après la propriété suivante.

Propriété 2. *Pour chaque graphe simple G de genre g, il existe une triangulation de $\mathbb{S}_g$ avec $O(n+g)$ sommets qui a G comme sous-graphe. Cette triangulation peut être calculée en temps $O(n+g)$.*

Démonstration. Il suffit de rajouter à l'intérieur de chaque face de G délimitée par un π-cheminement $y_1, \cdots, y_k$ un cycle C de k sommets $x_1, \cdots, x_k$, un sommet z relié à tous les sommets du cycle C et les arêtes $y_i x_i$ et $y_i x_{i+1 \mod k}$ pour $i = 1, \cdots, k$ (voir figure 5).

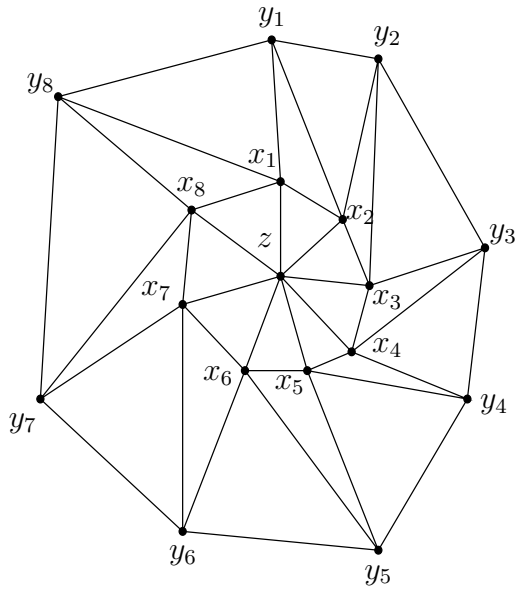


Figure 5 – Exemple de triangulation d'une face.

Le plongement obtenu ne contient bien que des triangles. On ne crée pas de boucles ni d'arête multiples car chaque sommet y_i est connecté qu'à deux sommets de G. Ces deux sommets sont les deux extrémités d'une même arête et donc ne peuvent être identiques puisque G est simple. Le nombre de sommets ajoutés est en $O(|E(G)|) = O(n+g)$ d'après la formule d'Euler. □

La construction de la partition de G comporte trois étapes principales.

– **Étape 1 :** Partition du graphe en un graphe planaire H plus un *tambourin* T.

  Afin d'utiliser les résultats connus sur les graphes planaires, on doit rendre G planaire. Pour cela, on enlève un sous-ensemble d'arêtes de $E(G)$ appelé tambourin T obtenant ainsi un graphe planaire H. La complexité de cette étape est due au fait que cet ensemble d'arêtes doit avoir des propriétés spécifiques afin de garantir la correction des deux étapes suivantes.

– **Étape 2 :** Partition d'arêtes de H en trois forêts F_1, F_2 et F_3 plus un ensemble d'au plus trois arêtes.

  C'est l'étape la plus difficile de l'algorithme. On doit trouver une partition de

H qui a certaines propriétés afin que l'on puisse ensuite obtenir la partition d'arêtes voulue pour G. On assure ces propriétés ainsi que l'efficacité de l'algorithme grâce au réaliseur de Schnyder.

– **Étape 3 :** partition d'arêtes de G en trois forêts F_1, F_2 et F_3 plus un ensemble d'au plus trois arêtes.

On utilise la partition de H calculée à l'étape précédente et on place chaque arête de T dans l'une des parties de la partition. Cette étape est la plus facile car les propriétés spécifiques de la planarisation de G et de la partition d'arêtes de H nous garantissent un ajout facile des arêtes du tambourin dans les parties de la partition.

2.4.2 Étape 1 : partition du graphe G en un graphe planaire H plus un tambourin T

Nous allons planariser le graphe G en un graphe H. Pour ce faire, un des outils à notre disposition est la notion de cycle non-contractible non-séparant. Dans le cas qui nous intéresse, les graphes de genre 1, les cycles séparants sont toujours contractibles. En effet, un cycle séparant coupe le graphe G en deux graphes dont la somme des genres est inférieure au genre de G [70]. Un cycle C séparant dans le tore sépare donc G en deux graphes dont l'un est planaire, donc C est contractible.

On définit l'action de *couper le long d'un cycle* C comme suit. Cela consiste à découper la surface le long du cycle C, de garder deux copies de C, nommées C' et C'' le long des deux bords obtenus et de coller une région cellulaire sur chacune afin d'obtenir une surface sans bords (voir figure 6 pour un exemple). On nomme G' le graphe ainsi obtenu. Un ensemble S de sommets de $V(G)$ est dit *séparant selon* C si chaque chemin dans G' d'un sommet de C' à C'' contient un sommet de S.

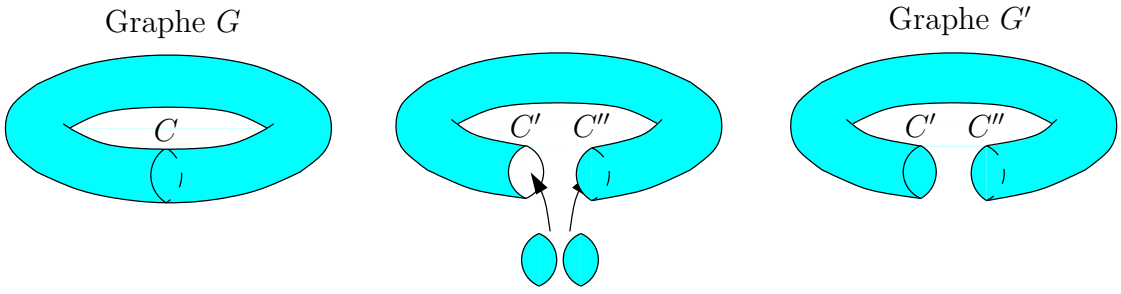


Figure 6 – Couper un graphe torique G le long d'un cycle C.

Une manière simple de planariser le graphe est de couper le long d'un cycle non-contractible non-séparant.

Propriété 3 ([3]). *Soit G un graphe plongé sur $\mathbb{S}_g$ avec $g > 0$ et soit C un cycle non-contractible non-séparant de G. Le plongement G' obtenu en découpant $\mathbb{S}_g$ le long de C est de genre $g - 1$.*

Il suffirait donc de supprimer toutes les arêtes incidentes aux sommets d'un cycle C non-contractible non-séparant pour rendre G planaire. Malheureusement, les étapes suivantes de l'algorithme nécessitent des propriétés spécifiques sur le graphe planaire obtenu et l'ensemble d'arêtes à enlever. Cette méthode de planarisation n'est donc pas utilisable dans notre cas.

On dit qu'une carte planaire simple est une *triangulation cylindrique* s'il existe deux faces F_1 et F_2, appelées les *embouts* de G, délimitées par des cycles sommet-disjoints, appelées les *cycles de la triangulation cylindrique* G, tel que toutes les faces de G à l'exception éventuelle de F_1 et F_2 soient de taille 3. On définit les arêtes *externes* d'une triangulation cylindrique comme étant les arêtes des deux cycles du tambourin. Par opposition, les arêtes *internes* sont les arêtes de la triangulation cylindrique qui ne sont pas externes. Un *tambourin* est l'ensemble des arêtes internes d'une triangulation cylindrique tel que chaque arête interne est incidente aux deux embouts.

Pour les besoins de notre algorithme, le graphe planaire obtenu par planarisation de G doit être une triangulation cylindrique et l'ensemble d'arêtes à enlever doit être un tambourin (voir figure 7).

Lemme 2. *Soit G une triangulation torique de n sommets. Il existe un tambourin T de G tel que $H = (V(G), E(G) \setminus T)$ soit une triangulation cylindrique. Un tel tambourin peut être calculé en temps $O(n)$.*

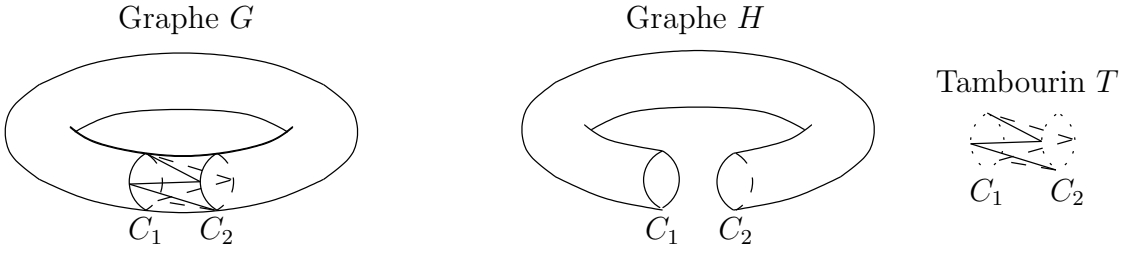


Figure 7 – Planarisation du graphe G.

On pourrait penser à utiliser le dual géométrique de G afin de trouver un tambourin puisque tous les tambourins dans une triangulation du tore correspondent à des cycles dans le dual géométrique de G. Malheureusement, l'inverse est faux et il semble difficile de calculer un tambourin à partir d'un cycle du dual géométrique.

Pour prouver le lemme 2, on a besoin de la propriété suivante.

Propriété 4 ([3]). *Soit G une triangulation de S_g ($g > 0$), C un cycle non-contractible non-séparant dans G et soit S un ensemble de sommets de G. Si S est séparant selon C,*

alors il existe un ensemble T de sommets tel que $T \subseteq S$ et le sous-graphe induit par T est un cycle non-contractible non-séparant.

Démonstration du lemme 2 On cherche à partitionner un graphe G torique en un tambourin et un graphe planaire.

On définit les arêtes *du coté gauche* et *du coté droit*, noté respectivement $E_g(C)$ et $E_d(C)$, d'un cycle $C = v_1, v_2, \cdots, v_l$ de G, comme suit. On note $e_i = v_i v_{i+1}$. Pour $i = 1 \cdots l$, si $e_{i+1} = \pi_{v_i}^{k_i}$ alors les arêtes $\pi_{v_i}(e_i), \pi_{v_i}^2(e_i), \cdots, \pi_{v_i}^{k_i-1}(e_i)$ sont du coté gauche de C. Les autres arêtes incidentes à v_i et qui ne font pas partie de C sont du coté droit de C. On définit *les voisins du coté droit* de C comme étant $V_d(C) = \{x \mid xy \in E_d(C) \text{ et } y \in C\}$. On définit *les voisins du coté gauche* de C noté $V_g(C)$ de manière analogue.

Le calcul du tambourin T s'effectue en trois temps :

1. On calcule un cycle non-contractible non-séparant C tel que $E_g(C) \cap E_d(C) = \varnothing$
2. A partir de C, on construit le cycle C_1 délimitant le premier embout du tambourin.
3. A partir de C_1, on construit le cycle C_2 délimitant le deuxième embout du tambourin.

On utilise tout d'abord l'algorithme linéaire de Djidjev et Venkatesan [36] afin d'obtenir un cycle non-contractible non-séparant C dans G. On calcule ensuite $E_g(C)$ et $E_d(C)$. Cette deuxième étape peut être réalisée en temps $O(|E(H)|) = O(n)$ grâce aux permutations associées à chaque sommet. La construction du tambourin nécessite un cycle C tel que $E_g(C) \cap E_d(C) = \varnothing$. En effet, sans cette condition, les deux cycles définiraient une triangulation cylindrique quelconque et non un tambourin.

Afin d'obtenir un tel cycle, on calcule tout d'abord l'ensemble d'arêtes $E_{g,d} = E_g(C) \cap E_d(C)$. Ensuite, on parcourt le cycle $C = x_1 x_2 \cdots x_r$ et à chaque fois que l'on rencontre une arête $x_i x_j$ de $E_{g,d}$, on remplace la partie $x_i x_{i+1} \cdots x_{j-1} x_j$ par l'arête $x_i x_j$ obtenant le cycle $x_1 \cdots x_i x_j \cdots x_r$.

Le cycle obtenu est non-contractible non-séparant. Il est non-séparant puisque l'arête va du coté gauche au coté droit du cycle et donc ne peut délimiter une région du tore. A la fin de l'algorithme, on obtient $E_g(C) \cap E_d(C) = \varnothing$. Grâce à un choix de structures de données appropriées (une liste pour l'encodage du cycle et un marqueur placé sur chaque arête de $E_{g,d}$ lors d'une étape de précalcul), cette partie de l'algorithme prends un temps $O(n)$.

On calcule ensuite l'ensemble $V_d(C)$. $V_d(C)$ est séparant selon C. Il existe donc un cycle C_1 non-contractible non-séparant $C_1 \subseteq V_d(C)$ d'après la Propriété 4. Le calcul du cycle C_1 commence par le calcul d'un premier sommet du cycle. On commence donc par déterminer un sommet x de C_1 de la façon suivante. On calcule un chemin P allant d'une extrémité d'une arête de $E_d(C)$ à une extrémité d'une arête de $E_g(C)$ et qui ne passe pas à travers le cycle C. Pour obtenir ce chemin, il suffit de couper G le long de C et de calculer un chemin d'un sommet de C' à C'' par un parcours en profondeur dans H. Le sommet x est défini comme étant le dernier sommet du chemin P qui est dans $V_d(C)$. Une fois ce premier sommet trouvé, les autres sommets du cycle C_1 sont faciles à calculer. Puisque $x \in V_d(C)$,

il existe z tel que $zx \in E_d(C)$. En commençant par l'arête xz, on parcourt la permutation cyclique associée au sommet x jusqu'à obtenir une arête $xy = \pi(xt)$ telle que $yt \in E_d(C)$. On choisi le sommet y comme deuxième sommet du cycle et on répète le processus sur y afin d'obtenir le troisième sommet du cycle et ainsi de suite.

Le cycle C_1 obtenu est tel que les deux extrémités de chaque arête de C_1 forment une face avec un sommet de C dans G. Clairement, $V(C_1)$ est séparant selon C, et donc C_1 est non-contractible and non-séparant. Afin d'obtenir le deuxième cycle C_2 du tambourin, on réapplique le processus sur C_1 mais en utilisant cette fois ci un chemin P allant d'une arête de $E_g(C_1)$ vers une arête de $E_d(C_1)$.

L'ensemble d'arêtes $T = E_d(C_1) = E_g(C_2)$ forme bien un tambourin car $V(C_1) \cap V(C_2) = \varnothing$ puisque $E_g(C_2) \cap E_d(C_2) = \varnothing$. Le graphe induit par l'ensemble des extrémités des arêtes de T est une triangulation cylindrique. Pour obtenir un plongement de cette triangulation sur la sphère, il suffit de découper le long des deux cycles C_1 et C_2 du coté correspondant aux arêtes du tambourin et de fermer la surface par l'ajout de deux disques. On obtient bien un graphe n'ayant que deux embouts délimitées par C_1 et C_2.

Par un choix de structures de données appropriées, les cycles C_1 et C_2 peuvent clairement être calculés en temps $O(m) = O(n)$. □

2.4.3 Étape 2 : partition de la triangulation cylindrique H

Il nous faut une partition de la triangulation cylindrique H ayant des propriétés spécifiques sur les sommets de ses faces non-triangulées afin d'obtenir la partition voulue pour G lors de l'ajout des arêtes de T à l'étape 3. Pour cela, on utilise un réaliseur de Schnyder. On définit le réaliseur de Schnyder en tant que partition d'arêtes puisque c'est la notion qui nous intéresse.

Définition 4 ([85]). *Soit H une triangulation du plan, soit xyz sa face externe et soit I l'ensemble des sommets internes du graphe ($I = V(G) - \{x, y, z\}$). Il est possible de calculer en temps $O(|V(H)|)$ une partition en* 3 *ensembles appelée* le réaliseur de Schnyder de G *telle que :*

- *T_1, T_2 et T_3 induisent des arbres arête-disjoints enracinés respectivement en x, y et z.*
- *Les voisins de chaque sommet $v \in I$ forment* 6 *blocs U_1, D_3, U_2, D_1, U_3 et D_2 dans le sens trigonométrique où U_j (respectivement D_j) est composé du père (respectivement des fils) de v dans T_j pour $j \in \{1, 2, 3\}$.*

On appelle la propriété nécessaire pour l'étape 3, *indépendance* et on l'a définit comme suit. Soit H un graphe, $A \subseteq E(H)$ et $W \subseteq V(H)$. Un sommet $x \in W$ est dit *indépendant selon* A et W si $V(H[A]) \cap W = \varnothing$. Soit H un graphe, $\mathcal{A} = \{A_1, \ldots, A_m\}$ une partition d'arêtes de H et $W \subseteq V(H)$. Le sommet $x \in W$ a pour *couleur manquante* $i = C_{\mathcal{A}}(x)$ selon $\mathcal{A}$ et W si x est indépendant selon A_i et W. L'ensemble W est dit *indépendant selon* $\mathcal{A}$ si $\forall x \in W$, x a une couleur manquante selon $\mathcal{A}$ et W.

Durant cette étape, on cherche donc à construire une partition $\mathcal{A} = \{A_1, A_2, A_3\}$ des arêtes en trois forêts de la triangulation cylindrique H telle que chaque sommet des deux embouts de H ait une couleur manquante. Le principe est que les couleurs manquantes des sommets des deux embouts vont permettre le rajout des arêtes du tambourin T dans la partition tout en gardant les graphes induits par les parties acycliques. En effet, puisqu'il n'existe pas de chemin dans A_i entre un sommet ayant comme couleur manquante i et n'importe quel autre sommet des deux embouts, on peut relier ces deux sommets par une arête dans A_i sans créer de cycle dans A_i. C'est ainsi que l'on va pouvoir placer les arêtes du tambourin T dans la partition et ainsi obtenir une partition de G à partir de celle de H.

Malheureusement, l'indépendance des sommets des deux embouts de la triangulation cylindrique H n'est pas suffisante pour obtenir la partition voulue de G lors de l'étape 3 de l'algorithme. En fait, on a besoin de deux partitions différentes de $E(H)$. On appelle 3-partitionnable la propriété requise et on la définit comme suit :

Définition 5. *Soit H une triangulation cylindrique ayant $F_1 = x_1, \cdots, x_r$ et $F_2 = y_1, \cdots, y_s$ comme embouts. On dit que H est* 3-partitionnable *s'il existe un ensemble A_0 de $k \leqslant 3$ arêtes de H, un ensemble $V_0 \subset V(F_1) \cup V(F_2)$ de $k-3$ sommets et deux partitions d'arêtes $\mathcal{A} = \{A_1, A_2, A_3\}$, $\mathcal{A}' = \{A'_1, A'_2, A'_3\}$ de $H' = H \setminus A_0$ tels que :*

- *Pour tout $i \in \{1, 2, 3\}$, $H[A_i]$ et $H[A'_i]$ sont des forêts.*
- *l'ensemble $V(F_1) \cup V(F_2) \setminus A_0$ est indépendant selon $\mathcal{A}$ et $\mathcal{A}'$ dans H'.*
- $\forall i, 1 \leqslant i \leqslant r, C_{\mathcal{A}}(x_i) \neq C_{\mathcal{A}'}(x_i)$
- $\forall i, 1 \leqslant i \leqslant s, C_{\mathcal{A}}(y_i) = C_{\mathcal{A}'}(y_i)$

Afin d'obtenir ces deux partitions, on commence par contracter les deux faces F_1 et F_2 de H. Une *contraction* d'une face F d'un carte H consiste à contracter tous les sommets au bord de F en un seul sommet x. La contraction de face peut créer des arêtes multiples. Deux arêtes sont dites *correspondantes* pour la face F si après de la contraction de F, elles correspondent à des arêtes multiples ayant les même extrémités. On remplace les arêtes multiples ainsi que tout sous-graphe compris entre deux arêtes multiples par une seule arête (voir figure 8). On appelle *sous-graphes contractibles de F* les sous-graphes délimités par deux arêtes correspondantes et une portion du bord de la face qui sont contenus entre deux arêtes multiples lors de la contraction de G. On les note $H_F(w)$ où w est le sommet incident aux deux arêtes correspondantes.

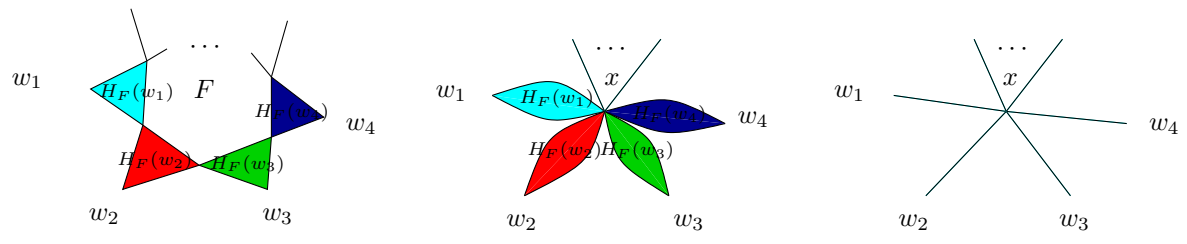


Figure 8 – Exemple de contraction d'une face F.

Avant de contracter les deux embouts de H, il est préférable de passer par une étape intermédiaire simplifiant le problème. Le graphe H_F obtenu par *simplification de la face* F de H est défini comme étant le graphe où l'on remplace chaque sous-graphe contractible $H_F(w)$ par des arêtes liant w aux sommets de $V(H_F(w)) \cap V(F)$. La figure 9 nous donne un exemple de simplification d'une face F.

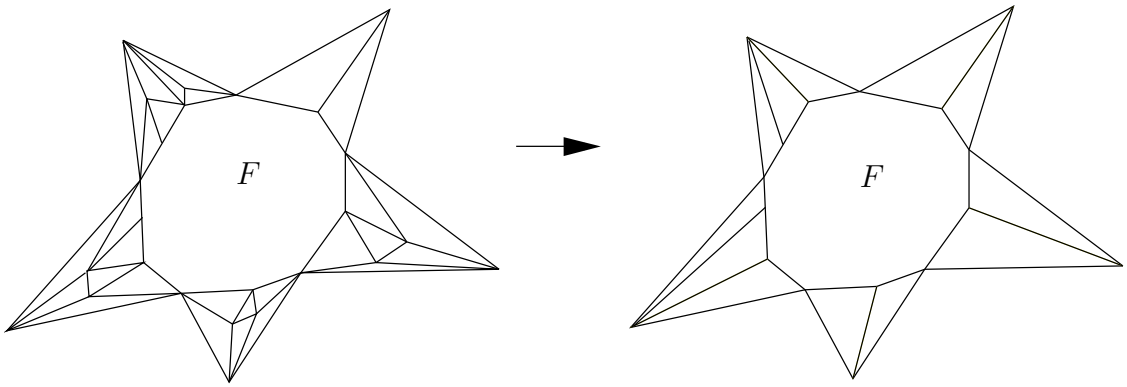


Figure 9 – Exemple de simplification d'une face F.

On dit qu'une face F est *simple* si le graphe issu de la simplification de F est égal au graphe de départ. On cherche à prouver que toutes les triangulations cylindriques sont 3-partitionnables. En fait, on peut se restreindre au cas des triangulations cylindriques dont les deux embouts sont simples.

Lemme 3. *Soit H une triangulation cylindrique de n sommets et H' le graphe obtenu par simplifications successives des embouts F_1 et F_2 de H. Si H' est 3-partitionnable et que pour $F = \{F_1, F_2\}$ et chaque sous-graphe contractible $H_F(w)$, les sommets de $V(H_F(w)) \cap V(F)$ ont la même couleur manquante et les arêtes correspondantes incidentes à w sont dans la même partie alors H est 3-partitionnable. De plus, les partitions d'arêtes de H peuvent être calculées en temps $O(n)$ à partir de celles de H'.*

Démonstration. Les seules différences entre H et H' se situe entre les sous-graphes contractibles $H_F(w)$ et $H'_F(w)$ pour $F = F_1, F_2$, $w \in V(H)$. Il nous suffit donc de décrire un algorithme qui partitionne $H_F(w)$ en fonction de la partition de $H'_F(w)$.

Le sous-graphe $H'_F(w)$ est composé d'un sommet w relié à des sommets $u_1, u_2, \cdots, u_r$ de $V(F)$. Le sous-graphe $H_F(w)$ est un graphe planaire dont toutes les faces sont des triangles à l'exception de sa face externe $wu_1, u_2, \cdots, u_r$. On considère la partition d'arêtes $\mathcal{A} = \{A_1, A_2, A_3\}$ d'arêtes de H'. Dans cette partition, il existe $c_1 \in \{1, 2, 3\}$ tel que les arêtes $wu_1, wu_2, \cdots, wu_r$ sont dans le même arbre F_{c_1}. De plus, Il existe $c_1 \in \{1, 2, 3\}$ avec $c_2 \neq c_1$ tel que les sommets $u_1, u_2, \cdots, u_r$ ont la même couleur manquante c_2. On note $c_3 \in \{1, 2, 3\}$ tel que $c_3 \neq c_1$ et $c_3 \neq c_2$. On triangule le graphe $H_F(w)$ en ajoutant un sommet y relié à tous les sommets de la face externe. On partitionne le graphe $H_F(x) \cup y$ grâce à un réaliseur de Schnyder en utilisant comme face externe la face xyu_1. On place

les arêtes de l'arbre du réaliseur enraciné en y plus l'arête xu_1 dans la partie F_{c_2}, les arêtes de l'arbre enraciné en x dans la partie F_{c_1} et les arêtes du troisième arbre dans F_{c_3}. On en déduit la partition de $H_F(x)$.

Il nous reste à vérifier que les partitions $\mathcal{A}$ et $\mathcal{A}'$ de $E(H)$ obtenues par ce biais respectent les conditions énoncées par la définition 5. Premièrement, l'indépendance des sommets des deux embouts est respectée. En effet, les sommets $u_1, u_2, \cdots, u_r$ ont pour couleur manquante c_2 puisque, par construction, les sommets $x, u_1, u_2, \cdots, u_r$ ne sont pas connectés entre eux dans F_{c_2}. Les sommets de $x, u_1, u_2, \cdots, u_r$ sont les fils de y dans l'arbre A_{c_2}. Par conséquent, une fois le sommet y enlevé, il n'existe plus de chemin reliant les sommets entre eux dans F_{c_2}.

Les égalités sur les couleurs manquantes des sommets des deux faces sont vérifiées puisque les couleurs manquantes des sommets dans H pour les deux partitions sont exactement les mêmes que celles dans H.

L'application de ce procédé sur tous les sous-graphes contractibles des deux faces prend un temps linéaire en la taille de H car le calcul d'un réaliseur s'effectue en temps linéaire en la taille du graphe [86]. □

On a maintenant tous les outils à notre disposition pour démontrer le résultat principal de cette partie.

Lemme 4. *Toute triangulation cylindrique est* 3*-partitionnable.*

Démonstration. Le principe de l'algorithme consiste à contracter les deux embouts de H obtenant ainsi le graphe H'. Ensuite, on partitionne H' en trois arbres grâce à un réaliseur. On en déduit la partition voulue de H en ajoutant les arêtes contractées durant la contraction des faces dans la partition.

Le lemme 3 nous permet de considérer où les deux embouts de H sont simples. Il est clair que l'opération de contraction de face ne crée pas de face de taille quatre ou plus. Par conséquent, H' est soit isomorphe à K_2, soit une triangulation du plan. On considère trois cas suivant l'ordre du graphe H' obtenu.

– **Cas 1 :** $V(H') = 2$

  Dans ce cas, on a $V(H) = V(F_1) \cup V(F_2)$. La figure 10 nous donne un exemple de graphe possible dans ce cas.
  On définit la partition d'arêtes $\mathcal{A}$ comme ceci :
  – $A_1 = E(H) - (E(F_1) \cup E(F_2)$
  – $A_2 = E(F_1)$
  – $A_3 = E(F_2)$
  et la partition d'arêtes $\mathcal{A}'$ comme ceci :
  – $A'_1 = E(H) - (E(F_1) \cup E(F_2)$
  – $A'_2 = \varnothing$
  – $A'_3 = E(F_1) \cup E(F_2)$

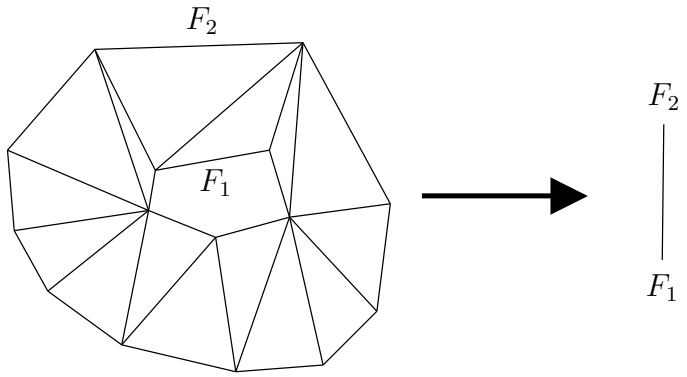


Figure 10 – Exemple de configuration pour le cas $H' = K_2$.

Dans ces deux partitions, il y a seulement 3 cycles dans les graphes induits par les parties de chaque partition. Ce sont les deux cycles des faces et un cycle composé des autres arêtes. Il suffit donc d'enlever trois arêtes au graphe pour que les graphes induits par les parties des deux partitions soient acycliques. Les sommets de $V(F_1)$ ont pour couleur manquante 1 selon la partition $\mathcal{A}$ puisqu'ils ne sont pas contenus dans le graphe induit par A_1. De même, les sommets de $V(F_2)$ ont pour couleur manquante 2 selon cette même partition. Pour la partition $\mathcal{A}'$, les sommets de $V(F_1) \cup V(F_1)$ ont pour couleur manquante 2. Il est facile de vérifier que les deux partitions respectent les conditions pour les couleurs manquantes des sommets des faces énoncées dans le lemme.

– **Cas 2 :** $V(H') = 3$

Dans ce cas, $V(H)$ est constitué des sommets des deux faces plus un sommet x. Puisque les deux faces sont simples, le sommet x est relié à au moins deux sommets de chaque face car sinon il serait situé dans un sous-graphe contractible d'une des deux faces. On utilise quasiment les mêmes partitions que dans le cas 1. La seule différence se situe au niveau de deux arêtes correspondantes de F_1 ayant x comme extrémité. On place l'une de ces arêtes dans A_2 et l'autre dans A_3.

– **Cas 3 :** $V(H') \geqslant 4$

Dans ce cas, il existe un sommet y dans H' qui n'est pas issu d'une contraction d'une des deux faces et qui est extrémité d'au moins deux arêtes correspondantes de F_1 dans H puisque sinon la contraction successive des deux faces nous donnerait le graphe K_2. On calcule un réaliseur de H' en prenant pour face externe xyz avec x issu de la contraction de F_1 et z qui n'est pas issu de la contraction de F_2. On place les arêtes des arbres enracinés en x, y et z respectivement dans les parties A_1, A_2 et A_3. On place les arêtes externes xy et xz dans A_1 et l'arête externe yz dans A_2. On en déduit une partition partielle des arêtes du graphe H en mettant chaque arête de

H dans la même partie que l'arête qui lui correspond dans H'.

– partition des arêtes de F_1.

Dans H', le sommet x n'est couvert que par des arêtes de A_1. Par conséquent, toutes les arêtes incidentes aux sommets des faces sont contenues dans A_1. Les arêtes correspondantes de F_1 induisent un cycle dans $H[A_1]$. Afin de rendre A_1 acyclique, on place une des arêtes du cycle ty dans la partie A_3. Ce procédé ne crée pas de cycle dans $H[A_3]$ car, par construction, les sommets de $V(F_1)$ et y n'étaient pas couverts par le graphe $H[A_3]$. On choisit une arête uv de $E(F_1)$ telle que $uy, vy \in E(H)$ et $u \neq t$. On place les arêtes de $E(F_1) \cup \{uy\} \setminus \{uv\}$ dans A_2 pour la partition $\mathcal{A}$ et dans A_3 pour la partition $\mathcal{A}'$. On place l'arête uv dans A_1. Il est facile de vérifier qu'aucun cycle n'est créé dans les graphes $H[A_1]$, $H[A_2]$ et $H[A_3]$. De plus, les sommets de F_1 ont pour couleur manquante 3 selon $\mathcal{A}$ puisque les sommets de F_1 ne sont connectés dans $H[A_3]$ qu'au sommet y. De même, les sommets de F_1 ont pour couleur manquante 2 selon $\mathcal{A}'$.

– partition des arêtes de F_2.

Pour cette deuxième face on applique la même partition pour les deux partition de $E(H)$: $\mathcal{A}$ et $\mathcal{A}'$
Le sommet s issu de la contraction de F_2 est un sommet interne de H' et donc vérifie la propriété locale du réaliseur de Schnyder. Les arêtes du sommet s forment donc des blocs.
- Bloc U_1 : Le père de s dans l'arbre A_1
- Bloc D_3 : les fils de s dans l'arbre A_3
- Bloc U_2 : le père de s dans l'arbre A_2
- Bloc D_1 : les fils de s dans l'arbre A_1
- Bloc U_3 : le père de s dans l'arbre A_3
- Bloc D_2 : les fils de s dans l'arbre A_2

Les arêtes incidentes aux sommets de F_2 forment donc aussi 6 blocs autour de la face. On note u_0, u_2, u_3, v_1, v_2 et v_3 les sommets délimitant ces 6 blocs où :
- u_1 et v_1 délimite le bloc U_1,
- v_1 et u_2 délimite le bloc D_3,
- u_2 et v_2 délimite le bloc U_2,
- v_2 et u_3 délimite le bloc D_1,
- u_3 et v_3 délimite le bloc U_3 et
- v_3 et u_1 délimite le bloc D_2.

On place les arêtes du chemin $u_1 \cdots v_1 \cdots u_2$ dans A_2, celles du chemin $u_2 \cdots v_2 \cdots u_3$ dans A_3, et celles du chemin $u_3 \cdots v_3 \cdots u_1$ dans A_1. Il reste à prouver qu'il existe un entier $0 \leqslant k \leqslant 3$ tel que :

– il y a au plus k cycles dans les graphes $H[A_1]$, $H[A_2]$ et $H[A_3]$ et donc qu'il suffit d'enlever k arêtes pour rendre les graphes acycliques.
– il y a au plus $3-k$ sommets de $V(F_1)$ n'ayant pas de couleur manquante.

Pour tout $i \in \{1,2,3\}$, $H'[A_i]$ est acyclique et donc il ne peut exister un cycle dans $H[A_i]$ que dans le sous-graphe composé des sommets de $V(F_2)$ et de leurs voisins. Il ne peut y avoir qu'un seul cycle dans $H[A_i]$ et cela si et seulement si $v_i = u_{(i \mod 3)+1} = v_{(i \mod 3)+1}$.
Tous les sommets de F_2 à l'exception de v_1, v_2 et v_3 ont une couleur manquante selon $V(F_1) \cup V(F_2) \setminus \{v_1, v_2, v_3\}$ puisque par construction il ne sont que seulement dans 2 des graphes induits par les parties A_1, A_2 et A_3. Les sommets internes des chemins $v_3 \cdots u_1 \cdots v_1$, $v_1 \cdots u_2 \cdots v_2$ et $v_2 \cdots u_3 \cdots v_3$ ont pour couleur manquante respectivement 3, 1 et 2.
On considère trois sous-cas suivant que v_1, v_2 et v_3 soient des sommets distincts ou non.

– **Cas 3.1 :** $v_1 \neq v_2$, $v_2 \neq v_3$ et $v_3 \neq v_1$.

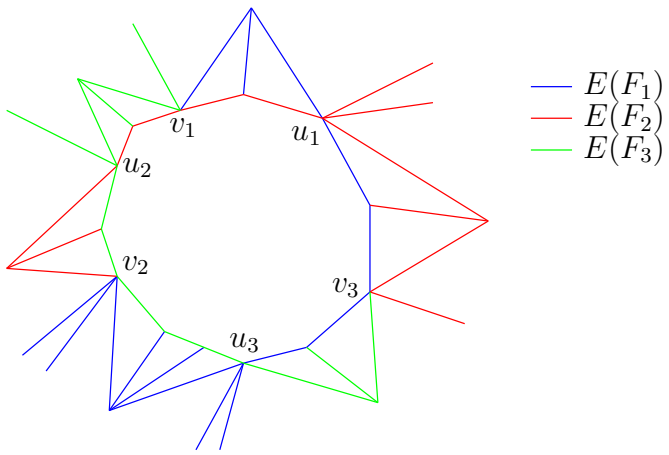


Figure 11 – Un exemple pour le cas 3.1.

Dans ce cas, il n'y a aucun cycle dans les graphes $H[A_1]$, $H[A_2]$ et $H[A_3]$. Il y a exactement 3 sommets sans couleur manquante car v_1, v_2 et v_3 sont des sommets distincts.

– **Case 3.2 :** $v_1 = v_2$ et $v_1 \neq v_3$ et $v_2 \neq v_3$
ou $v_1 \neq v_2$ et $v_1 = v_3$ et $v_2 \neq v_3$
ou $v_1 \neq v_2$ et $v_1 \neq v_3$ et $v_2 = v_3$

Dans ce cas, il y a un cycle dans l'un des graphes $H[A_1]$, $H[A_2]$ et $H[A_3]$. Il y a exactement 2 sommets sans couleur manquante parmi v_1, v_2 et v_3, seuls deux sommets sont distincts.

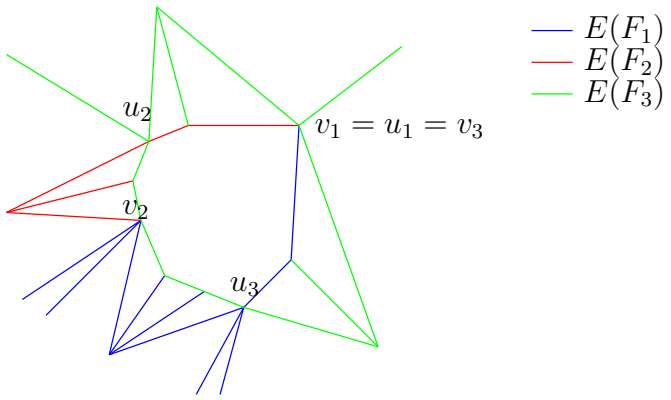


Figure 12 – Un exemple pour le cas 3.2 avec $v_1 = u_1 = v_3$.

– **Cas 3.3 :** $v_1 = v_2 = v_3$.

Dans ce cas, il y a un cycle dans deux des graphes $H[A_1]$, $H[A_2]$ et $H[A_3]$ et un seul sommet sans couleur manquante car $v_1 = v_2 = v_3$.

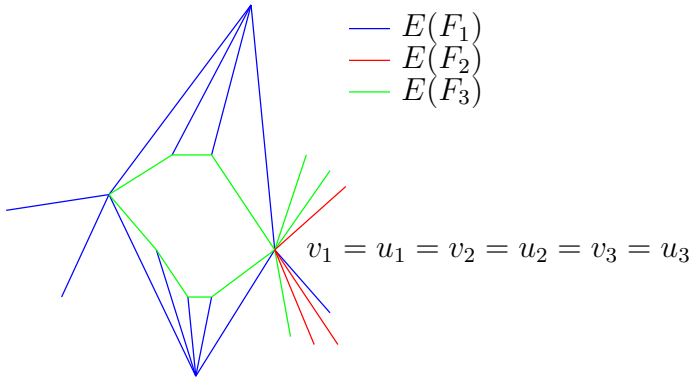


Figure 13 – Un exemple pour le cas 3.3.

Ceci termine la preuve du lemme 4 □

2.4.4 Étape 3 : partition du tambourin

C'est la dernière étape de notre algorithme, il nous faut trouver une partition d'arêtes du graphe G à partir de la partition d'arêtes de la triangulation cylindrique H en ajoutant chaque arête de T dans l'une des quatre parties de la partition. La difficulté réside dans le fait de ne pas créer de cycle lors de l'ajout des arêtes dans les parties correspondant à des forêts.

Lemme 5. *Si la triangulation cylindrique H est 3-partitionnable alors G peut être partitionné en trois forêts plus un ensemble d'au plus trois arêtes en temps linéaire.*

Démonstration. Puisque H est 3-partitionnable il existe deux partitions d'arêtes $\mathcal{A}$ et $\mathcal{A}'$ respectant les conditions énoncées par la définition 5. On considère d'abord la partition $\mathcal{A}$ qui partitionne les arêtes du sous-graphe H en trois forêts A_1, A_2 et A_3 plus un ensemble de k arêtes : A_0. Il nous reste à placer chaque arête de T dans l'un de ses quatre ensembles A_0, A_1, A_2 ou A_3.

On fixe la couleur manquante des sommets de l'ensemble V_0 décrit dans la définition 5 comme étant égal à 0. Le principe de notre partition de T est d'orienter d'une manière spécifique les arêtes de T et ensuite de placer chaque arête de T dans la partie correspondant à la couleur manquante de sa cible. Toute la difficulté de la partition de T réside donc dans l'orientation des arêtes de T. Avant de décrire l'orientation de T, il nous faut remarquer que les sommets de T peuvent être partitionné en deux parties (voir figure 14) :

- Les sommets de degré 1 dans T.
- Les sommets de degré supérieur ou égal à 2 qui induisent un cycle D dans le tambourin.

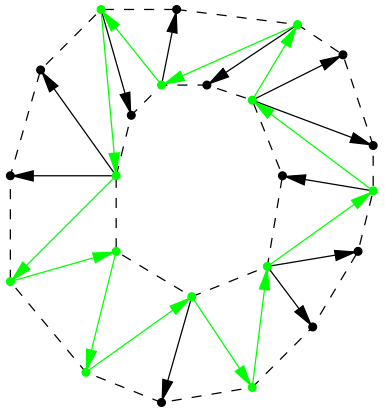


Figure 14 – Exemple d'un tambourin T orienté.

On oriente le cycle D afin d'obtenir un circuit et on oriente les arêtes ayant une extrémité v de degré 1 vers v. On obtient ainsi une orientation de T de degré entrant 1. Chaque sommet ne va donc être utilisé qu'une seule fois pour déterminer les parties des arêtes de T. Par conséquent, on ajoute donc au maximum $3-k$ arêtes à la part A_0 car $3-k$ sommets ont pour couleur manquante 0 et donc $|A_0| = 3$ dans la partition de G.

Cet algorithme de partition en forêts peut créer un cycle dans le graphe $T \cap F_i$ si tous les sommets de D ont la même couleur manquante i dans $\mathcal{A}$ car dans ce cas les arêtes de D sont dans une même partie. C'est dans ce cas précis qu'on utilise la deuxième partition $\mathcal{A}'$ de H. On applique notre algorithme de partition de T sur cette partition $\mathcal{A}'$ de $H \setminus A_0$. D'après la définition 5, on a $\forall i, 1 \leqslant i \leqslant r, C_{\mathcal{A}}(x_i) = C_{\mathcal{A}'}(x_i)$ et $\forall i, 1 \leqslant i \leqslant s, C_{\mathcal{A}}(y_i) = C_{\mathcal{A}'}(y_i)$. Par conséquent, tous les sommets de degré supérieur ou égal à 2 dans T n'ont qu'une même couleur manquante dans $\mathcal{A}'$. Donc, il n'y a pas de cycle dans $T \cap F_i$. Pour le reste de la preuve, on note $\mathcal{A} = \{A_0, A_1, A_2, A_3\}$ la partition de G ainsi obtenue.

Il nous reste à montrer que A_i induit une forêt pour $i = \{1, 2, 3\}$. Puisque $A_i \cup T$ et

$A_i \cup H$ sont des forêts pour $i = \{1, 2, 3\}$, il nous suffit de montrer qu'il n'existe pas dans F_i de cycle formé d'au moins une arête de T et au moins une de $E(H)$.

Supposons par l'absurde qu'il existe un tel cycle C dans $G[A_i]$ avec

$$C = z_0 \cdots z_{r-1} z_r \cdots z_t z_{t+1} \cdots z_s$$

avec $z_{r-1}z_r, z_t z_{t+1} \in H \cap A_i$ et $e_r e_{r+1}, e_{r+1}e_{r+2}, \cdots e_{t-1}e_t \in T \cap A_i$. Le sommet z_r n'a pas comme couleur manquante i puisqu'il existe un chemin composé d'arêtes de F_i entre z_r et un sommet de $C_1 \cup C_2$. De même z_t n'a pas comme couleur manquante i. Il y a donc un chemin dans T composé d'arêtes de F_i entre deux sommets z_r et z_t de couleurs manquantes différentes de i. On peut exclure tout de suite le cas où $\deg_T(z_r) = 1$ puisque dans ce cas l'arête $z_r z_{r+1}$ ne serait pas dans F_i d'après notre algorithme. De même, on peut exclure le cas $\deg_T(z_t) = 1$. Il ne nous reste donc plus que le cas ou les deux sommets ont un degré 2. Dans ce cas, notre chemin est fait un chemin dirigé sous-graphe du circuit D. On en déduit que z_t ou z_r est cible d'une des arêtes du chemin. Par conséquent, une des arêtes du chemin n'est pas dans A_i et on obtient une contradiction. Ceci termine la preuve du lemme 5 et aussi du théorème 7. □

2.5 Autres partitions d'arêtes des graphes de genre borné

2.5.1 Partition d'arêtes des graphes de genre borné en g-forêts

Une forêt plongée dans la sphère $\mathbb{S}_0$ ne sépare pas la surface puisqu'elle ne contient pas de cycle. Il est donc raisonnable de généraliser cette notion de forêt sur des surfaces autre que la sphère de la même manière. On définit une g-forêts comme étant un plongement de graphe sur la surface $\mathbb{S}_g$ ne séparant pas $\mathbb{S}_g$. D'après la formule d'Euler, les g-forêts de n sommets ont au plus $n - 1 + 2g$ arêtes puisqu'elles n'ont qu'une seule face. En fait les graphes que l'on peut plonger de manière à obtenir une g-forêt peuvent être construits à partir d'une forêt en ajoutant $2g$ arêtes sans rajouter de sommets. Il est donc possible de partitionner les graphes de genre g en 3 g-forêts qui ne sont pas nécessairement bien plongées. En effet, les graphes de genre orientable g peuvent être partitionnés en trois forêts plus $6g - 3$ arêtes d'après le corollaire 1. Il suffit de rajouter $2g - 1$ à chaque forêt de la partition afin d'obtenir une partition en g-forêts.

On peut se poser la question de savoir s'il on peut *décomposer* une carte G de genre orientable g en 3 g-forêts, c'est-à-dire trouver une partition des arêtes en trois parties de G dont chaque partie induit une sous-carte correspondant à une g-forêt plongée sur $\mathbb{S}_g$. C'est bien entendu possible pour $g = 0$ et ceci en temps linéaire grâce au réaliseur de Schnyder [85]. En fait, il est possible de réaliser une telle décomposition pour $g = 1$.

Théorème 8. *Les arêtes d'une triangulation torique peuvent être décomposées en* 3 1*-forêts bien plongées en temps linéaire.*

Démonstration. On utilise l'algorithme de partition décrit dans la partie précédente. Il suffit de placer les 3 arêtes de A_0 dans les parties A_1, A_1 et A_3 en créant des cycles non-séparants dans les graphes induits par les parties. Pour les arêtes placées dans A_0 durant l'étape 2 de l'algorithme, il suffit de les replacer dans les parties dont on les a enlevées afin de rendre les sous-graphes induits par les parties acycliques. Les cycles dont on les avait supprimées sont non-séparants. Pour les arêtes placées dans A_0 durant l'étape 3 de l'algorithme, on considère deux cas :

- **Cas 1 :** $A_0 \subset T$ et pour toute arête xy de A_0, avec $x = \text{ref}(xy)$, on a $C_F(y) = c$.

  Pour $i = 1, 2, 3$, on place une arête de A_0 dans la partie A_i. On crée ainsi 1 cycle non-séparant dans le graphe $G[A_i]$ car dans le cas ou $i = c$, on crée un cycle séparant selon le tambourin et dans le cas $i \neq c$, on crée un cycle allant du coté gauche au coté droit du tambourin et donc non-séparant. Les sous-graphes obtenus sont donc non-séparant.
- **Cas 2 :** Les autres cas.

  Dans tous les autres cas, on choisit de placer les arêtes xy de A_0 dans F_i de telle sorte que l'on mette au plus une arête dans chaque partie et que $i \neq C_{\mathcal{A}}(x)$. Dans ce cas on crée un cycle non-séparant qui n'est pas séparant selon les cycles du tambourin. Les graphes induits par les parties ont donc au plus deux cycles non-séparants dont chacun n'est pas séparants selon l'autre. Les sous-graphes obtenus sont donc non-séparant.

□

On conjecture que cette propriété est vraie pour $g > 1$. Prouver ou réfuter cette conjecture semble assez difficile car c'est une décomposition d'une carte en sous-cartes plutôt qu'une partition d'arêtes.

Conjecture 1. *Les arêtes d'une triangulation de $\mathbb{S}_g$ peuvent être décomposées en 3 g-forêts.*

En plus des arbres de Schnyder, il existe une autre partition d'arêtes des graphes planaires en forêts ayant des propriétés qui nous seront utiles dans les prochains chapitres.

Théorème 9 ([53])**.** *Les arêtes de tout graphe planaire G peuvent être partitionnées en trois forêts dont une est de degré borné.*

S'inspirant de ce résultat sur les planaires, on conjecture la propriété suivante sur les graphes de genre g.

Conjecture 2. *Les arêtes d'une triangulation de $\mathbb{S}_g$ peuvent être décomposées en 3 g-forêts dont une est de degré bornée dépendant uniquement de g.*

Cette conjecture semble très difficile car on ne sait même pas prouver ou réfuter cette conjecture si on se restreint à une partition des arêtes du graphe plutôt qu'à une décomposition de la carte. On pourrait penser que le résultat de [53] pour les graphes

planaires combiné avec notre partition des graphes toriques en un graphe planaire et un tambourin permettrait de prouver cette conjecture pour $g = 1$. Malheureusement, cette partition ne satisfait pas la condition locale de Schnyder ou une propriété approchant. Or une telle condition locale est nécessaire pour notre construction. Par conséquent, il n'est pas possible de résoudre cette conjecture pour $g = 1$ en utilisant la même construction.

2.5.2 Partition d'arêtes des graphes de genre borné en graphes g-extérieurs

Les graphes planaires extérieurs sont les graphes que l'on peut plonger sur la sphère $\mathbb{S}_0$ tels que tous leurs sommets soient sur le bord d'une seule face appelée *face extérieure*. Il semble naturel d'étendre cette notion aux surfaces de genre supérieur. On définit donc les graphes g-extérieurs comme les graphes que l'on peut plonger sur $\mathbb{S}_g$ tels que tous leurs sommets soient sur le bord d'une seule face. Ces familles de graphes n'ont été étudiées que depuis peu [33, 29] et peu de propriétés leur sont connues. Ce qui nous intéresse ici est une généralisation d'une partition connue pour les graphes planaires. Il est possible de partitionner les arêtes d'un graphe planaire en deux graphes planaires extérieurs [54]. Par contre, la décomposition d'une carte planaire en deux cartes planaires extérieurs n'est pas toujours possible [61]. On conjecture la généralisation suivante pour les graphes de genre supérieur.

Conjecture 3. *Les arêtes de tout graphe de genre g peuvent être partitionnées en un graphe $(g-k)$-extérieur et un graphe k-extérieur avec $k \leqslant g$.*

Cette conjecture est vraie pour toute carte ayant un cycle hamiltonien séparant. En effet, dans ce cas il est possible de couper le graphe en deux graphes un $(g-k)$-extérieur et l'autre k-extérieur qui ont pour face extérieure le cycle hamiltonien. Cette conjecture est donc vraie pour les triangulations du tore 5-connexes [90]. Prouver cette conjecture semble très complexe car contrairement au graphe planaires extérieurs peu de propriétés sont connues sur les graphes g-extérieurs.

2.6 Conclusion

Le résultat principal de cette section est la partition des graphes de genre g en 3 forêts. Une amélioration significative pourrait être de trouver une partition en forêts ayant des propriétés intéressantes, comme par exemple une borne sur le diamètre des arbres par rapport a celui du graphe ou encore des propriétés locales similaires à celle des réaliseurs de Schnyder. Dans ce cadre, on propose la généralisation suivante des réaliseurs de Schnyder aux graphes de genre supérieur.

Conjecture 4. *Les arêtes de toute carte de genre g ayant n sommets peuvent être partitionnées en 3 g-forêts en temps $O(k.n)$ avec $k = k(g)$. De plus, les arêtes autour de chaque sommet forment au plus k blocs de couleurs différentes.*

Cette conjecture est vérifiée pour $g = \{0, 1\}$. En effet, dans notre partition des graphes toriques, les arêtes autour des sommets forment au plus un nombre constant de blocs de couleurs différentes. Il est donc raisonnable de penser qu'elle est aussi vraie pour les graphes de genre supérieur.

Deuxième partie

Représentation implicite de graphes

Chapitre 3

Graphes universels et variantes

3.1 Introduction

On considère une famille de graphes $\mathcal{F}$. Un graphe U est dit *$\mathcal{F}$-universel* s'il contient tous les graphes de la famille $\mathcal{F}$ en tant que sous-graphes. Bien qu'il soit possible de considérer le cas où $\mathcal{F}$ est une famille infinie et que de nombreux travaux ont été entrepris en ce sens [52, 78, 79, 95, 26, 63, 64, 62], on se restreindra ici au cas où $\mathcal{F}$ est finie. On va considérer des familles de graphes d'au plus n sommets. Le graphe complet à n sommets K_n est clairement universel pour ces familles. Le problème consiste à trouver un graphe universel d'ordre n dont le nombre d'arêtes est minimum.

Une des motivations originelles pour l'étude des graphes universels est liée à la conception de circuits intégrés [17]. La conception de ces circuits est très coûteuse. Afin de rentabiliser les coûts de conception, les fabricants de circuits se doivent donc de vendre de nombreuses copies d'un même circuit. Une des solutions à ce problème est d'utiliser des circuits "universels", c'est-à-dire un circuit qui contient plusieurs circuits. En effet, il est possible de modifier un circuit lors de sa dernière étape de fabrication afin de supprimer des connexions ou composantes superflues et donc de concevoir un circuit ayant plusieurs usages. Les circuits intégrés étant facilement modélisables par des graphes, le problème se ramène aisément au problème du graphe universel minimum.

Cette application a motivé l'étude de graphes universels pour de nombreuses familles de graphes de n sommets, dont les forêts [28], les chenilles [1] [43] et les graphes planaires [15]. Plus récemment, la communauté s'est intéressée au cas des familles de graphes de degré borné. Dans ce cadre, les forêts de degré borné [14, 16], les graphes de degré borné [4] et les graphes planaires de degré borné [25] ont été étudiés.

Il existe une variante de la notion de graphe universel se définissant comme suit. Un graphe est dit *$\mathcal{F}$-universel induit* (notion introduite pour la première fois par [41]) s'il

1. Un arbre dont les sommets internes induisent un chemin

contient tous les graphes de la famille $\mathcal{F}$ en tant que sous-graphes induits. Cette notion est plus forte puisque chaque graphe $\mathcal{F}$-universel induit est $\mathcal{F}$-universel. Contrairement au cas des graphes universels simple le problème est de minimiser le nombre de sommets du graphe universel induit plutôt que son nombre de arêtes. L'intérêt des graphes universels induits réside dans le fait qu'ils sont fortement liés à une notion de structures de donnée distribuée connue sous le nom de *schéma d'étiquetage d'adjacence* (notion introduite pour la première fois par [23]) ou *représentation implicite*. Un schéma d'adjacence consiste à donner à chaque sommet des graphes de $\mathcal{F}$ une étiquette binaire de telle sorte que l'on puisse tester l'adjacence entre deux sommets en utilisant seulement les étiquettes des sommets concernés. De plus, on force les étiquettes à être distinctes pour les sommets d'un même graphe afin qu'elles puissent servir d'identifiants pour les sommets. L'ensemble des étiquettes constitue une représentation du graphe puisqu'il est possible de recalculer celui-ci par le biais de requêtes d'adjacence sur toutes les paires de sommets. On peut donc manipuler le graphe en gardant seulement les étiquettes des sommets. Le but est bien sûr de minimiser l'espace mémoire nécessaire au stockage de cette représentation et donc de minimiser la longueur des étiquettes. Pour des raisons évidentes, il est aussi utile de minimiser la complexité des fonctions d'étiquetage et d'adjacence.

Kannan, Naor et Rudich [58] ont établi que l'existence d'un schéma d'adjacence avec des étiquettes de k bits pour une famille $\mathcal{F}$ est équivalente à l'existence d'un graphe $\mathcal{F}$-universel induit ayant 2^k sommets. Pour passer du graphe $\mathcal{F}$-universel induit U à un schéma d'adjacence de $\mathcal{F}$, il suffit d'étiqueter les sommets de U. La fonction d'étiquetage d'un graphe $G \in \mathcal{F}$ consiste à *encastrer* G dans U, c'est-à-dire associer de manière injective chaque sommet de G à un sommet du sous-graphe induit correspondant dans U, et à donner à chaque sommet l'étiquette du sommet de U correspondant. La fonction d'adjacence consiste tout simplement à tester l'adjacence dans U. Dans le sens inverse, on construit le graphe universel induit en prenant pour ensemble de sommets toutes les étiquettes pouvant être attribuées par le schéma et comme ensemble d'arêtes, les paires d'étiquettes donnant une réponse positive pour le test d'adjacence. L'encastrement d'un graphe de $\mathcal{F}$ s'effectue en étiquetant le graphe et en encastrant chaque sommet dans le sommet ayant la même étiquette dans U. Par conséquent, le problème consistant à concevoir un schéma d'adjacence le plus compact possible est équivalent au problème combinatoire consistant à trouver un graphe universel induit minimal. Un résultat pour un des deux problèmes implique donc directement un résultat pour l'autre problème. Malheureusement, cette équivalence ne tient pas compte des complexités en temps des fonctions d'étiquetage et d'adjacence. L'existence d'un graphe universel induit n'implique donc pas nécessairement un schéma d'adjacence efficace en termes de complexité en temps.

L'une des familles les plus étudiées pour ces deux problèmes est la famille des forêts et par extension la famille des graphes d'arboricité bornée. Kannan, Naor et Rudich [58] ont décrit en 1988 le premier schéma connu pour la famille des graphes d'arboricité k qui utilise des étiquettes de $(k+1)\log n$ bits. Ce schéma a été ensuite amélioré par Chung [27] en un schéma utilisant $k\log n + k\log\log n + O(1)$ bits. En fait, ce schéma est déduit d'un graphe universel induit de $O((n\log n)^k)$ sommets. Finalement, le meilleur schéma existant utilise

$k \log n + O(k \log^* n)$ [9]. Pour de nombreuses familles, ce schéma est le meilleur connu. Par exemple, pour les graphes planaires, le schéma le plus compact est en $3 \log n + O(\log^* n)$ et est déduit du fait que les graphes planaires ont une arboricité 3.

Dans ce chapitre, on s'intéresse particulièrement aux graphes universels induits pour la famille des graphes de degré borné. On décrit la construction d'un graphe universel induit $U_{n,k}$ pour la famille $\mathcal{F}_{n,k}$ des graphes de degré maximum k ayant n sommets. Pour k pair, le graphe $U_{n,k}$ a $O(n^{k/2})$ sommets et pour k impair, $U_{n,k}$ a $O(n^{\lceil k/2 \rceil - 1/k} \log^{2+2/k} n)$ sommets. Dans le cas où $k \equiv 0 \pmod 2$, le graphe universel induit est obtenu par une construction similaire à celle décrite dans [24] mais avec une amélioration du graphe servant de base de la construction. Dans le cas où $k \equiv 1 \pmod 2$, $U_{n,k}$ est déduit d'un résultat récent de Alon et Capalbo [4] pour les graphes $\mathcal{F}_{n,k}$-universel combiné avec une construction de Chung [27] permettant de construire un graphe universel induit à partir d'un graphe universel. Le nombre de sommets d'un graphe $\mathcal{F}_{n,k}$-universel induit étant d'au moins $\Omega(n^{k/2})$ [24], il s'en suit que pour k pair $U_{n,k}$ est optimal à une constante multiplicative près. En revanche, pour k impair, $U_{n,k}$ a un nombre de sommets égal à $O(n^{1/2-1/k} \log^{2+2/k} n)$ fois ce minorant.

Dans la deuxième partie du chapitre, on s'intéresse au cas des graphes universels induits pour des graphes orientés. On donne d'abord une construction pour un graphe universel induit de $O(n^k)$ sommets pour la famille $\mathcal{O}_{n,k}$ des graphes de degré entrant et sortant au plus k. Ce majorant est optimal à une constante près car tout les graphes de $\mathcal{O}_{n,k}$ peuvent être obtenus en orientant un graphe de $\mathcal{F}_{n,2k}$. Par conséquent tout graphe $\mathcal{O}_{n,k}$-universel induit est $\mathcal{F}_{n,2k}$-universel induit s'il on ne tient pas compte de l'orientation et le minorant de n^k pour le nombre de sommets d'un graphe $\mathcal{F}_{n,2k}$-universel induit est un minorant pour le nombre de sommets d'un graphe $\mathcal{O}_{n,k}$-universel induit.

Ensuite, on décrit une construction générale permettant de passer du cas orienté au cas non-orienté. On aborde après le cas des graphes de genre borné. Enfin, on conclut par quelques problèmes ouverts.

L'ensemble de ce travail a fait l'objet d'un rapport de recherche [42].

3.2 Graphe universel induit pour les graphes de degré borné

Notre but principal dans cette partie est de construire un graphe $\mathcal{F}_{n,k}$-universel induit pour tout $k \geqslant 2$. On étudie d'abord le cas $k = 2$.

3.2.1 Graphe universel induit pour les graphes de degré deux

Lemme 6. *Le graphe $U_{n,2}$ représenté dans la figure 15 ayant $\frac{5}{2}n + O(1)$ sommets et $\frac{9}{2}n + O(1)$ arêtes est un graphe $\mathcal{F}_{n,2}$-universel induit.*

Démonstration. Il suffit de prouver que chaque graphe $G \in \mathcal{F}_{n,2}$ est un sous-graphe induit

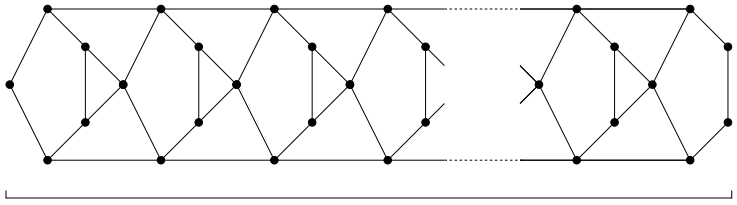


Figure 15 – Le graphe $\mathcal{F}_{n,2}$-universel induit $U_{n,2}$.

de $U_{n,2}$. Pour $1 \leqslant i \leqslant n$, soit n_i le nombre de composantes connexes de G ayant i sommets. Le degré de G étant borné par deux, les composantes connexes de G sont donc des chemins ou bien des cycles. Par conséquent, G contient n_1 sommets isolés, n_2 K_2 disjoints et pour $i \geqslant 3$, n_i cycles ou chemins disjoints de longueur i.

Le graphe $U_{n,2}$ est constitué de cycles de longueur 5 appelés *tuiles* qui sont reliées en série par quatre arêtes chacune. On va montrer qu'il est possible d'encastrer les composantes connexes de G dans au plus $\lfloor \frac{n}{2} \rfloor + 5$ tuiles et donc dans le graphe $U_{n,2}$. Pour ce faire, on trie les composantes connexes de G d'après leurs tailles et on les encastre dans les tuiles de $U_{n,2}$ de gauche à droite par taille croissante. Les sommets encastrés sont représentés dans les figures par des sommets blancs.

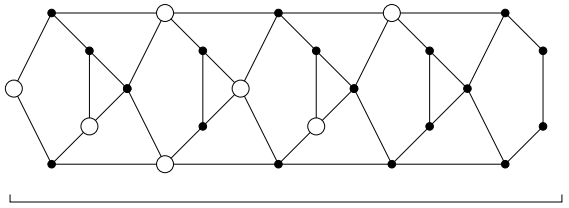


Figure 16 – L'encastrement du stable de n_1 sommets, utilisant $\lceil \frac{n_1}{2} \rceil + 1$ tuiles.

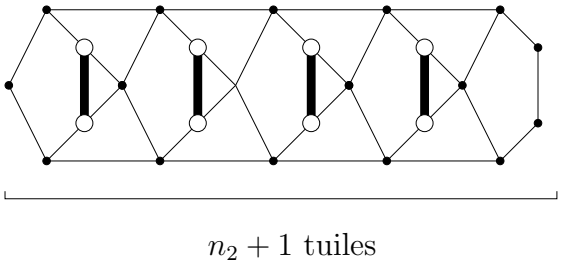
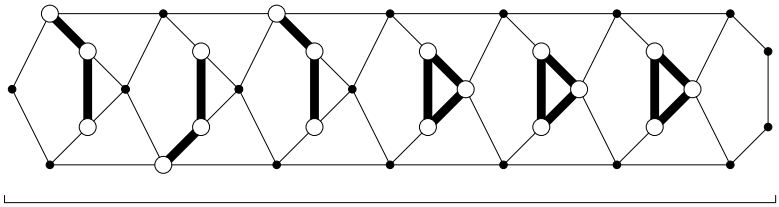
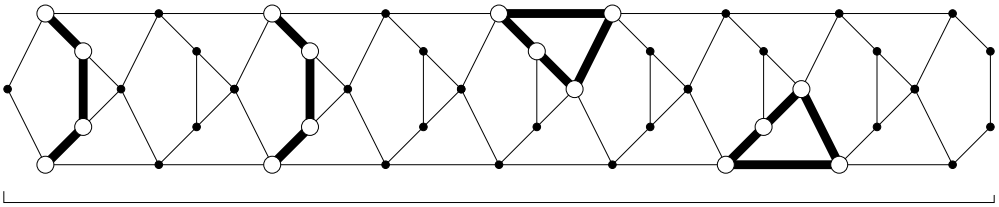


Figure 17 – L'encastrement de n_2 K_2, utilisant $n_2 + 1$ tuiles.



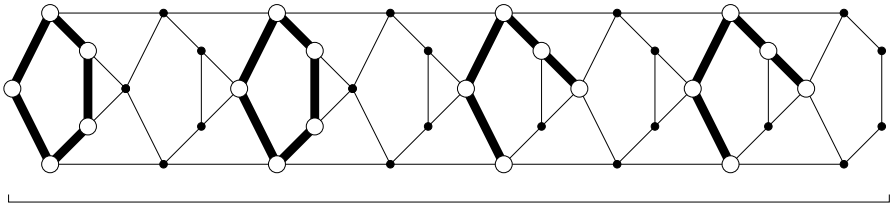
$n_3 + 1$ tuiles

Figure 18 – L'encastrement des n_3 composantes connexes de taille 3, utilisant $n_3 + 1$ tuiles.



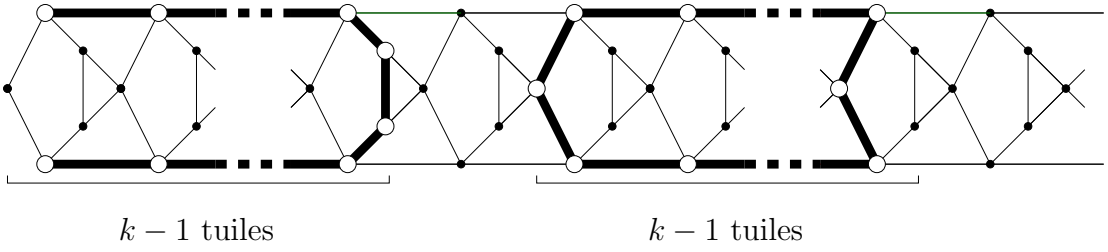
$2n_4 + 1$ tuiles

Figure 19 – L'encastrement des n_4 composantes connexes de taille 4, utilisant $2n_4 + 1$ tuiles.



$2n_5$ tuiles

Figure 20 – L'encastrement des n_5 composantes connexes de taille 5, utilisant $2n_5$ tuiles.



$k - 1$ tuiles $\qquad$ $k - 1$ tuiles

Figure 21 – Pour $k \geqslant 3$, l'encastrement des n_{2k} composantes connexes de taille $2k$, utilisant kn_{2k} tuiles.

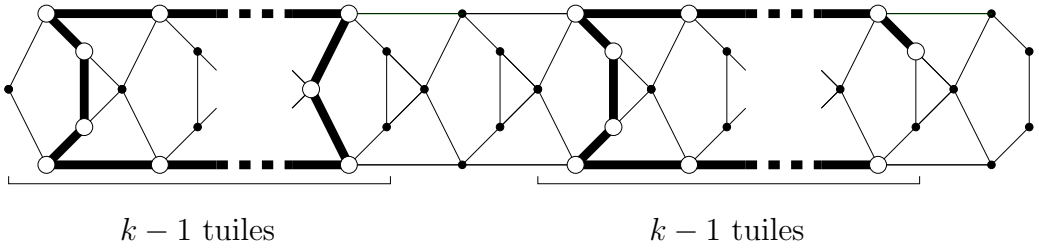


Figure 22 – Pour $k \geqslant 3$, l'encastrement des n_{2k+1} composantes connexes de taille $2k+1$, utilisant kn_{2k+1} tuiles.

Il est facile de vérifier que pour tout i, l'encastrement des composantes connexes de taille i est induit. De plus, il n'y a pas de sommet de G encastré dans la dernière tuile de la partie de $U_{n,2}$ utilisée pour encastrer les composantes connexes de taille i. Il n'y a donc par d'arêtes reliant des sommets de composantes connexes de tailles différentes. Par conséquent, l'encastrement de G dans $U_{n,2}$ est bien induit. Il ne reste donc plus qu'à majorer le nombre l de tuiles nécessaire pour encastrer G.

$$\begin{aligned} l &= \frac{n_1}{2} + 2 + n_2 + 1 + n_3 + 1 + 2n_4 + 1 + 2n_5 + \sum_{k=3}^{\lfloor n/2 \rfloor} kn_{2k} + \sum_{k=3}^{\lfloor n/2 \rfloor} kn_{2k+1} \\ &\leqslant 5 + \sum_{i=1}^{n} i\frac{n_i}{2} \\ &\leqslant 5 + \left\lfloor \frac{n}{2} \right\rfloor \text{ , puisque } \sum_{i=1}^{n} in_i = n \text{ et que } l \text{ est un entier.} \end{aligned}$$

□

Remarque 3. *Pour $n \geqslant 10$, il est possible de construire un schéma d'adjacence en $\lceil \log n \rceil + 3$ bits pour la famille $\mathcal{F}_{n,2}$ à partir de $U_{n,2}$ de telle sorte que l'adjacence se teste en temps constant et l'étiquetage s'effectue en temps $O(n)$.*

Démonstration. L'idée est tout simplement d'étiqueter $U_{n,2}$ de telle sorte que l'on puisse tester l'adjacence en temps constant à partir des étiquettes des sommets. On étiquette chaque sommet $v \in V(U_{n,2})$ avec un couple (i_v, j_v) où i_v est le numéro de la tuile de v, les tuiles étant numérotées de gauche à droite, et j_v sa position dans sa tuile (voir figure 23). Ce couple peut se coder avec $\log n + 3$ bits. Pour tout $v \in V(U_{n,2})$, $i_v \leqslant n/2 + 5 \leqslant n$ car $n \geqslant 10$ et $j_v \leqslant 4$ et donc i_v peut se coder avec $\lceil \log n \rceil$ bits et j_v avec 3 bits.

L'étiquetage d'un graphe G consiste simplement à encastrer G et à mettre les étiquettes de $U_{n,2}$ aux sommets correspondants de G. L'encastrement de G peut se faire en temps

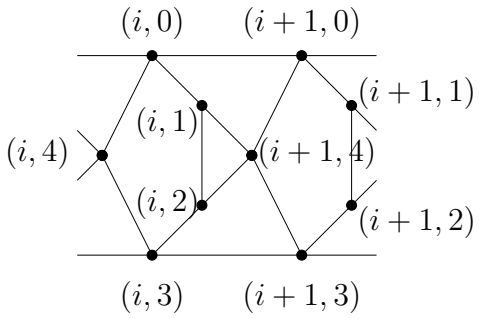


Figure 23 – L'étiquetage de $U_{n,2}$.

linéaire étant donné un découpage du graphe en composantes connexes. Or, ce découpage se calcule facilement puisque G a un degré borné.

On cherche à tester l'adjacence entre deux sommets $u, v \in V(G)$. Le test peut se calculer très facilement grâce à la condition suivante en supposant sans perte de généralité que $i_u \leqslant i_v$:

$$(u,v) \in E(G) \Leftrightarrow \begin{cases} i_u = i_v \text{ et } (i_u \equiv j_v + 1 \pmod 5 \text{ ou } j_v \equiv j_u + 1 \pmod 5) \\ \text{ou } i_v = i_u + 1 \text{ et } (j_u = j_v \in \{0,3\} \text{ ou } (j_u \in \{1,2\} \text{ et } j_v = 4)) \end{cases}$$

□

On peut se demander s'il est possible d'obtenir un graphe universel induit ayant moins de sommets et si oui combien de sommets on peut réduire le graphe. On donne un élément de réponse à cette question dans le fait suivant.

Fait 2. *Chaque graphe $\mathcal{F}_{n,2}$-universel induit contient au moins $11\left\lfloor \frac{n}{6} \right\rfloor$ sommets.*

Démonstration. Soit n un multiple de 6. On considère les trois graphes suivants :
- Le stable de n sommets noté G_1,
- L'union disjointe de $n/2$ K_2 ou arêtes, noté G_2,
- L'union disjointe de $n/3$ K_3 ou triangles, noté G_3.

Ces trois graphes ont n sommets et leur degré est clairement borné par deux. On en déduit que tout graphe $U_{n,2}$ $\mathcal{F}_{n,2}$-universel induit doit contenir ces trois graphes en tant que sous-graphe induit.

Les stables dans G_3 ont au plus $n/3$ sommets car chaque triangle ne peut contenir qu'un seul sommet par stable. Il s'en suit que $U_{n,2}$ doit contenir un stable de $2n/3$ sommets qui est sommet-disjoint de l'encastrement de G_3 car autrement il ne contiendrait pas un stable de n sommets (G_1).

On considère maintenant l'encastrement du graphe G_2. Deux arêtes de G_2 ne peuvent avoir leurs extrémités encastrées dans les trois sommets correspondant à l'encastrement d'un triangle de G_3. En effet, dans le cas contraire, il existerait une arête entre deux arêtes disjointes de G_2 et l'encastrement ne serait pas induit. On en déduit que $U_{n,2}$ doit contenir

$n/3$ triangles plus un graphe H_n ayant un stable de taille $2n/3$ et le graphe composé de $n/2 - n/3 = n/6$ arêtes disjointes en tant que sous-graphe induit. Le graphe H_n doit avoir au moins $2n/3 + n/6 = 5n/6$ sommets puisque chaque arête de G_2 ne peut avoir plus d'une extrémité contenue dans un stable. Au final, $U_{n,2}$ a au moins $11n/6$ sommets pour n multiple de 6. Donc, n'importe quel graphe $\mathcal{F}_{n,2}$-universel induit a au moins $11\lfloor n/6 \rfloor$ sommets pour n quelconque. □

3.2.2 Graphes universels induits pour les graphes de degré pair

On va utiliser le graphe $\mathcal{F}_{n,2}$-universel induit décrit dans la section 3.2 afin d'obtenir une construction générique pour le cas où les graphes ont un degré maximum k avec k pair. Une construction similaire a déjà été utilisée dans [24] mais avec un graphe $\mathcal{F}_{n,2}$-universel induit de $13n/2$ sommets et $15n/2$ arêtes.

Théorème 10. *Soit $k \geqslant 2$ un entier pair. Il existe un graphe $\mathcal{F}_{n,k}$-universel induit $U_{n,k}$ tel que :*

$$|V(U_{n,k})| = (1 + o(1)) \left(\frac{5n}{2}\right)^{k/2} \text{ et } |E(U_{n,k})| = \left(\frac{9k}{10} + o(1)\right) \left(\frac{5n}{2}\right)^{k-1}.$$

Démonstration. Pour $k > 0$, il est connu qu'il existe une partition des arêtes des graphes de degré maximum k en $k/2$ sous-graphes de degré maximum 2 [77]. D'après le lemme 6, il existe un graphe $\mathcal{F}_{n,2}$-universel induit ayant $\frac{5}{2}n + O(1)$ sommets et $\frac{9}{2}n + O(1)$ arêtes. Pour la version algorithmique du problème, celle du schéma d'adjacence, il est facile de construire un schéma à partir d'une partition des arêtes. Pour cela, il suffit tout simplement de concaténer les étiquettes de toutes les parties et de tester l'adjacence dans chaque sous-graphe induit par les parties. Il s'en suit que si l'on peut décomposer un graphe d'une famille $\mathcal{F}$ en k sous-graphes de $\mathcal{H}$ alors on peut construire un graphe $\mathcal{H}$-universel induit W à partir d'un graphe $\mathcal{F}$-universel induit U tel que [27] :

$$|V(W)| = |V(U)|^k \text{ et } |E(W)| = k|V(U)|^{2k-2}|E(U)|.$$

Puisque les graphes de $\mathcal{F}_{n,k}$ peuvent être décomposés en $k/2$ graphes de $\mathcal{F}_{n,2}$, on déduit qu'il existe un graphe $\mathcal{F}_{n,k}$-universel induit $U_{n,k}$ tel que :

$$\begin{aligned}
|V(U_{n,k})| &= |V(U)|^{k/2} \\
&= \left(\frac{5}{2}\right)^{k/2} n^{k/2} + o(n^{k/2}) \\
|E(U_{n,k})| &= \frac{k}{2}|V(U)|^{k-2}|E(U)| \\
&= \frac{k}{2} \cdot \frac{9}{2} \left(\frac{5}{2}\right)^{k-2} n^{k-1} + o(n^{k-1}).
\end{aligned}$$

□

3.2.3 Graphes universels induits pour les graphes de degré impair

On considère maintenant le cas des graphes de degré maximum k avec k impair. Les meilleurs graphes $\mathcal{F}_{n,k}$-universels induits connus à ce jour sont issus des graphes $\mathcal{F}_{n,k+1}$-universels induits $U_{n,k+1}$ car il n'existe pas de constructions spécifiques aux graphes de degré impair. En effet, la meilleure partition pour la construction d'un graphe $\mathcal{F}_{n,k}$-universel induit consiste en la partition des graphes de degré k en $\lceil k/2 \rceil$ graphes de degré 2. Le meilleur graphe $\mathcal{F}_{n,k}$-universel induit que l'on peut obtenir par ce procédé est donc le graphe $U_{n,k+1}$ décrit précédemment. Malheureusement, il existe un facteur multiplicatif $O(n^{1/2})$ entre l'ordre de ce graphe et l'ordre minimal d'un graphe $\mathcal{F}_{n,k+1}$-universel induit [24]. En fait, il est possible de réduire significativement cet écart, d'un facteur multiplicatif de presque $O(n^{1/k})$, en utilisant un résultat récent sur les graphes $\mathcal{F}_{n,k}$-universel [4] combiné avec une construction de [27] permettant de passer d'un graphe universel à un graphe universel induit.

Théorème 11. *Soit $k \geqslant 3$ un entier impair. Il existe un graphe $\mathcal{F}_{n,k}$-universel induit $U_{n,k}$ tel que*

$$|V(U_{n,k})| = c_1(k) n^{\lceil k/2 \rceil - 1/k} \log^{2+2/k} n \text{ et } |E(U_{n,k})| = c_2(k) n^{k-2/k} \log^{4+4/k} n$$

Démonstration. Pour $k \geqslant 3$, il existe un graphe $\mathcal{F}_{n,k}$-universel $H_{n,k}$ ayant n sommets et au plus $c(k) n^{2-2/k} \log^{4/k} n$ arêtes où $c(k)$ a une valeur indépendante de n [4]. De plus, $H_{n,k}$ est implicitement décrit comme étant régulier et donc a un degré maximum au plus $c(k) n^{1-2/k} \log^{4/k} n$.

On utilise ensuite un résultat de Chung [27] permettant de construire un graphe universel induit à partir d'un graphe universel. Ce résultat se base sur l'arboricité des graphes de la famille. Pour une famille $\mathcal{F}$ dont les graphes ont une arboricité r, on peut construire un graphe $\mathcal{F}$-universel induit U à partir d'un graphe $\mathcal{F}$-universel G tel que :

$$|V(U)| = \sum_{v \in V(G)} (d_G(v) + 1)^r \text{ et } |E(U)| = \sum_{uv \in E(G)} (d_G(u) + 1)^r d_G(v)^{r-1}.$$

À partir de la définition de l'arboricité, il est facile de déduire que les graphes de degré $k \equiv 1 \pmod 2$ ont une arboricité au plus $\lceil k/2 \rceil$. Par conséquent, on peut construire un

graphe $\mathcal{F}_{n,k}$-universel induit $U_{n,k}$ tel que :

$$\begin{aligned}
|V(U_{n,k})| &= \sum_{v\in V(H_{n,k})} (d_{H_{n,k}}(v)+1)^{\lceil k/2\rceil} \\
&\leqslant |V(H_{n,k})|(2d_{H_{n,k}})^{\lceil k/2\rceil} \\
&\leqslant n(2c(k)n^{1-2/k}\log^{4/k} n)^{\lceil k/2\rceil} \\
&\leqslant c_1(k)n^{\lceil k/2\rceil-1/k}\log^{2+2/k} n \text{ , où } c_1(k)=(2c(k))^{\lceil k/2\rceil}
\end{aligned}$$

$$\begin{aligned}
|E(U_{n,k})| &= \sum_{uv\in E(H_{n,k})} (d_{H_{n,k}}(u)+1)^{\lceil k/2\rceil} d_{H_{n,k}}(v)^{\lceil k/2\rceil-1} \\
&\leqslant |E(H_{n,k})|(2d_{H_{n,k}})^{\lceil k/2\rceil}(d_{H_{n,k}})^{\lceil k/2\rceil-1} \\
&\leqslant c(k)n^{2-2/k}\log^{4/k} n(2c(k)n^{1-2/k}\log^{4/k} n)^{\lceil k/2\rceil}(c(k)n^{1-2/k}\log^{4/k} n)^{\lceil k/2\rceil-1} \\
&\leqslant c_2(k)n^{k-2/k}\log^{4+4/k} \text{ , où } c_2(k)=(2c(k))^{k+1}.
\end{aligned}$$

□

3.3 Graphes universels induits pour les graphes orientés

3.3.1 Graphes universels induits pour les graphes orientés de degré borné

La construction de la partie 3.2.2 peut facilement se généraliser à la famille $\mathcal{O}_{n,k}$ des graphes orientés de degré entrant et sortant au plus k.

Théorème 12. *Il existe un graphe $\mathcal{O}_{n,k}$-universel induit $\overrightarrow{O_{n,k}}$ tel que*

$$|V(\overrightarrow{O_{n,k}})| = (1+o(1))\,(3n)^k \text{ et } |E(\overrightarrow{O_{n,k}})| = (2+o(1))\,(3n)^{2k-1}.$$

La preuve se base sur une partition des arêtes des graphes de degré entrant et sortant au plus k en k graphes de degré entrant et sortant au plus un [77]. De manière similaire à la méthode employée précédemment pour les graphes de degré pair, on ramène le problème à la construction d'un graphe $\mathcal{O}_{n,1}$-universel induit.

Lemme 7. *Le graphe $\overrightarrow{U_{n,1}}$ représenté dans la figure 24 est un graphe $\mathcal{O}_{n,1}$-universel induit.*

Démonstration. Soit $\overrightarrow{G}$ un graphe de $\mathcal{O}_{n,1}$. Les composantes connexes de $\overrightarrow{G}$ sont soit des chemins dirigés soit des circuits. On encastre $\overrightarrow{G}$ dans $\overrightarrow{U_{n,1}}$ en utilisant quasiment la même méthode que pour l'encastrement d'un graphe G dans $U_{n,2}$ décrit dans la section 3.2.1. Le graphe $\overrightarrow{U_{n,1}}$ tout comme le graphe $U_{n,2}$ est constitué de tuiles. Néanmoins, ces

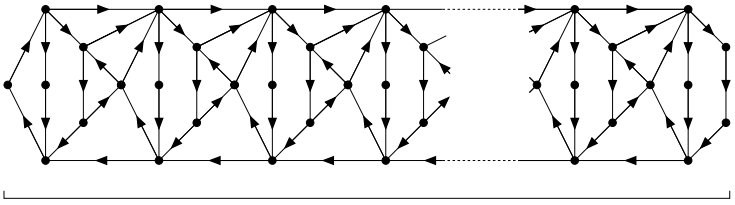


Figure 24 – Le graphe $\mathcal{O}_{n,1}$-universel induit $\overrightarrow{U_{n,1}}$.

tuiles sont légèrement différentes. En effet, elles sont constituées d'un circuit de 5 sommets plus d'un sommet reliant deux sommets à distance 2 du circuit pour un total de 6 sommets contrairement au tuiles de $U_{n,2}$ qui consistent en des cycles de longueur 5. De plus, les tuiles de $\overrightarrow{U_{n,1}}$ sont reliées entre elles par plus d'arêtes que celles de $U_{n,2}$. Malgré ces différences, il est possible d'encastrer dans $\overrightarrow{U_{n,1}}$ les composantes connexes de taille 1 et 2 en utilisant les mêmes sommets que ceux utilisés dans $U_{n,2}$. Les composantes connexes de taille supérieure peuvent être encastrées suivant le schéma suivant qui diffère légèrement de celui utilisé pour $U_{n,2}$.

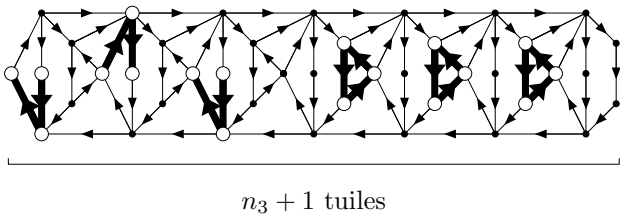


Figure 25 – L'encastrement orienté des n_3 composantes connexes de taille 3, utilisant n_3+1 tuiles.

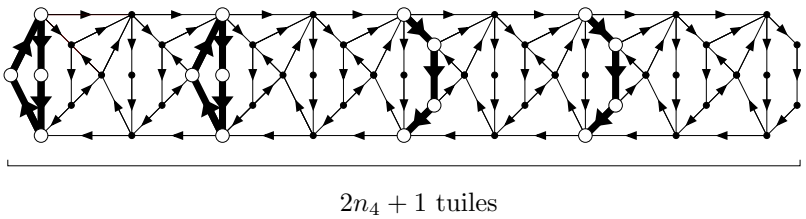


Figure 26 – L'encastrement orienté des n_4 composantes connexes de taille 4, utilisant $2n_4+1$ tuiles.

Il est facile de vérifier que l'encastrement décrit est induit et ce pour les même raisons que pour celui de $U_{n,2}$. Au final, le graphe $\overrightarrow{U_{n,1}}$ contient $3n + 30$ sommets et $6n + 60$ arcs puisqu'il possède le même nombre de tuiles que $U_{n,2}$. □

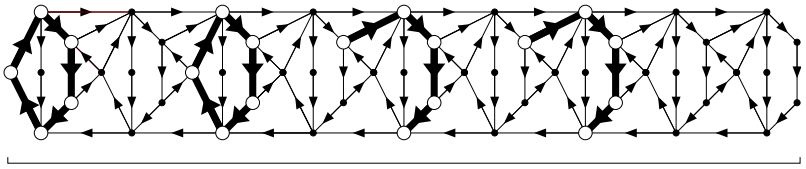


Figure 27 – L'encastrement orienté des n_5 composantes connexes de taille 5, utilisant $2n_5$ tuiles.

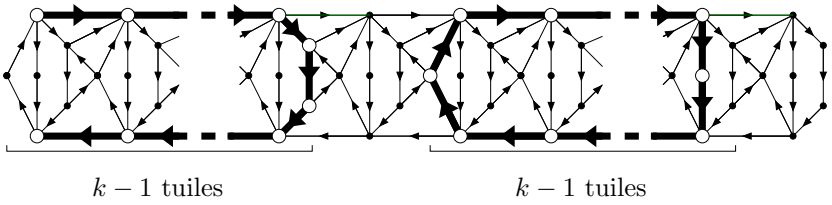


Figure 28 – Pour $k \geqslant 3$, l'encastrement orienté des n_{2k} composantes connexes de taille $2k$, utilisant kn_{2k} tuiles.

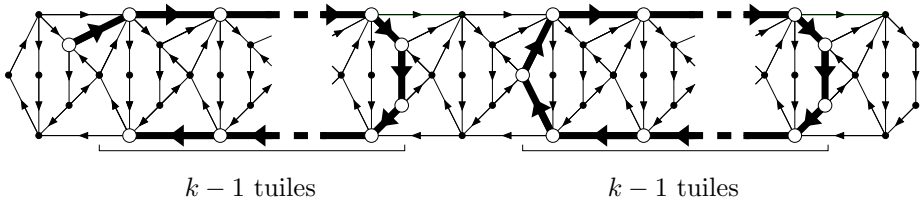


Figure 29 – Pour $k \geqslant 3$, l'encastrement orienté des n_{2k+1} composantes connexes de taille $2k+1$, utilisant kn_{2k+1} tuiles.

Démonstration du théorème 12 Pour $k > 0$, il existe une partition des arêtes des graphes de $\mathcal{O}_{n,k}$ en k graphes de $\mathcal{O}_{n,1}$. D'après le lemme 7, il existe un graphe $\mathcal{F}_{n,2}$-universel induit ayant $3n + O(1)$ sommets et $6n + O(1)$ arcs. La composition des graphes universels induits pouvant se faire de la même façon que dans le cas non orienté, on en déduit que :

$$\begin{aligned} |V(\overrightarrow{U_{n,k}})| &= |V(\overrightarrow{U})|^k \\ &= (3n)^k + o(n^k) \\ |E(\overrightarrow{U_{n,k}})| &= k|V(\overrightarrow{U})|^{2k-2}|E(\overrightarrow{U})| \\ &= 2.3^{2k-1}n^{k-2}n^{2k-1} + o(n^{2k-1}). \end{aligned}$$

□

3.3.2 Passer d'un graphe universel à un graphe universel orienté

On a montré avec le théorème 12 qu'il était possible de construire un graphe universel pour la famille des orientations des graphes de $\mathcal{F}_{n,2}$ en ajoutant des sommets dans le graphe universel et en orientant ses arêtes. Soit $\mathcal{F}$ une famille de graphes et $\overrightarrow{\mathcal{F}}$ une famille d'orientations de graphes de $\mathcal{F}$. Il semble naturel de se demander s'il est possible d'obtenir un graphe $\overrightarrow{\mathcal{F}}$-universel induit $\overrightarrow{U}$ à partir d'un graphe $\mathcal{F}$-universel induit U. En fait, une telle construction est possible si l'on dispose d'un graphe $\overrightarrow{H}$ qui soit $\overrightarrow{\mathcal{F}}$-*universel pour l'homomorphisme*, c'est-à-dire un graphe $\overrightarrow{H}$ tel que pour tout $\overrightarrow{G} \in \overrightarrow{\mathcal{F}}$ il existe un homomorphisme de $\overrightarrow{G}$ vers $\overrightarrow{H}$. Par exemple, le circuit de longueur trois est universel pour l'homomorphisme pour la famille des forêts orientées. Le graphe $\overrightarrow{U}$ peut être obtenu par l'utilisation d'un produit spécial des graphes $\overrightarrow{H}$ et U. Le *produit direct orienté* d'un graphe non-orienté U et d'un graphe orienté $\overrightarrow{H}$ est un graphe orienté $U \times \overrightarrow{H}$ ayant pour ensemble de sommets $V(U \times \overrightarrow{H}) = V(U) \times V(\overrightarrow{H})$ et pour ensemble d'arcs $E(U \times \overrightarrow{H}) = \Big\{[(x,u),(y,v)] \mid xy \in E(U) \text{ et } uv \in E(\overrightarrow{H})\Big\}$.

Théorème 13. *Soit $\mathcal{F}$ une famille de graphes et $\overrightarrow{\mathcal{F}}$ une famille d'orientations de $\mathcal{F}$. Si U est $\mathcal{F}$-universel induit et $\overrightarrow{H}$ est $\overrightarrow{\mathcal{F}}$-universel pour l'homomorphisme alors $U \times \overrightarrow{H}$ est $\overrightarrow{\mathcal{F}}$-universel induit.*

Démonstration. Il suffit de montrer que l'on peut encastrer n'importe quel graphe $\overrightarrow{G} \in \overrightarrow{\mathcal{F}}$ dans $U \times \overrightarrow{H}$. Soit $v \in \overrightarrow{G}$. Il existe un homomorphisme h de $\overrightarrow{G}$ dans $\overrightarrow{H}$ puisque $\overrightarrow{H}$ est $\overrightarrow{\mathcal{F}}$-universel pour l'homomorphisme. Si on ne tient pas compte de l'orientation, il existe un encastrement de $\overrightarrow{G}$ dans U puisque U est $\mathcal{F}$-universel induit. Pour chaque $v \in V(\overrightarrow{G})$, on note $h(v)$ le sommet correspondant à v dans $\overrightarrow{G}$ pour l'homomorphisme h et $u(v)$ le sommet dans lequel v est encastré dans U. L'encastrement de $\overrightarrow{G}$ dans $U \times \overrightarrow{H}$ consiste à encastrer chaque sommet $v \in V(\overrightarrow{G})$ dans le sommet $(u(v), h(v))$ dans $U \times \overrightarrow{H}$.

Cet encastrement est valide dans le sens où pour chaque couple (x, y) de sommets reliés par un arc dans $\overrightarrow{G}$, il existe l'arc correspondant $((u(x), h(x)), (u(y), h(y)))$ dans $U \times \overrightarrow{H}$. En effet, on a $u(x)u(y) \in E(U)$ à cause de l'encastrement de $\overrightarrow{G}$ dans U et $(h(x), h(y)) \in E(\overrightarrow{H})$ d'après la définition d'un homomorphisme.

L'encastrement est aussi induit. En effet, si x et y ne sont pas adjacents dans $\overrightarrow{G}$ alors $u(x)$ et $u(x)$ ne le sont pas dans U car l'encastrement de $\overrightarrow{G}$ dans U est induit. Par conséquent, $(u(x), h(x))$ et $(u(y), h(y))$ ne sont pas adjacents dans $U \times \overrightarrow{H}$ par définition du produit direct.

Finalement, l'encastrement est valide et induit et donc le graphe $U \times \overrightarrow{H}$ est $\overrightarrow{\mathcal{F}}$-universel induit. □

Pour une famille $\overrightarrow{\mathcal{F}}$ d'orientations de $\mathcal{F}$, le problème consistant à trouver un graphe $\overrightarrow{\mathcal{F}}$-universel pour l'homomorphisme est directement relié à la coloration orientée des graphes

de $\mathcal{F}$. En effet, si toutes les orientations d'un graphe admettent un homomorphisme vers un graphe d'ordre k alors le graphe est k-coloriable. Certaines preuves de colorations orientées utilisent ce fait pour prouver des majorants pour le nombre chromatique orienté pour des familles de graphes spécifiques. Par exemple, il existe un graphe universel pour l'homomorphisme pour la famille des orientations des graphes planaires ayant 80 sommets [22], un de 7 sommets pour les graphes de largeur arborescente 2 [87], un de 5 sommets pour les graphes de degré 2 [87], et un de 11 sommets pour les graphes de degré 3 [87]. Plus généralement, il existe un graphe universel pour l'homomorphisme de $2k^2 + 2^k$ sommets pour la famille de graphes de nombre chromatique acyclique au plus k [80], c'est-à-dire les graphes que l'on peut colorier proprement avec k couleurs de telle sorte qu'il n'y ait pas de cycles bicoloriés. Par conséquent, les familles de graphes ayant un nombre chromatique acyclique borné comme celles des graphes de genre borné [5] et de largeur arborescente bornée [87] ont un graphe universel pour l'homomorphisme de taille bornée. Cela implique que pour les familles, il est possible de construire un graphe universel induit pour leurs orientations ayant un nombre de sommets égal à une constante fois le nombre de sommets du graphe universel induit non-orienté.

3.4 Schéma d'adjacence pour les graphes de genre borné

On considère dans cette partie la famille $\mathcal{F}_{n,g}$ des graphes de genre d'Euler g ayant n sommets. On utilise les partitions d'arêtes du deuxième chapitre afin de construire des schémas d'adjacence pour $\mathcal{F}_{n,g}$. D'après le fait 1, les graphes de $\mathcal{F}_{n,g}$ ont une arboricité au plus $O(\sqrt{g})$. Selon [9], il est donc possible de construire un schéma d'adjacence en $O(\sqrt{g}\log n)$ pour cette famille. En utilisant la partition du corollaire 1, on peut significativement améliorer ce résultat préliminaire.

Corollaire 3. *Il existe un schéma d'adjacence pour $\mathcal{F}_{n,g}$ avec des étiquettes de taille bornée par $3\log n + O(\log^* n) + O(\sqrt{g}\log g)$ bits.*

Démonstration. Il suffit d'utiliser le résultat d'Alstrup et Rauhe [9] pour le schéma d'adjacence compact pour la famille des forêts de taille au plus n avec des étiquettes de taille bornée par $\log n + O(\log^* n)$ bits. Cela nous donne un codage des arêtes de A_1, A_2 et A_3 en $3\log n + O(\log^* n)$ bits. On peut coder A_0 en utilisant $O(\sqrt{g}\log g)$ bits car le graphe induit par A_0 est un graphe de genre g ayant au plus $O(g)$ sommets et donc son arboricité est en au plus $O(\sqrt{g})$. □

On verra dans le cinquième chapitre, que l'on peut obtenir un schéma encore plus compact en utilisant la partition des graphes de genre d'Euler g en deux graphes de largeur arborescente dépendant de g.

3.5 Conclusion

Dans la première partie du chapitre, on a prouvé qu'un graphe $\mathcal{F}_{n,2}$-universel induit minimal contient au moins $11n/6 + \Omega(1)$ sommets et au plus $5n/2 + O(1)$ sommets. Une question naturelle qui vient à l'esprit est de savoir s'il est possible de réduire l'écart entre $11/6$ et $5/2$ pour la constante multiplicative. Du fait de la simplicité de la structure des graphes étudiés, les graphes de degré 2, la question peut sembler facile. En fait, le problème est relativement difficile et améliorer le majorant pour cette constante n'est pas évident car cela nécessiterait une construction totalement différente. Malgré tout, on conjecture qu'il est possible d'améliorer à la fois le minorant et le majorant pour ce problème et qu'il existe un graphe universel induit d'ordre minimum ayant $2n + O(1)$ sommets.

Pour le cas des graphes de degré k impair, si on ne tient pas compte du facteur multiplicatif polylogarithmique, il reste un facteur multiplicatif de $n^{1/2-1/k}$ entre le majorant et le minorant pour la taille du graphe $\mathcal{F}_{n,k}$-universel induit. Un problème intéressant pourrait être de réduire cet écart, particulièrement pour des valeurs de k élevées. Une amélioration sensible du majorant requerrait une approche totalement différente de celle utilisée dans ce chapitre. En effet, on utilise une construction déduite d'un graphe universel de la famille $\mathcal{F}_{n,k}$ et ce graphe utilisé est optimal à un facteur polylogarithmique près [4].

Le graphe $\mathcal{F}_{n,k}$-universel induit pour k pair a un degré maximum $4^{k/2}$ dépendant seulement de k. En revanche, pour k impair, le graphe $\mathcal{F}_{n,k}$-universel induit a un degré maximum $c_2(k)n^{k-1-2/k}\log^{4+4/k} n$ et qui dépend donc à la fois de k et n. Considérant le fait que dans le cas où k est pair notre construction est quasi optimale alors qu'elle ne l'est pas pour k impair, on conjecture qu'il existe une fonction $f(k)$ telle que l'existence d'un graphe $\mathcal{F}_{n,k}$-universel induit implique l'existence d'un graphe $\mathcal{F}_{n,k}$-universel de même ordre mais ayant un degré borné par $f(k)$.

Pour le cas $k = 2$ et plus généralement le cas où k est pair, on a montré que le graphe universel donne un schéma d'adjacence efficace. Malheureusement, pour le cas k impair, le graphe universel induit ne donne pas de schéma d'adjacence efficace. En effet, l'étiquetage peut prendre un temps sur-linéaire et surtout l'adjacence ne peut être testée de manière rapide. Cela montre que les deux approches du problème, combinatoires et algorithmiques, ne sont donc pas totalement équivalentes puisque la version algorithmique du problème prend en compte deux paramètres supplémentaires, les complexités en temps des fonctions d'adjacence et d'étiquetage. C'est pour cela que par la suite on considérera le problème uniquement sous sa forme algorithmique.

Chapitre 4

Schémas d'adjacence utilisant le parcours bondissant

4.1 Introduction

Représenter un graphe en mémoire est un problème basique mais néanmoins fondamental dans le domaine des structures de données. Il existe deux manières élémentaires d'accomplir cette tache, utiliser une matrice d'adjacence (une matrice de booléens qui code pour chaque paire de sommets la présence ou non d'une arête entre les deux sommets) ou bien des listes d'adjacence (pour chaque sommet, on stocke une liste de ses voisins). Cette deuxième représentation est quasi-optimale en terme d'espace mémoire utilisé mais les requêtes d'adjacence requièrent une recherche dans une liste. En revanche, dans la première représentation, les requêtes d'adjacence se font en temps constant mais par contre l'espace de stockage est sur-linéaire en la taille du graphe.

Une autre technique, connue sous les noms de *représentation distribuée* et de *schéma d'étiquetage*, consiste à distribuer localement la représentation globale du graphe parmi ses sommets. Le but est de stocker localement une information utile sur chaque sommet et de la rendre commodément accessible. Pour cela, on se fixe une requête portant sur un ou plusieurs sommets du graphe à laquelle la représentation distribuée doit permettre de répondre localement. Plus précisément, on attribue une étiquette binaire à chaque sommet du graphe telle que la requête choisie puisse être calculée en utilisant seulement les étiquettes des sommets concernés. De plus, on force les étiquettes à être distinctes afin d'avoir en plus de l'information encodée, un identifiant pour chaque sommet.

Concrètement, pour le cas de la requête d'adjacence, il doit être possible de tester la présence d'une arête entre deux sommets en examinant seulement leurs étiquettes. Cette représentation distribuée connue sous les noms de schéma d'adjacence et de représentation implicite est équivalente à la notion de graphe universel induit comme indiqué dans le chapitre précédent. Il est possible grâce à cette représentation de manipuler le graphe en utilisant uniquement les étiquettes du schéma puisque le graphe peut entièrement être

recalculé en interrogeant la représentation implicite pour toutes les paires de sommets. Le problème est de minimiser la longueur des étiquettes tout en gardant un calcul des requêtes rapide.

Les représentations distribuées sont largement utilisées dans le domaine du calcul distribué comme explicité dans [50, 66]. Elles ont aussi des applications dans les moteurs de recherches XML [1] ainsi que pour les applications distribuées telles que les réseaux pair-à-pair ou le routage dans les réseaux. Dans ce cadre, de nombreuses représentations distribuées ont été développées pour des requêtes diverses et variées. Les résultats existants concernant la représentation implicite de graphe ont déjà été abordés dans le chapitre précédent puisque ils sont directement liés à la notion de graphe universel. On conseille donc au lecteur de se référer à l'introduction du chapitre précédent afin d'en savoir plus sur ceux-ci. Les autres types de requêtes étudiées comprennent en autre le routage [37, 38, 44, 67, 92], les requêtes de distance [12, 49], la connectivité et les flots [60, 65]. Plus particulièrement pour les arbres enracinés, le cas des requêtes d'ascendance (déterminer si un sommet est l'ancêtre d'un autre) [2, 1], de frères [1], de plus petit ancêtre commun [8] et de distance bornée [59, 7] ont été étudiés. Un récapitulatif des résultats connus sur les schémas d'adjacence peut être trouvé dans [50].

Dans ce chapitre, on s'intéresse à un schéma d'adjacence pour les arbres enracinés de degré interne borné par $\hat{d}$, c'est-à-dire les arbres dont les sommets internes ont au plus $\hat{d}$ fils étant internes. Cette sous-famille des arbres contient la famille des chenilles (arbres dont les sommets internes induisent un chemin) pour $\hat{d} \leqslant 2$ et des arbres de degré borné par $\hat{d}$. On décrit tout d'abord, en guise d'introduction au schéma global, un schéma spécifique aux chenilles qui utilise des étiquettes de $\log n + 6$ bits pour les chenilles de n sommets. Bien que ce schéma reste relativement simple, il est le premier schéma connu pour une famille de graphes de degré non borné avec des étiquettes de $\log n + O(1)$ bits. Ensuite, on présente un schéma pour les arbres de degré interne borné par $\hat{d}$ ayant n sommets avec des étiquettes de $\log n + O(\log \hat{d})$ bits. Bien que ces résultats soit en eux-mêmes intéressants car les schémas connus pour les arbres ne se simplifient pas pour ces sous-familles, c'est surtout la technique utilisée qui est la plus importante. En effet, cette technique dite de "parcours bondissant" diffère grandement des techniques d'étiquetage précédemment connues et on peut espérer pouvoir l'utiliser sur d'autres familles de graphe. Ces résultats ont fait l'objet de présentations à des conférences internationales [20, 21].

4.2 Technique de parcours bondissant

4.2.1 Principes généraux du parcours bondissant

Afin d'introduire notre technique d'étiquetage, il convient d'abord de considérer l'étiquetage des chenilles. Soit T une chenille de n sommets dont le chemin interne est $x_1, \ldots, x_k$ et la j-ème feuille de x_i est $y_{i,j}$.

Une première approche naïve pour étiqueter T consiste à donner à chaque sommet x_i

l'étiquette $(i, 0)$ et à chaque sommet $y_{i,j}$ l'étiquette $(i, 1)$. Clairement cet étiquetage permet le test de l'adjacence mais il n'est pas valide car il ne respecte pas la propriété d'unicité des étiquettes du graphe. Un schéma correct peut être obtenu en assignant à chaque sommet $y_{i,j}$ l'étiquette (i, j). L'étiquetage ainsi obtenu n'est pas optimal puisque chaque paire d'entiers positifs (i, j) avec $i+j \leqslant n$ est assignée par le schéma pour au moins une chenille. De telles paires sont au nombre de $(n/2)^2$, ce qui donne des étiquettes d'au moins $2\log n - O(1)$ bits.

Une deuxième approche possible, qui est un tout petit peu moins triviale, consiste à assigner à chaque sommet $y_{i,j}$ la paire (r_i, j) et à chaque sommet x_i l'entier r_i où r_i est le rang du nombre de feuilles de x_i. De cette manière, moins de bits sont utilisés pour encoder r_i quand x_i a beaucoup de feuilles, laissant ainsi de la place pour le codage de l'index j. Puisque $j \leqslant n/r_i$, la longueur des étiquettes se réduit à $\lceil \log r_i \rceil + \lceil \log(n/r_i) \rceil + \lceil \log l \rceil$ où l est la longueur du plus petit des deux champs de la paire (r_i, j). En effet, les champs de cette paire ont une longueur variable, et donc on a besoin d'encoder la valeur $l \leqslant \log \min \{r_i, j\}$ pour pouvoir extraire les deux champs. L'encodage de cette valeur l nécessite $\log\log \sqrt{n}$ bits puisque $\min \{r_i, j\}$ peut prendre toutes les valeurs entières comprises entre 1 et $\sqrt{n}$. On obtient donc des étiquettes d'au moins $\log n + \Omega(\log\log n)$ bits. De plus, ce schéma ne donne pas l'adjacence entre deux sommets du chemin.

Une troisième solution possible consiste à utiliser une quelconque décomposition récursive du graphe, comme celle réalisée dans [9]. Cependant, toute décomposition ayant un nombre non constant de niveaux de récursion produit des étiquettes ayant un nombre non constant de champs, ce qui mène à des étiquettes de $\log n + \omega(1)$ bits. De plus, les schémas existants pour les arbres [9] ne se simplifient pas dans le cas d'arbre de degré borné, les étiquettes étant toujours de $\log n + \omega(1)$ bits. Pour obtenir des étiquettes de longueur optimale $\log n + O(1)$, il convient donc de considérer une autre technique. Cette nouvelle technique appelée parcours bondissant est constituée de trois étapes principales :

1. Sélectionner un parcours approprié de l'arbre (ou plus généralement du graphe),
2. Associer à chaque sommet x de l'information code(x) nécessaire au test d'adjacence,
3. Effectuer le parcours de l'arbre et assigner dans l'ordre du parcours choisi des étiquettes binaires de valeurs croissantes mais pas nécessairement consécutives aux sommets (effectuant ainsi des "bonds" dans l'étiquetage) permettant de coder code(x) pour x.

Le test d'adjacence entre deux sommets x et y est évidemment effectué sur la base de code(x) et code(y). En fait, ce sont les bonds effectués dans l'étiquetage du graphe à l'étape 3 qui permettent le stockage de l'information code(x) pour chaque sommet de x. On assigne à chaque sommet x un intervalle de valeurs $I(x)$ dans l'ensemble des étiquettes possibles. Pour plus de simplicité, on confond lorsque cela ne prête pas à confusion, les étiquettes avec leurs valeurs entières associées et donc on considère des intervalles d'entiers. Ces intervalles sont assignés de telle sorte qu'ils soient deux à deux disjoints et ordonnés dans le même sens que les sommets correspondants dans le parcours choisi. L'étiquette $\ell(x)$ est contenue dans l'intervalle $I(x)$ et c'est sa position exacte dans l'intervalle qui va permettre de coder

code(x). La longueur de l'intervalle $I(x)$ dépend donc de l'information code(x) à stocker. Généralement, on peut déduire à partir de code(x) un intervalle $B(x)$ contenant toutes les étiquettes de tous les voisins de x qui lui sont postérieurs dans le parcours, et seulement ces sommets. Cette information permet le test d'adjacence car x et y sont adjacents si et seulement si $\ell(x) \in B(y)$ ou $\ell(y) \in B(x)$.

4.2.2 Code suffixe

Afin de pouvoir coder efficacement l'information code(x) pour chaque sommet x de l'arbre on a besoin de la notion de code suffixe.

On assume un modèle RAM de calcul sur des mots de $\Omega(\log n)$ bits. Dans ce modèle, les opérations arithmétiques standards sur les mots de longueur $O(\log n)$ peuvent être effectuées en temps constant. Cela inclue les additions, comparaisons, masques binaires, décalages de mot et recherches du bit de poids faible et de poids fort.

Soit A un mot binaire, on note $|A|$ sa longueur. Pour un mot binaire B, $A \circ B$ dénote la concaténation de A suivi de B.

Soit $x \in \mathbb{N}$, on dénote par $\lg x = \log \max\{x, 1\}$ et par $\mathrm{bin}(x)$ sa représentation binaire standard. On a $|\mathrm{bin}(x)| = \lfloor \lg x \rfloor + 1$.

Un *code* est un ensemble de mots. Un code C est dit *suffixe* si aucun mot de C n'est le suffixe d'un autre mot de C. Une propriété basique des codes suffixes est que l'on peut les composer afin d'obtenir d'autres codes suffixes. En effet si $C_1, C_2, \cdots, C_k$ sont des codes suffixes, alors le code $C_1 \circ C_2 \circ \cdots \circ C_k = \{w_1 \circ w_2 \circ \cdots \circ w_k \mid \forall i \leqslant k, w_i \in C_i\}$ est lui aussi suffixe. Un des codes suffixes les plus simple est défini par $\mathrm{code}_0(x) = 1 \circ 0^x$, où 0^x est le mot binaire composé de x zéros. On peut étendre ce code à des codes plus succincts en définissant récursivement le code $\mathrm{code}_{i+1}(x) = \mathrm{bin}(x) \circ \mathrm{code}_i(|\mathrm{bin}(x)| - 1)$ pour $i \geqslant 0$. Il est facile de vérifier que pour $i \geqslant 0$, le code code_i est suffixe. À titre d'exemple, $\mathrm{code}_0(5) = 100000$, $\mathrm{code}_1(5) = 101\,100$, et $\mathrm{code}_2(5) = 101\,10\,10$.

La principale utilisation des codes suffixes est le codage de valeurs entières. Si un mot w a $\mathrm{code}_i(x)$ comme suffixe, alors x peut être extrait de w en temps $O(i)$ grâce à des recherches de bit de poids faible, des décalages et des masques binaires. De même, n'importe quelle suite d'entiers $x_1, \ldots, x_k$ peut être stockée comme suffixe $\mathrm{code}_i(x_1) \circ \cdots \circ \mathrm{code}_i(x_k)$ et extrait en temps $O(ik)$.

Dans ce chapitre, on va faire l'usage de code_i pour $i \in \{0, 1, 2\}$. Il est facile de vérifier que pour tout $x \in \mathbb{N}$, $|\mathrm{code}_0(x)| = x + 1$, $|\mathrm{code}_1(x)| = 2\lfloor \lg x \rfloor + 2$, et $|\mathrm{code}_2(x)| = \lfloor \lg x \rfloor + 2\lfloor \lg \lfloor \lg x \rfloor \rfloor + 3$.

Fait 3. *Soit w un mot binaire, et z un entier. Il existe un entier $x \in [z, z + 2^{|w|})$ tel que w soit suffixe de* $\mathrm{bin}(x)$*. De plus, cet entier x peut être calculé en temps constant.*

Démonstration. On note $\mathrm{val}(w)$ l'entier x tel que $w = \mathrm{bin}(x)$.

On peut observer que pour tous mots A et B, $\mathrm{val}(A \circ B) = \mathrm{val}(A) \cdot 2^{|B|} + \mathrm{val}(B)$, et $\mathrm{val}(A) < 2^{|A|}$. On considère $u = \left\lfloor z/2^{|w|} \right\rfloor$ et $v = z \bmod 2^{|w|}$ tel que $z = u \cdot 2^{|w|} + v$. On fixe

$b = 0$ si $\text{val}(w) \geqslant v$ et $b = 1$ sinon. L'entier x est défini par $\text{bin}(x) = \text{bin}(u + b) \circ w$ qui a clairement w comme suffixe. Le mot x peut être calculé en temps constant grâce à des décalages, des masques binaires et des recherches de bit de poids fort.

Il reste à vérifier que $x \in [z, z+2^{|w|})$. On a $x = (u+b)\cdot 2^{|w|} + \text{val}(w) = z - v + b\cdot 2^{|w|} + \text{val}(w)$.

Si $b = 0$, alors $x = z - v + \text{val}(w) \leqslant z + \text{val}(w) < z + 2^{|w|}$. Pour $b = 0$, $\text{val}(w) \geqslant v$, et donc $x \geqslant z$.

Si $b = 1$, alors $x = z - v + 2^{|w|} + \text{val}(w) > z + \text{val}(w) \geqslant z$ puisque $v < 2^{|w|}$. Pour $b = 1$, $\text{val}(w) < v$, et donc $z - v + 2^{|w|} + \text{val}(w) < z + 2^{|w|}$. □

4.3 Schéma d'adjacence pour les chenilles

Dans cette section, on décrit le schéma spécifique pour les chenilles. L'intérêt premier est de décrire l'utilisation de la technique de parcours bondissant sur une famille simple avant d'aborder un cas plus complexe. Le schéma obtenu est néanmoins intéressant car les étiquettes obtenues sont à peine plus longues que des identifiants de $\lceil \log n \rceil$ bits.

Théorème 14. *La famille des chenilles de n sommets admet un schéma d'adjacence avec des étiquettes de taille au plus $\lceil \log n \rceil + 6$ bits. De plus, le test d'adjacence peut s'effectuer en temps constant et l'étiquetage en temps $O(n)$.*

4.3.1 Description de la fonction d'étiquetage

On considère une chenille T de n sommets. On note $P = \{x_1, \ldots, x_k\}$ le chemin induit par l'ensemble des sommets internes de T. Pour tout i, on note $Y_i = \{y_{i,1}, \ldots, y_{i,d_i}\}$ l'ensemble des feuilles du sommet x_i, avec $d_i = 0$ si $Y_i = \varnothing$. Pour un intervalle d'entiers $I = [a, b[$, on fixe $I_[= a$ et $I_) = b$. La taille de l'intervalle I est $|I| = b - a$.

On utilise la technique de parcours bondissant pour la construction du schéma d'adjacence de T. Comme indiquée dans l'introduction, la première étape de cette technique consiste à choisir un parcours. Le parcours choisi est un parcours préfixe enraciné en x_1 où pour chaque sommet x_i de P, on parcourt les fils de x_i en commençant par les feuilles $y_{i,1} \cdots y_{i,d_i}$ et en finissant par le sommet interne x_{i+1}. La deuxième étape consiste à fixer l'information $\text{code}(v)$ que va stocker chaque sommet v de T pour le test d'adjacence. L'étiquette $\ell(x_i)$ de chaque sommet interne x_i va coder trois entiers :

- La longueur d'un intervalle $I(x_i)$ contenant l'étiquette $\ell(x_i)$.
- La longueur d'un intervalle $C(x_i)$ contenant les étiquettes des sommets de Y_i.
- La longueur d'un intervalle $I(x_{i+1})$ contenant l'étiquette $\ell(x_{i+1})$.

Grâce au codage de ces trois valeurs, il va être possible déduire de l'étiquette $\ell(x_i)$, les intervalles $C(x_i)$ et $I(x_{i+1})$ car l'intervalle $C(x_i)$ va commencer à la valeur $\ell(x_i)$ ($C_[(x_i) = \ell(x_i)$) et que $I(x_{i+1})$ est contigu à $C(x_i)$ ($C_)(x_i) = I_[(x_{i+1})$). Il va donc être possible de déterminer l'adjacence car tester si un sommet v est fils de x_i va se résumer à tester l'appartenance de $\ell(v)$ aux intervalles $C(x_i)$ ou $I(x_{i+1})$. La complexité du schéma réside

dans le codage de ses trois entiers par l'étiquette $\ell(x_i)$. Afin de réduire l'information à encoder, on force à ce que les longueurs de $I(x_i)$ et $C(x_i)$ soient égales. Cela nous permet d'avoir à coder seulement deux entiers au lieu de trois. De plus, on va fixer ces valeurs à des puissances entières de deux. Ceci va nous permettre de stocker ces entiers en utilisant leur logarithme en base deux, réduisant ainsi énormément la taille d'information à stocker.

On note $p_i = 2^{-5}|I(x_i)| = 2^{-5}|C(x_i)|$. On note $\lceil n \rceil_2 = 2^{\lceil \log n \rceil}$ et $\lfloor n \rfloor_2 = 2^{\lfloor \log n \rfloor}$. La valeur p_i se définit comme suit :

$$\begin{aligned} p_{k+1} &= 0 \\ p_i &= \max\left\{ \lfloor \sqrt{p_{i+1}} \rfloor_2 , \lceil d_i \rceil_2 \right\} \end{aligned}$$

L'information stockée par chaque sommet x_i est composé de deux valeurs : p_i et p_{i+1}. Cette information va être stockée sous la forme du mot binaire suivant :

$$\begin{aligned} \mathrm{code}(x_i) &= 1\,00\cdots001\underbrace{\underbrace{00\cdots00}_{\log p_i \ '0'}}_{\lfloor \log(p_{i+1})/2 \rfloor \ '0'}1b1 \\ &= 1\,00\cdots001\quad\underbrace{\underbrace{00\cdots00}_{\lfloor \log(p_{i-1})/2 \rfloor \ '0'}}_{\log p_i \ '0'}\quad 0b1 \\ &\quad \text{où } b = \log(p_{i-1}) \pmod 2 \\ \mathrm{code}(y_{i,j}) &= 0 \end{aligned}$$

Il ne reste plus qu'à déterminer, durant la troisième étape, les bonds à faire dans l'étiquetage pour le codage de l'information pour chaque sommet. Clairement l'ensemble des mots $\mathrm{code}(v)$ forme un code suffixe. Chaque sommet v va stocker ce mot en tant que suffixe de son étiquette $\ell(v)$. On verra dans la prochaine partie comment cette information permet le test d'adjacence. Afin de pouvoir fixer le suffixe de $\ell(v)$ comme étant $\mathrm{code}(v)$, on va associer à chaque sommet v un intervalle $I(v)$ de longueur $2^{|\mathrm{code}(v)|}$ contenant seulement l'étiquette de v. Pour un mot binaire w et un entier z, on définit l'entier $\Phi(z, w)$ comme étant l'entier compris dans l'intervalle $[z, z + 2^{|w|}[$ calculé d'après le fait 3 tel que w soit suffixe de $\mathrm{bin}(\Phi(z, w))$. L'intervalle $I(v)$ et l'étiquette $\ell(v)$ sont calculés d'après le système d'équations suivant :

$$\begin{aligned} I_{[}(x_1) &= 0 \\ I_{]}(x_1) &= 2^5 p_1 \\ I_{[}(x_{i+1}) &= \ell(x_i) + 2^5 p_i \\ I_{]}(x_{i+1}) &= \ell(x_i) + 2^5(p_i + p_{i+1}) \\ I_{[}(y_{i,j}) &= \ell(x_i) + 2j - 1 \\ I_{]}(x_{i,j}) &= \ell(x_i) + 2j \\ \forall v \in \ell(v) &= \Phi(I_{[}(v), \text{code}(v)) \end{aligned}$$

Ces intervalles sont deux à deux disjoints puisque $2^5 p_i > 2d_i$ et sont bien de longueur $2^{|\text{code}(v)|}$. Au final, l'étiquetage de T peut se faire en temps $O(n)$ grâce à deux parcours de T, un premier parcours dans le sens inverse du parcours des sommets (de x_k à x_1) afin de calculer les valeurs p_i et un deuxième dans le sens normal (de p_1 à p_k) afin de calculer les étiquettes des sommets.

4.3.2 Description du test d'adjacence

On doit maintenant détailler précisément le test d'adjacence utilisé. Celui-ci est assez simple car il consiste grossièrement à extraire de l'étiquette des sommets les intervalles contenant leurs fils.

Fait 4. *Soit deux sommets u, v de T. Le test d'adjacence entre u et v peut être effectué en temps constant à partir de $\ell(u)$ et $\ell(v)$.*

Démonstration. On note X l'ensemble des sommets internes de T et Y son ensemble de feuilles. On va utiliser la condition d'adjacence suivante.

Les sommets u et v sont adjacents, $\ell(u) < \ell(v)$, si et seulement si :

- Cas 1 : $u \in X$ et $v \in Y$:
  u est adjacent à v si et seulement si $v \in C(u)$ et donc

$$\ell(u) < \ell(v) < \ell(u) + 2^5 p_i$$

- Cas 2 : $u \in X$ et $v \in X \Rightarrow w \exists i, j \in 1, \cdots, k$ tel que $u = x_i$ et $v = x_j$ $(i < j)$:
  x_i est adjacent à x_j si et seulement si
  $x_j \in I(x_{i+1})$ et donc

$$\ell(x_i) + p_i.2^5 \leqslant \ell(x_j) \leqslant \ell(p_i) + p_i.2^5 + p_{i+1}.2^5$$

A partir de l'"étiquette $\ell(u)$ d'un sommet u, on peut :

- Tester l'appartenance de u à X (dernier bit de $\ell(u)$ égal à 1).
- Tester l'appartenance de u à Y (dernier bit de $\ell(u)$ égal à 0).

- Si $u \in X$ ($\exists i$ tel que $u = x_i$) :
  - Obtenir $q_{i+1} = \log p_{i+1}/2$ et $r_{i+1} = \log p_{i+1} \mod 2$ ce qui nous permet de recalculer $\log p_{i+1} = 2q_{i+1} + r_{i+1}$ et donc p_{i+1} car p_{i+1} est une puissance de 2.
  - Obtenir $\log p_i$ et donc p_i.

Il est donc possible de tester la condition d'adjacence grâce à l'information stockée par chaque sommet. Il ne reste donc plus qu'à prouver la validité de cette condition.

- Cas 1 : $u \in X$ et $v \in Y \Rightarrow \exists i, j, k \in \{1, \cdots, k\}$ tel que $u = x_i$ et $v = y_{j,k}$:
  - u est adjacent à $v \Rightarrow \ell(u) < \ell(v) < \ell(u) + 2^5 p_i$

    On a $u = x_i$ et $v = y_{i,j}$. Par construction, on a :

    $$\ell(x_i) + 2j - 1 \leqslant \ell(y_{i,j}) \leqslant \ell(x_i) + 2j$$

    et donc

    $$\ell(x_i) < \ell(y_{i,j}) < \ell(x_i) + 2^5 p_i$$

    puisque $j \leqslant d_i \leqslant p_i$ et donc $2^5 p_i > 2d_i$

  - u n'est pas adjacent à $v \Rightarrow \ell(v) > \ell(u) + 2^5 p_i$

    On a donc $v = y_{k,j}$ avec $k > i$. Par construction, $\forall j, k$, on a :

    $$\ell(x_k) < \ell(y_{k,j}) < \ell(x_k) + 2^5 p_k$$

    Par conséquent $\ell(y_{k,j}) > \ell(x_k) \geqslant \ell(x_i) + 2^5 p_i$.
- Cas 2 : $u \in X$ et $v \in X \Rightarrow \exists i, j \in \{1, \cdots, k\}$ tel que $u = x_i$ et $v = x_j$:

  - x_i adjacent à $x_j \Rightarrow \ell(x_i) + p_i.2^5 \leqslant \ell(x_j) \leqslant \ell(x_i) + 2^5(p_i + p_{i+1})$

    Si x_i adjacent à x_j et $i < j$ alors $j = i + 1$. Or, par construction :

    $$\ell(x_i) + p_i.2^5 \leqslant \ell(x_{i+1}) \leqslant \ell(x_i) + (p_i + p_{i+1}).2^5$$

  - x_i n'est pas adjacent à $x_j \Rightarrow \ell(x_j) > \ell(x_i)) + (p_i + p_{i+1}).2^5$

    Si x_i n'est pas adjacent à x_j et $i < j$ alors $j \geqslant i + 2$. Dans ce cas, on a :

    $$\ell(x_j) \geqslant \ell(x_{i+2})$$

    Par conséquent on obtient que :

    $$\begin{aligned} \ell(x_{i+2}) &> \ell(x_{i+1}) + p_{i+1}.2^5 \\ \ell(x_{i+2}) &> \ell(x_i) + (p_i + p_{i+1}).2^5 \\ \ell(x_j) &> \ell(x_i) + (p_i + p_{i+1}).2^5 \end{aligned}$$

□

4.3.3 Longueur des étiquettes

Finalement, il ne reste plus qu'à majorer la taille des étiquettes. Contrairement aux autres schémas connus pour les arbres dans lesquelles cette étape est triviale, c'est ici l'étape la plus difficile. En effet, chaque sommet codant de l'information grâce à des bonds dans l'étiquetage, il faut s'assurer qu'au total la somme de ces bonds ne soit pas trop élevée.

Lemme 8. *Les étiquette ont au plus* $\lceil \log n \rceil + 8$ *bits.*

Démonstration. La longueur maximale des étiquettes est déterminée par l'étiquette du dernier sommet parcouru qui est la dernière feuille y_{k,d_k} du sommet x_k. Il suffit donc de majorer $\ell(y_{k,d_k})$ pour borner la taille des étiquettes. Pour majorer la valeur $\ell(y_{k,d_k})$, on a besoin de majorer $\sum_{i=1}^{i=k} p_i$. Soit P_n la propriété suivante :

$$(P_n): \quad \sum_{i=n}^{i=k} p_i \leqslant 2 \sum_{i=n}^{i=k} \lceil d_i \rceil_2 - p_n$$

On va prouver P_1 par récurrence afin d'obtenir un majorant de $\sum_{i=1}^{i=k} p_i$.
P_k est vrai car $\sum_{i=k}^{i=k} p_i = p_k = \lceil d_k \rceil_2 = 2 \sum_{i=k}^{i=k} \lceil d_i \rceil_2 - p_k$.
On suppose que P_n est vrai et on cherche à démonter P_{n-1} On a :

$$\begin{aligned} \sum_{i=n-1}^{i=k} p_i &= \sum_{i=n}^{i=k} p_i + p_{n-1} \\ \sum_{i=n-1}^{i=k} p_i &\leqslant 2 \sum_{i=n}^{i=k} \lceil d_i \rceil_2 - p_n + p_{n-1} \\ & \quad \text{puisque } P_n \text{ est vérifiée} \end{aligned}$$

Deux cas possibles :
– Cas 1 : $\lceil d_{n-1} \rceil_2 \leqslant \lfloor \sqrt{p_n} \rfloor_2 \Rightarrow p_{n-1} = \lceil d_{n-1} \rceil$

$$\begin{aligned} \sum_{i=n-1}^{i=k} p_i &\leqslant 2 \sum_{i=n}^{i=k} \lceil d_i \rceil_2 - p_n + \lceil d_{n-1} \rceil \\ \sum_{i=n-1}^{i=k} p_i &\leqslant 2 \sum_{i=n-1}^{i=k} \lceil d_i \rceil_2 - p_n - \lceil d_{n-1} \rceil \\ \sum_{i=n-1}^{i=k} p_i &\leqslant 2 \sum_{i=n-1}^{i=k} \lceil d_i \rceil_2 - p_{n-1} \end{aligned}$$

– Cas 2 : $\lceil d_{n-1} \rceil_2 > \lfloor \sqrt{p_n} \rfloor_2 \Rightarrow p_{n-1} = \lfloor \sqrt{p_n} \rfloor_2$

$$\begin{aligned}
\sum_{i=n-1}^{i=k} p_i &\leqslant 2\sum_{i=n}^{i=k} \lceil d_i \rceil_2 - p_n + \lfloor \sqrt{p_n} \rfloor_2 \\
\sum_{i=n-1}^{i=k} p_i &\leqslant 2\sum_{i=n}^{i=k} \lceil d_i \rceil_2 - p_n + \frac{p_n}{2} \\
\sum_{i=n-1}^{i=k} p_i &\leqslant 2\sum_{i=n}^{i=k} \lceil d_i \rceil_2 - \frac{p_n}{2} \\
\sum_{i=n-1}^{i=k} p_i &\leqslant 2\sum_{i=n-1}^{i=k} \lceil d_i \rceil_2 - p_{n-1}
\end{aligned}$$

P_n est donc vrai pour tout n. Pour $n = 1$, on obtient :

$$\sum_{i=1}^{i=k} p_i \leqslant 2\sum_{i=1}^{i=k} \lceil d_i \rceil_2$$

Par construction, on a :

$$\begin{aligned}
\ell(y_{k,d_k}) &\leqslant 2\sum_{i=1}^{i=k} p_i 2^5 \\
&\leqslant 2^6 \sum_{i=1}^{i=k} p_i \\
&\leqslant 2^7 \sum_{i=1}^{i=k} \lceil d_i \rceil_2 \\
&\quad \text{d'après } P_1 \\
&\leqslant 2^8 \sum_{i=1}^{i=k} d_i \\
&\quad \text{car } \forall n \in \mathbb{N}, \lceil n \rceil_2 \leqslant 2n \\
&\leqslant 2^8 n
\end{aligned}$$

Au final, les étiquettes ont donc au plus $\lceil \log(2^8 n) \rceil = \lceil \log n \rceil + 8$ bits. □

4.4 Schéma d'adjacence pour les arbres de degré interne borné

Dans cette partie, on utilise toujours la technique de parcours bondissant mais cette fois sur une famille plus complexe. Les principes généraux sont les mêmes mais le schéma est évidemment plus complexe. On choisit un parcours tel que les fils d'un même sommet y soient consécutifs tout comme le schéma pour les chenilles. Le parcours ainsi choisi est en quelque sorte un mélange entre un parcours en largeur et un parcours en profondeur. On a besoin de stocker plus d'informations afin de déterminer l'adjacence car les arbres considérés n'ont pas une structure aussi simple que celle des chenilles. La description de l'étiquetage et du test d'adjacence restent relativement simple mais la majoration de la longueur des étiquettes devient encore plus complexe.

On considère un arbre enraciné T. Pour sommet v de T, on note $F(v)$ l'ensemble des fils de T. On définit le *degré interne* $\hat{d}(v)$ d'un sommet v de T comme étant égal à $|F(v)|$. Le *degré interne maximum* de T est le maximum des degrés internes de ses sommets.

Théorème 15. *La famille des arbres admet un schéma d'adjacence avec des étiquettes de* $\log n + O(\log \hat{d})$ *bits où* n *est le nombre de sommets du graphe et* $\hat{d}$ *son degré interne maximum. De plus, l'adjacence peut être testée en temps constant et l'étiquetage peut s'effectuer en temps* $O(n)$*.*

4.4.1 Description du schéma

On considère un arbre enraciné T de n sommets ayant un degré interne $\hat{d}$. On note r_T la racine de T. Pour chaque sommet v, on note T_v le sous-arbre de T enraciné en v et le *poids* de v est défini comme étant égal à $|T_v|$. On ordonne les fils de chaque sommet de T de gauche à droite de telle manière à ce que l'on ait d'abord les feuilles puis ensuite les sommets internes ordonnés qui sont classés par poids décroissant de gauche à droite.

Premièrement, on doit sélectionner un parcours pour le graphe. Le parcours utilisé pour notre technique de parcours bondissant est défini comme suit : le parcours d'un sommet v consiste à visiter en premier les fils de v de gauche à droite puis à parcourir récursivement ces mêmes fils de droite à gauche. Le parcours de l'arbre T consiste tout simplement à visiter sa racine puis à la parcourir. La figure 30 donne un exemple d'un tel parcours.

L'avantage de ce parcours est que les fils d'un même sommet sont consécutifs dans le parcours. Ceci va permettre de réduire l'information à stocker pour le test d'adjacence.

Comme spécifié dans la description générale de notre technique de parcours bondissant, on associe à chaque sommet v de T un intervalle de valeurs $I(v)$, dont la borne gauche est ordonnée dans le même ordre que les sommets correspondants dans le parcours. L'étiquette binaire de v, noté $\ell(v)$, correspond en fait à un entier sélectionné dans l'intervalle $I(v)$. L'étiquette $\ell(x)$ code l'information nécessaire au test d'adjacence sous la forme d'un mot suffixe code(v). Grossièrement, cette information code(v) va permettre de calculer à partir de $\ell(v)$, un intervalle $B(v)$ contenant les étiquettes des fils de v et seulement les étiquettes

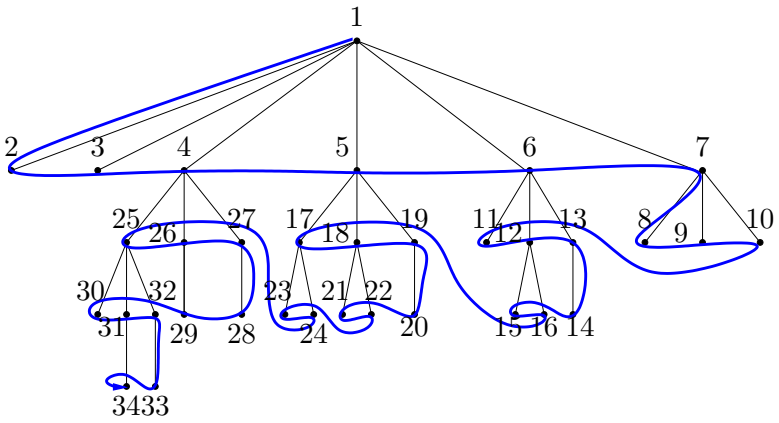


Figure 30 – Exemple de parcours d'un arbre.

de ces sommets, ce qui va permettre de tester l'adjacence entre les sommets.

Le calcul de l'étiquette $\ell(v)$ et de l'intervalle $I(v)$ nécessite le calcul préalable des entiers $p(v)$, $r(v)$, $c(v)$, $b(v)$, et du mot binaire $\text{code}(v)$ grâce aux relations suivantes. Pour un sommet v de T, on note v^+ le premier fils de v et v^- son frère étant situé juste après lui dans l'ordre des fils. Dans les équations ci-dessous, $p^+(v)$ est défini comme étant égal à $p(v^+)$ si v^+ existe et 0 sinon. On définit $p^-(v)$ de manière similaire. On fixe $\Psi(x) = \lceil \sqrt{x} \rceil$.

$$
\begin{aligned}
p(v) &= \Psi^2(p^-(v)) + p^+(v) + 2r(v) + b(v) \\
r(v) &= 2^{|\text{code}(v)|} \\
c(v) &= \sum_{u \in F(v)} r(u) \\
b(v) &= \begin{cases} 0 & \text{si } v \text{ est une feuille} \\ 2^{\lceil \log c(v) \rceil} - c(v) & \text{sinon} \end{cases} \\
\text{code}(v) &= \begin{cases} '0' & \text{si } v \text{ est une feuille} \\ \text{code}_1(\lceil \log c(v) \rceil) \circ \text{code}_2(\Psi(p^-(v))) \circ '1' & \text{sinon} \end{cases}
\end{aligned}
$$

Ces valeurs sont très importantes pour le bon fonctionnement du schéma. La valeur $p(v)$ correspond à la taille d'intervalle nécessaire au schéma afin d'attribuer des étiquettes à v et ses descendants ainsi que tous les autres sommets compris entre dans le parcours. Cette valeur est utilisée afin de déterminer à l'avance la place à réserver pour l'étiquetage de sommets. La preuve que cette valeur est suffisament grande fait l'objet du lemme 10. La valeur $r(v)$ correspond à la longueur de l'intervalle $I(v)$ contenant l'étiquette $\ell(v)$. Cette valeur doit être assez grande afin de fixer $\text{code}(v)$ en suffixe de $\ell(v)$ par le biais du fait 3. On fixe donc tout simplement $r(v)$ à $2^{|\text{code}(v)|}$. La valeur $c(v)$ correspond à la taille d'un intervalle contenant tous les étiquettes des fils de v. Elle est égale à la somme des tailles des intervalles des fils car ceux-ci sont contigus. La valeur $b(v)$ est le plus petit entier tel

que $b(v) + c(v)$ est une puissance entière de deux. Elle est utile afin de simplifier le codage de $c(v)$ car elle permet de coder $c(v)$ par son logarithme. Enfin, code(v) est le suffixe de l'étiquette $\ell(v)$. Il encode deux entiers $\lceil \log c(v) \rceil$ et $\Psi(p^-(v))$. Ces deux entier encode respectivement la taille et la distance par rapport à $\ell(v)$ d'un intervalle $B(v)$ contenant les étiquettes des fils de v et uniquement ceux-ci. Cette information permet de tester si un sommet est fils de v et permet donc bien l'adjacence.

Le calcul de ces valeurs pour tous les sommets de T peut se faire en temps $O(n)$ en parcourant les sommets de l'arbre T dans l'ordre inverse du parcours. L'information stockée par chaque sommet est donc son type (feuille ou non) et pour les sommets internes les valeurs $\Psi(p^-(v))$, $r(v)$ et $c(v)$ qui vont permettre le calcul de l'intervalle $B(v)$ contenant les étiquettes des fils de v. On augmente artificiellement la valeur $p^-(v)$ à $\Psi(p^-(v))^2$ afin de réduire l'information à stocker. En effet, la valeur $\Psi(p^-(v))^2$ peut être recalculée à partir de $\Psi(p^-(v))$ qui nécessite moitié moins de bits pour le codage.

Pour un intervalle d'entiers $I = [a, b[$, on fixe $I_[= a$ et $I_) = b$. La taille de l'intervalle I est $|I| = b - a$. Pour un mot binaire w et un entier z, on définit l'entier $\Phi(z, w)$ comme étant l'entier compris dans l'intervalle $[z, z + 2^{|w|}[$ calculé d'après le fait 3 tel que w soit suffixe de bin$(\Phi(z, w))$. L'intervalle $I(v)$ et l'étiquette $\ell(v)$ sont calculés d'après le système d'équations suivant :

$$\begin{aligned} I_[(r_T) &= 0 \\ \ell(v) &= \Phi(I_[(v), \text{code}(v)) \\ I_)(v) &= I_[(v) + r(v) \\ I_[(v^-) &= I_)(v) \\ I_[(v^+) &= \ell(v) + r(v) + \Psi^2(p^-(v)) + b(v) \end{aligned}$$

Les bornes de $I(v)$ et l'étiquette $\ell(v)$ peuvent être calculées en un seul parcours du graphe prenant un temps $O(n)$: on calcule la borne inférieure $I_[(v)$ en premier, ensuite l'étiquette $\ell(v)$ et finalement la borne $I_)(v)$. L'ordre de calcul de ces valeurs est le même que le parcours de T.

4.4.2 Description du test d'adjacence

Le test d'adjacence va se résumer à extraire l'information des étiquettes des sommets et à tester l'appartenance d'une étiquette d'un des sommets à un intervalle de valeurs. A chaque sommet interne v on va associer un intervalle $B(v)$. Le principe est que $B(v)$ contient toutes les étiquettes des fils de v et seulement les étiquettes de ces sommets. De plus, les bornes de $B(v)$ peuvent être calculées à partir des trois valeurs stockées dans code(v) : $\lceil \log c(u) \rceil$, $|\text{code}(u)|$ et $\Psi(p^-(u))$. Un sommet v va donc être le père d'un sommet u si et seulement si $\ell(u) \in B(v)$. Définir le test d'adjacence se résume donc à définir $B(v)$ pour tout sommet v de T.

Lemme 9. *Le test d'adjacence entre deux sommets u et v peut être effectué en temps constant à partir de leurs étiquettes respectives.*

Pour un sommet v de T, on définit la forêt F_v comme étant le sous-graphe de T induit par les sommets étant situés entre v et ses descendants, v compris, dans le parcours de l'arbre T. On définit pour tout sommet v de T l'intervalle $P(v)$ où $P_[(v) = I_[(v)$ et $|P(v)| = p(v)$. Afin de prouver le lemme 9, on a besoin de prouver le lemme suivant. Ce lemme va permettre de prouver que l'intervalle $B(v)$ pour tout sommet v de T ne contient bien que les étiquettes des fils de v.

Lemme 10. *Pour tout sommet v de T, on a :* $(Q_v) \quad \bigcup_{u \in V(F_v)} I(u) \subseteq P(v)$

Démonstration. Pour une feuille v, Q_v est vérifiée car $V(F_v) = \{v\}$ et $r(v) \leqslant p(v)$. On suppose par induction que Q_{v^-} et Q_{v^+} sont vérifiées. On peut remarquer que $V(F_v) = \{v\} \cup V(F_{v^-}) \cup V(F_{v^+})$ (voir Figure 31). Afin de montrer Q_v, on montre que $I(v) \cup P(v^-) \cup P(v^+) \subseteq P(v)$.

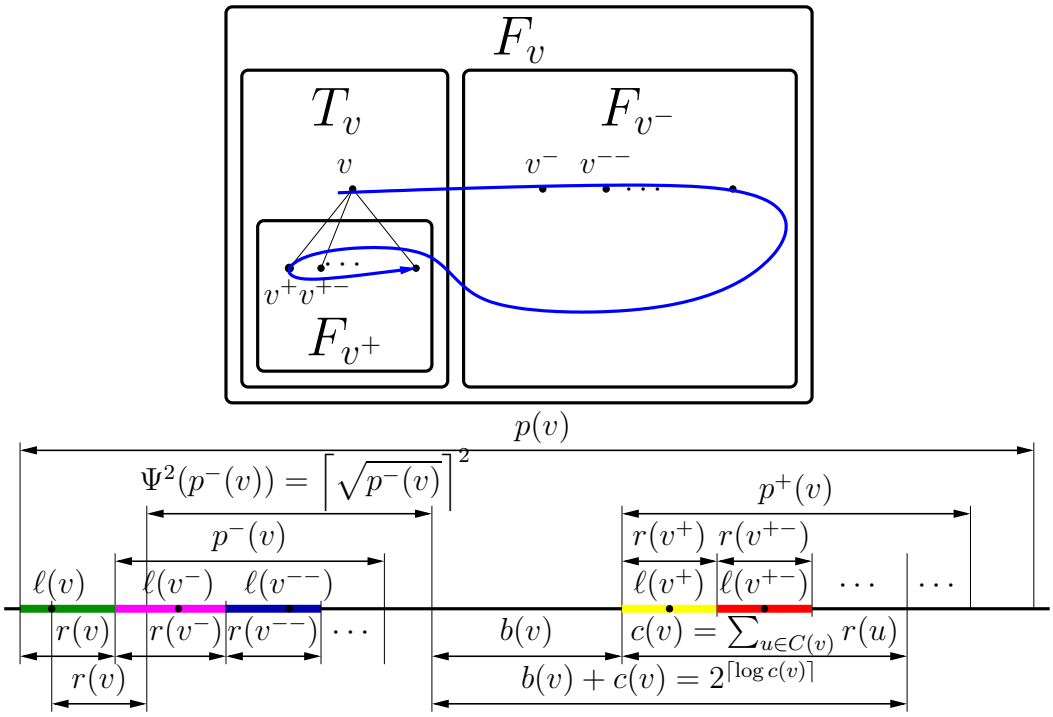


Figure 31 – Intervalles associés aux sommets.

Par définition, on a $P_)(v^+) = P_[(v^+) + p(v^+) = R_[(v^+) + p(v^+)$ et donc,

$$\begin{aligned} P_)(v^+) &= \ell(v) + r(v) + \Psi^2(p^-(v)) + b(v) + p(v^+) \\ &\leqslant R_[(v) + 2r(v) + \Psi^2(p^-(v)) + b(v) + p(v^+) \\ &\leqslant R_[(v) + p(v) \\ &\leqslant P_)(v). \end{aligned}$$

On en déduit que $I(v) \cup P(v^-) \cup P(v^+) \subseteq P(v)$ et donc Q_v est vérifiée puisque Q_{v^-} et Q_{v^+} sont vérifiées. Par induction, P_v est vérifiée pour tout sommet v de T. On en déduit que pour tout sommet v de T : □

Démonstration du lemme 9 On suppose sans perte de généralité que $\ell(u) < \ell(v)$. Dans ce cas, u et v sont adjacents si et seulement si u est le père de v car chaque sommet v de T est précédé par son père dans le parcours et donc dans l'ordre des étiquettes. Le test d'adjacence est calculé comme suit :

- Si u est une feuille, alors u et v ne sont pas adjacents. Déterminer si u est une feuille ou non peut être fait en examinant le dernier bit de $\ell(u)$. En effet, ce bit, qui est le dernier bit de code(v), est égal à $'0'$ si u est une feuille et $'1'$ sinon.
- Si u n'est pas une feuille, alors on extrait $\lceil \log c(u) \rceil$ et $\Psi(p^-(v))$ de code(u) suffixe de $\ell(u)$ en temps constant. On utilise alors la condition d'adjacence suivante : le sommet u est adjacent à v si et seulement si

$$\ell(u) - \ell(v) - 2^{|\text{code}(u)|} - \Psi^2(p^-(v)) \ \in \ [0, 2^{\lceil \log c(v) \rceil}[\ .$$

Il suffit donc de prouver la validité de cette condition d'adjacence en montrant que toutes les étiquettes des fils d'un sommet u, et seulement ces étiquettes, sont comprises dans l'intervalle $B(u) = [\ell(u)+2^{|\text{code}(u)|}+\Psi^2(p^-(u)), \ell(u)+2^{|\text{code}(u)|}+\Psi^2(p^-(u))+2^{\lceil \log c(u) \rceil}[$. Tout d'abord, on montre que l'intervalle $B(u)$ contient tous les intervalles associés aux fils de u. Pour chaque v de T, on a $I_[(v^-) = I_)(v)$. Donc, les intervalles des fils de u sont contigus. Soit $[a, b) = \bigcup_{v \in F(u)} I(v)$. Il suffit de montrer que $[a, b) \subseteq B(v)$. On a :

$$\begin{aligned} a &= I_[(u^-) \\ &= \ell(v) + 2^{|\text{code}(u)|} + \Psi(p^-(u))^2 + b(u) \\ &\geqslant B_[(u). \end{aligned}$$

et aussi

$$\begin{aligned} b &= I_[(u^-) + c(u) \\ &= \ell(u) + 2^{|\text{code}(u)|} + \Psi(p^-(u))^2 + b(u) + c(u) \\ &= B_)(u). \end{aligned}$$

Finalement, on a bien $[a, b[\subseteq B(u)$.

Il ne reste plus qu'à montrer que l'intervalle $B(u)$ ne contient pas d'étiquettes associées à des sommets de l'arbre qui ne sont pas des fils de u. On déduit du lemme 10 que pour tout sommet v de T, on a :

$$\begin{aligned} P_)(v^-) &= I_)(v) + p(v^-) \\ &\leqslant I_[(v) + 2r(v) + \Psi^2(p^-(v)) \\ &\leqslant B_[(v). \end{aligned}$$

Donc, il n'y a pas d'étiquettes correspondant à des sommets de F_{u^-} comprises dans l'intervalle $B(u)$. De plus, par construction du schéma, on a $b = B_)(u)$. On en déduit qu'il n'y a pas d'étiquettes comprises dans $B(u)$ correspondant à des sommets de $V(T)$ étant après les fils de u dans le parcours. Les étiquettes des fils de u sont donc bien les seules étiquettes comprises dans $B(v)$. □

4.4.3 Longueur des étiquettes

Il ne reste plus qu'à majorer la longueur des étiquettes du schéma. Pour cela on va majorer l'intervalle de valeurs utilisées par le schéma pour étiqueter les sommets. Cet intervalle est compris dans l'intervalle $P(r_T)$ (avec r_T racine de t) car $F_{r_T} = T$. On va donc chercher à borner $p(r_T) = |P(r_T)|$. Pour cela, on va partir de la récursion sur $p(v)$ définie dans la partie 4.4.1. On va tout d'abord simplifier cette récursion grâce au lemme 12. Ensuite, on va majorer $p(v)$ grâce à une étude par cas. L'idée générale est que la technique de majoration va différer suivant le rapport entre les valeurs $\log c(v)$ et $\Psi(v)$. En effet, la principale difficulté va être de borner la valeur $r(v)$ et celle-ci dépend des deux valeurs $\log c(v)$ et $\Psi(v)$ codées par code(v). On va donc appliquer deux méthodes de majorations différentes suivant celle des deux valeurs qui la plus grande.

Lemme 11. *Les étiquettes ont au plus* $\log n + O(\lg \hat{d})$ *bits.*

La démonstration du lemme 11 nécessite l'utilisation du lemme 12.

Lemme 12. *Soit* $c \in \mathbb{R}^+$ *une constante absolue. Soit* $r(v)$, $f(v)$ *et* $g(v)$ *trois fonctions entières définies sur les sommets de* T *et* $h(x)$ *une fonction de* $\mathbb{R}$ *dans* $\mathbb{R}$. *Si* h *est strictement croissante et si* f *et* g *vérifient le système suivant d'équations*[1] *:*

$$\begin{aligned} f(v) &\leqslant f^-(v) + h(f^-(v)) + f^+(v) + c \cdot r(v) + \sum_{u \in F(v)} r(u) \\ g(v) &= g^-(v) + h(g^-(v)) + g^+(v) + (c+1) \cdot r(v) \end{aligned}$$

alors $\forall v \in T, g(v) \geqslant f(v)$.

Démonstration. Pour tout sommet v de T, on note $S(v)$ l'ensemble des sommets frères de v. Le sous-ensemble des frères de v situé a sa droite (respectivement sa gauche) dans l'ordre des fils est noté $S_>(v)$ (respectivement $S_<(v)$). On définit le sous-ensemble $S_\geqslant(v)$ comme étant égal à $S_>(v) \cup \{v\}$.

On cherche à montrer la propriété suivante pour tous les sommets $v \in V(T)$:

$$R_v : \quad g(v) \geqslant f(v) + \sum_{u \in S_\geqslant(v)} r(u).$$

1. On fixe $f^-(v) = 0$ si v^- n'existe pas et $f^-(v) = f(v^-)$ sinon. On définit de manière similaire $f^+(v)$, $g^-(v)$ et $g^+(v)$.

Soit un sommet v de T. Par soucis de simplification, on va considérer que la propriété est vérifiée pour les sommets v^- et v^+ si ceux-ci n'existent pas dans l'arbre, c'est-à-dire, dans le cas où v n'as pas de frère à droite ou bien est une feuille. En effet, dans le cas où v n'as pas de frère à droite, on a $f(v^-) = f^-(v) = g(v^-) = g^-(v) = 0$ et donc R_{v^-} est vérifiée. De même, dans le cas ou v est une feuille pour v^+, R_{v^+} est vérifiée. Ces sommets virtuels forment la base de la récursion.

On suppose par induction que R_{v^-} et R_{v^+} sont vérifiées. Il reste à prouver que R_v est vérifiée.

On a :

$$\begin{aligned} g(v) &= g^-(v) + h(g^-(v)) + g^+(v) + (c+1)r(v) \\ g(v) &\geqslant f^-(v) + \sum_{u \in S_{\geqslant}(v^-)} r(u) + h(f^-(v)) + f^+(v) + \sum_{u \in S_{\geqslant}(v^+)} r(u) + (c+1)r(v) \\ &\quad \text{car } R_{v^-} \text{ et } R_{v^+} \text{ sont vérifiées et } h(x) \text{ est strictement croissante} \end{aligned}$$

Il est clair que $S_{>}(v) = S_{\geqslant}(v^-)$ et donc $S_{\geqslant}(v) = S_{\geqslant}(v) \cup \{v\}$. On en déduit que :

$$\begin{aligned} g(v) &\geqslant f^-(v) + h(f^-(v)) + f^+(v) + c \cdot r(v) + \sum_{u \in S_{\geqslant}(v)} r(u) + \sum_{u \in S_{\geqslant}(v^+)} r(u) \\ g(v) &\geqslant f(v) + \sum_{u \in S_{\geqslant}(v)} r(u) \end{aligned}$$

Donc, R_v est vérifiée. Par induction, R_v est vérifiée pour tout sommet v de T. □

Démonstration du lemme 11. Afin de majorer la longueur des étiquettes, on montre que $\forall v \in V(T), \quad p(v) = O(|T_v|\hat{d}^{O(1)})$. Par définition de $p(v)$, $\forall v \in V(T)$, on a :

$$p(v) = \Psi^2(p^-(v)) + p^+(v) + 2r(v) + b(v)$$

Soit $f(v)$ la fonction définie par :

$$f(v) = f^-(v) + 2\sqrt{f^-(v)} + f^+(v) + 2r(v) + b(v)$$

Il est facile de vérifier que $\forall x \in \mathbb{R}^+, \Psi^2(x) \leqslant x + 2\sqrt{x}$. Par conséquent, $f(v) \geqslant p(v)$ pour tout $v \in V(T)$. Soit $g(v)$ la fonction définie par :

$$g(v) = g^-(v) + 2\sqrt{g^-(v)} + g^+(v) + 3r(v)$$

On rappelle que $b(v) \leqslant c(v) = \sum_{u \in F(v)} r(u)$ et que donc $f(v) \leqslant f^-(v) + h(f^-(v)) + f^+(v) + 2r(v) + \sum_{u \in F(v)} r(u)$. D'après le lemme 12 appliqué sur $f(v)$ et $g(v)$ avec $c = 2$ et $h(x) = 2\sqrt{x}$, on déduit que $\forall v \in T, g(v) \geqslant f(v)$.

Il suffit donc de majorer la fonction $g(v)$. Pour cela, on définit deux fonctions $\alpha = \alpha(\hat{d})$ et $\beta = \beta(\hat{d})$ grâce au système d'inégalités suivant :

$$\begin{cases} 2(d\alpha)^{5/8} \leqslant \left(\frac{1-(2\hat{d}+1)^{7/8}}{(2\hat{d})^{7/8}} + 1\right)\beta \\ \frac{1}{2}(\alpha - \beta) \geqslant d\alpha^{5/8}\log^8 x_0 \\ 8 + \beta \leqslant \alpha \end{cases} \qquad (1)$$

où d et x_0 sont des constantes absolues.

Soit $\tau(\hat{d}) = \frac{2c^{7/8}(2\hat{d})^{7/8}}{(2\hat{d})^{7/8}+1-(2\hat{d}+1)^{7/8}}$. En fixant $\beta = \tau(\hat{d})\alpha^{5/8}$, on s'assure que la première inégalité est vérifiée. Clairement, les valeurs de β respectant ce système d'inégalités sont supérieures à 8. Par conséquent, le système d'inégalités suivant implique le précédent.

$$\begin{aligned} (1) \quad &\Leftarrow \quad \begin{cases} \beta = \tau(\hat{d})\alpha^{5/8} \\ \frac{\alpha - \tau(\hat{d})\alpha^{5/8}}{2} \geqslant c\alpha^{5/8}\log^8 x_0 \\ 2\tau(\hat{d})\alpha^{5/8} \leqslant \alpha \end{cases} \\ &\Leftarrow \quad \begin{cases} \beta = \tau(\hat{d})\alpha^{5/8} \\ \alpha \geqslant 2\tau(\hat{d})\alpha^{5/8} + 2c\alpha^{5/8}\log^8 x_0 \end{cases} \\ &\Leftarrow \quad \begin{cases} \beta = \tau(\hat{d})\alpha^{5/8} \\ \alpha^{3/8} \geqslant 2\tau(\hat{d}) + 2c\log^8 x_0 \end{cases} \end{aligned}$$

On en déduit que $\alpha^{\frac{3}{8}} = O(\tau(\hat{d}))$ et donc $\alpha = O(\tau(\hat{d})^{\frac{8}{3}}) = O(\hat{d}^{\frac{7}{3}})$.

Le reste de la preuve va consister à prouver que les étiquettes ont au plus $\log n + O(\log \alpha)$ bits. Dans ce but, on va montrer la propriété Q_v pour tout sommet v de T.

$$Q_v : g(v) \ \leqslant \ \alpha|F_v| - \beta|F_v|^{7/8} - \sum_{u \in S_{\geqslant}(v)} r(u).$$

Soit un sommet v de T. On suppose par induction que Q_{v^-} et Q_{v^+} sont vérifiées. Il reste à prouver que Q_v est vérifiée. On considère deux cas.

Cas 1 : v est une feuille.

Dans ce cas, on a code(v) = '0', $r(v) = 2$ et $g^+(v) = 0$.

On distingue deux sous-cas possibles.

Cas 1.1 : v n'a pas de frère à droite.

Dans ce cas, on a $S_{\geqslant}(v) = \{v\}$ et Q_v est équivalent à $4 \ \leqslant \ \alpha - \beta - 2$. La propriété Q_v est donc vérifiée car $\beta + 8 \leqslant \alpha$ d'après le système d'inégalités 1.

Cas 1.2 : v as un frère à droite.

Par hypothèse d'induction, Q_{v^-} est vérifiée. On a donc :

$$\begin{aligned} g^-(v) &\leqslant \alpha|F_{v^-}| - \beta|F_{v^-}|^{7/8} - \sum_{u\in S_{\geqslant}(v^-)} r(u) \\ &\leqslant \alpha(|F_v|-1) - \beta(|F_v|-1)^{7/8} - \sum_{u\in S_{\geqslant}(v^-)} r(u) \\ &\leqslant \alpha|F_v| - \beta|F_v|^{7/8} - \sum_{u\in S_{\geqslant}(v)} r(u) + r(v) - \alpha + \beta \end{aligned}$$

et donc on obtient :

$$\begin{aligned} g(v) &= g^-(v) + 3r(v) \\ &\leqslant \alpha|F_v| - \beta|F_v|^{7/8} - \sum_{u\in S_{\geqslant}(v)} r(u) + (4r(v) - \alpha + \beta) \\ &\leqslant \alpha|F_v| - \beta|F_v|^{7/8} - \sum_{u\in S_{\geqslant}(v)} r(u) \end{aligned}$$

puisque $4r(v) - \alpha + \beta \leqslant 0 \Leftrightarrow 8 + \beta \leqslant \alpha$ d'après le système d'inégalités 1

Par conséquent, la propriété Q_v est vérifiée.

Cas 2 : v est un sommet interne.

On a :

$$\log r(v) = |\mathrm{code}(v)| \leqslant |\mathrm{code}_1(\lceil \log c(v)\rceil)| + \left|\mathrm{code}_2\left(\Psi(g^-(v))\right)\right| + 1$$

On rappelle que $\forall x \in \mathbb{N}$, $2^{\lg x} \leqslant x+1$. Le premier terme de $\log r(v)$ peut donc être majoré par :

$$\begin{aligned} |\mathrm{code}_1(\lceil \log c(v)\rceil)| &= 2\lfloor \lg \lceil \log c(v)\rceil \rfloor + 2, \text{ et donc} \\ |\mathrm{code}_1(\lceil \log c(v)\rceil)| &\leqslant 2\lfloor \lg \log c(v)\rfloor + 4 \\ 2^{|\mathrm{code}_1(\lceil \log c(v)\rceil)|} &\leqslant 8\log^2 c(v) + 8 \end{aligned}$$

Cas 2.1 : v a un frère à droite.

Dans ce cas, on a $g^-(v) \neq 0$. Soit $u = \sqrt{g^-(v)}$. Le second terme de $\log r(v)$ peut être majoré par :

$$\begin{aligned} |\mathrm{code}_2(\lceil u\rceil)| &= \lfloor \lg \lceil u\rceil \rfloor + 2\lfloor \lg \lfloor \lg \lceil u\rceil \rfloor \rfloor + 3, \text{ et donc} \\ 2^{|\mathrm{code}_2(\lceil u\rceil)|} &\leqslant 8\cdot(\lceil u\rceil + 1)\cdot(\lfloor \lg \lceil u\rceil \rfloor^2 + 1) \leqslant 16u(\log^2 u + 2)\ . \end{aligned}$$

On en déduit que :

$$\begin{aligned} r(v) &\leqslant (\log^2 \lceil g^-(v)+2)\rceil \cdot 8(\log^2 c(v)+2)\cdot 16\left\lceil \sqrt{g^-(v)}\right\rceil \\ r(v) &\leqslant c_1 \cdot \log^2 g^-(v)\cdot \log^2 c(v)\cdot \sqrt{g^-(v)} \end{aligned}$$

où c_1 est une constante car $\log^2 c(v) \geqslant 1, \log^2 g^-(v) \geqslant 1$ et $\sqrt{g^-(v)} \geqslant 1$.

On sait que $\log^2 x = O(x^{1/8})$. il existe $a, n_0 \in \mathbb{R}^+$ tels que :

$$\forall x \in I, \quad x \geqslant n_0, \quad \log^2 x \leqslant a \cdot x^{1/8} \leqslant a \cdot x^{1/8} + \log^2 n_0$$

On a

$$\forall x \in I, \quad x \leqslant n_0, \quad \log^2 x \leqslant \log^2 n_0 \leqslant a \cdot x^{1/8} + \log^2 n_0$$

parce que $\log^2 x$ est strictement croissant sur $\mathbb{R}^+$ et donc

$$\forall x \in I, \quad x \leqslant n_0, \quad \log^2 x \leqslant \log^2 n_0 \leqslant a \cdot x^{1/8} + \log^2 n_0$$

On en déduit qu'il existe deux constantes a et $b = \log^2 n_0$ telles que :

$$\begin{aligned}
r(v) &\leqslant c_1\sqrt{g^-(v)} \cdot (a \cdot g^-(v)^{\frac{1}{8}} + b) \cdot \log^2 c(v) \\
r(v) &\leqslant c_1\sqrt{g^-(v)} \cdot a\left(g^-(v)^{\frac{1}{8}} + \frac{b}{a}\right) \cdot \log^2 c(v) \\
r(v) &\leqslant c_1 a\left(\frac{b}{a} + 1\right) g^-(v)^{5/8} \cdot \log^2 c(v) \\
&\qquad \text{car } g^-(v)^{5/8} > 1 \quad \text{et donc } g^-(v)^{5/8} + \frac{b}{a} < \left(\frac{b}{a} + 1\right) g^-(v)^{5/8}
\end{aligned}$$

On obtient que :

$$\begin{aligned}
r(v) &\leqslant c_2 g^-(v)^{5/8} \log^2 c(v) \\
&\qquad \text{où } c_2 \text{ est une constante}
\end{aligned}$$

On distingue deux sous-cas :

Case 2.1.1 : $\log^2 c(v) \geqslant |F_{v^-}|^{\frac{1}{4}}$.
Dans ce cas, on a :

$$\begin{aligned}
r(v) &\leqslant c_2 g^-(v)^{5/8} \log^2 c(v) \\
r(v) &\leqslant c_2 \alpha^{5/8} |F_{v^-}|^{5/8} \log^2 c(v) \\
r(v) &\leqslant d\alpha^{5/8} \log^8 c(v) \\
&\qquad \text{où } d \text{ est la constante du système d'inégalité 1}
\end{aligned}$$

Du fait que Q_{v^-} et Q_{v^-} sont vérifiées, on déduit que :

$$\begin{aligned}
g(v) &\leqslant \alpha|F_v| - \beta|F_{v^-}|^{7/8} - \beta|F_{v^+}|^{7/8} - \sum_{u \in S_{\geqslant}(v^-)} r(u) - \sum_{u \in S_{\geqslant}(v^+)} r(u) + d\alpha^{5/8} \log^8 c(v) - \alpha \\
g(v) &\leqslant \alpha|F_v| - \beta(|F_{v^-}|^{7/8} + |F_{v^+}|^{7/8} + 1) - \sum_{u \in S_{\geqslant}(v^-)} r(u) - c(v) + d\alpha^{5/8} \log^8 c(v) - \alpha + \beta \\
&\qquad \text{parce que } c(v) = \sum_{u \in F(v)} r(u) = \sum_{u \in S_{\geqslant}(v^+)} r(u) \\
g(v) &\leqslant \alpha|F_v| - \beta|F_v|^{7/8} - c(v) - \sum_{u \in S_{\geqslant}(v^+)} r(u) + d\alpha^{5/8} \log^8 c(v) - \alpha + \beta
\end{aligned}$$

En effet, on a $|F_{v^-}| + |F_{v^+}| + 1 = |F_v|$ et donc $(|F_{v^-}| + |F_{v^+}| + 1)^{7/8} = |F_v|^{7/8}$ et $|F_{v^-}|^{7/8} + |F_{v^+}|^{7/8} + 1 \geqslant |F_v|^{7/8}$.

On considère la fonction $h(x) = \frac{x}{2} - d\alpha^{5/8} \log^8 x$. Clairement, on a que :

$$\lim_{x \to +\infty} h(x) = +\infty$$

et donc il existe x_0 dépendant de α et donc de $\hat{d}$ tel que $\forall x \geqslant x_0, \frac{x}{2} - d\alpha^{5/8} \log^8 x \geqslant 0$. De plus, la fonction $\log^8 x$ est strictement croissante sur $[1, +\infty)$. Par conséquent, $\forall x \geqslant 1, \frac{x}{2} - d\alpha(\log^8 x - \log^8 x_0) \geqslant 0$.

D'après le système d'inégalité 1, on a $\frac{1}{2}(\alpha - \beta) \geqslant d\alpha^{5/8} \log^8 x_0$ et donc $\forall x \geqslant 1, \frac{x}{2} \geqslant \alpha^{5/8} d \log^8 x - \frac{1}{2}(\alpha - \beta)$. En prenant $x = c(v)$, il en suit que :

$$\begin{aligned}
g(v) &\leqslant \alpha|F_v| - \beta|F_v|^{7/8} - c(v) - \sum_{u \in S_{\geqslant}(v^+)} r(u) + \frac{c(v)}{2} - \frac{1}{2}(\alpha - \beta) \\
g(v) &\leqslant \alpha|F_v| - \beta|F_v|^{7/8} - \frac{c(v)}{2} - \frac{1}{2}(\alpha - \beta) - \sum_{u \in S_{\geqslant}(v^+)} r(u) \\
g(v) &\leqslant \alpha|F_v| - \beta|F_v|^{7/8} - d\alpha^{5/8} \log^8 c(v) - \sum_{u \in S_{\geqslant}(v^+)} r(u) \\
g(v) &\leqslant \alpha|F_v| - \beta|F_v|^{7/8} - r(v) - \sum_{u \in S_{\geqslant}(v^+)} r(u) \\
g(v) &\leqslant \alpha|F_v| - \beta|F_v|^{7/8} - \sum_{u \in S_{\geqslant}(v)} r(u)
\end{aligned}$$

Cas 2.1.2 : $\log^2 c(v) < |F_{v^-}|^{\frac{1}{4}}$.

Dans ce cas, on a :

$$\begin{aligned}
r(v) &\leqslant d g^-(v)^{5/8} \log^2 c(v) \\
r(v) &\leqslant d\alpha^{5/8} |F_{v^-}|^{7/8} \qquad (2)
\end{aligned}$$

Du fait que Q_{v^-} et Q_{v^-} sont vérifiées, on obtient que :

$$\begin{aligned}
g(v) &\leqslant \alpha|F_v| - \beta|F_{v^-}|^{7/8} - \beta|F_{v^+}|^{7/8} - c(v) - \sum_{u \in S_{\geqslant}(v^-)} r(u) + d\alpha^{5/8}|F_{v^-}|^{7/8} - \alpha + \beta \\
&\leqslant \alpha|F_v| - (\beta - d\alpha^{5/8})|F_{v^-}|^{7/8} - \beta|F_{v^+}|^{7/8} - \sum_{u \in S_{\geqslant}(v^-)} r(u)
\end{aligned}$$

Il reste à montrer l'inégalité suivante :

$$\begin{aligned} -(\beta-(d\alpha)^{5/8})|F_{v^-}|^{7/8}-\beta|F_{v^+}|^{7/8} &\leqslant -\beta|F_v|^{7/8}-r(v)\\ \Leftrightarrow -(\beta-(d\alpha)^{5/8})|F_{v^-}|^{7/8}-\beta|F_{v^+}|^{7/8} &\leqslant -\beta|F_v|^{7/8}-d\alpha^{5/8}|F_{v^-}|^{7/8}\\ \text{à cause de l'inégalité 2}&\\ \Leftrightarrow (\beta-2(d\alpha)^{5/8})|F_{v^-}|^{7/8}+\beta|F_{v^+}|^{7/8} &\geqslant \beta|F_v|^{7/8} \end{aligned}$$

Les fils internes de v sont ordonnés par poids décroissants. Donc, $\forall u \in S_{\geqslant}(v), \quad |T_u| \leqslant |T_v|$. On obtient que :

$$\begin{aligned} \sum_{u\in S_{\geqslant}(v^-)} |T_u| &\leqslant (\hat{d}-1)|T_v|\\ |F_{v^-}| &\leqslant (\hat{d}-1)(|F_{v^+}|+1)\\ |F_{v^-}| &\leqslant 2\hat{d}|F_{v^+}| \end{aligned}$$

Par conséquent, $|F_{v^-}| \leqslant \frac{2\hat{d}|F_v|}{2\hat{d}+1}$ et $|F_{v^+}| \geqslant \frac{|F_v|}{2\hat{d}+1}$ car $|F_{v^-}|+|F_{v^+}| \leqslant |F_v|$. Consécutivement, il existe $x \in \left(0, \frac{2\hat{d}}{2\hat{d}+1}\right]$ tel que $|F_{v^-}| = x|F_v|$ et $|F_{v^+}| \leqslant (1-x)|F_v|$. On en déduit que :

$$\begin{aligned} (\beta-2(d\alpha)^{5/8})|F_v|^{7/8}x^{7/8}+\beta|F_v|^{5/8}(1-x)^{7/8} &\geqslant \beta|F_v|^{7/8}\\ (\beta-2(d\alpha)^{5/8})x^{7/8}+\beta(1-x)^{7/8} &\geqslant \beta \end{aligned}$$

D'après le système d'inégalités 1, on a $2(d\alpha)^{5/8} \leqslant \left(\frac{1-(2\hat{d}+1)^{7/8}}{(2\hat{d})^{7/8}}+1\right)\beta$, et donc :

$$\begin{aligned} \left(\frac{(2\hat{d}+1)^{7/8}-1}{(2\hat{d})^{7/8}}\right)\beta x^{7/8}+\beta(1-x)^{7/8} &\geqslant \beta\\ \left(\frac{(2\hat{d}+1)^{7/8}-1}{(2\hat{d})^{7/8}}\right)x^{7/8}+(1-x)^{7/8} &\geqslant 1 \qquad (3) \end{aligned}$$

On considère la fonction $g(x) = \left(\frac{(2\hat{d}+1)^{7/8}-1}{(2\hat{d})^{7/8}}\right)x^{7/8}+(1-x)^{7/8}$. On calcule sa dérivée en fonction de x :

$$\frac{\partial g(x)}{\partial x} = \frac{7}{8}\left(\frac{(2\hat{d}+1)^{7/8}-1}{(2\hat{d})^{7/8}x^{\frac{1}{8}}}-\frac{1}{(1-x)^{\frac{1}{8}}}\right)$$

$\frac{\partial g(x)}{\partial x}$ est positive sur $(0,\gamma]$ et négative sur $\left[\gamma, \frac{2\hat{d}}{2\hat{d}+1}\right]$ avec γ tel que :

$$\frac{7}{8}\left(\frac{(2\hat{d}+1)^{7/8}-1}{(2\hat{d})^{7/8}\gamma^{\frac{1}{8}}}-\frac{1}{(1-\gamma)^{\frac{1}{8}}}\right)=0.$$

Par conséquent, la fonction $g(x)$ est croissante sur $(0, \gamma]$ et décroissante sur $\left[\gamma, \frac{2\hat{d}}{2\hat{d}+1}\right]$. Clairement, $\forall x \in \left[0, \frac{2\hat{d}}{2\hat{d}+1}\right], g(x) \geqslant \min\left\{g(0), g\left(\frac{2\hat{d}}{2\hat{d}+1}\right)\right\}$. On a $g(0) = 1$ et

$$\begin{aligned} g\left(\frac{2\hat{d}}{2\hat{d}+1}\right) &= \left(\frac{(2\hat{d}+1)^{7/8}-1}{(2\hat{d})^{7/8}}\right)\frac{(2\hat{d})^{7/8}}{(2\hat{d}+1)^{7/8}} + \frac{1}{(2\hat{d}+1)^{7/8}} \\ &= 1 \end{aligned}$$

Finalement, le système d'inégalité 3 est prouvé et donc Q_v est vérifiée.

Cas 2.2 : v n'a pas de frère à droite.

Dans ce cas, on a :

$$\begin{aligned} g(v) &\leqslant g^-(v) + g^+(v) + d\log^2 c(v) \\ &\quad \text{où } d \text{ est la constante du système d'inégalité 1.} \end{aligned}$$

Par la même preuve que le cas 2.1.2, on obtient que Q_v est vérifiée.

Par induction, Q_v est vérifié pour chaque v de T et $p(v) \leqslant \alpha|T_v|$ et donc $p(v) = O(|T_v|\hat{d}(v)^{O(1)})$ parce que $\alpha = O(\hat{d^{\frac{7}{3}}})$. On rappelle que $p(v)$ est la taille de l'intervalle $P(v)$ contenant toutes les étiquettes des sommets de F_v. Pour la racine r_T de T, $F_{r_T} = T$ et l'intervalle $P(r_T)$ contient toutes les étiquettes des sommets de $V(T)$ et est de taille $O(|T|\hat{d^{\frac{7}{3}}})$. Au final, les étiquettes ont au plus $\log n + O(\log \hat{d})$ bits. □

4.5 Conclusion

Les schémas d'adjacence pour les graphes planaires sont basés sur des schémas d'adjacence pour les arbres et une partition d'arêtes des graphes planaires en trois arbres. En effet, le meilleur schéma connu avait des étiquettes de $3\log n + O(\log^* n)$ bits et utilise le schéma pour les arbres en $\log n + O(\log^* n)$ bits [9]. Il est possible d'améliorer légèrement ce résultat en utilisant le schéma pour les arbres de degré interne borné. On utilise une partition des arêtes d'un graphe planaire en trois forêts dont une est de degré au plus 4 [55]. On peut coder la forêt de degré borné en utilisant des étiquettes de $\log n + O(1)$ bits d'après le théorème 15. Ces étiquettes sont par définition deux à deux distinctes et peuvent donc servir d'identifiants. Pour coder les deux autres forêts, il suffit donc que chaque sommet code l'identifiant de son père dans chacune de ces deux forêts en plus de son identifiant personnel. Il est clair que les étiquettes ainsi obtenues permettent le test de l'adjacence dans les trois forêts et donc le graphe planaire. Finalement, ces étiquettes ont $3\log n + O(1)$ bits puisqu'elles sont issues de la concaténation de trois identifiants de $\log n + O(1)$ bits.

Il semble difficile d'améliorer ce résultat en utilisant une décomposition en forêts. En effet, les graphes d'arboricité α requièrent des étiquettes d'au moins $\alpha \log n - O(\alpha^2)$ bits d'après le minorant de [9]. Pour le cas des graphes planaires qui sont d'arboricité 3, cela

nous donne des étiquettes d'au moins $3 \log n - O(1)$ bits. Pour espérer obtenir un schéma plus compact, il faut donc utiliser une partition qui contienne moins de parts. C'est dans cette optique que l'on va utiliser une partition en sous-graphes de largeur arborescente bornée dans le chapitre suivant associé à un schéma en $(1 + o(1)) \log n$ bits pour ses sous-graphes.

Chapitre 5

Schéma d'adjacence pour les graphes planaires

5.1 Introduction

Notre schéma d'adjacence pour les graphes planaires est déduit d'un schéma pour les graphes de largeur arborescente k avec des étiquettes de $(1 + o(1)) \log n$ bits combiné avec une partition d'arêtes appropriée. On utilise une partition d'arêtes en deux graphes de largeur arborescente bornée plutôt qu'une partition en trois forêts (graphes de largeur arborescente 1). En effet, on peut espérer obtenir avec la première partition des étiquettes de $(2 + o(1)) \log n$ bits au lieu de $3 \log n$ bits. Tout comme la partition en trois forêts, cette partition en deux graphes de largeur arborescente bornée peut se faire en temps $O(n)$ de la manière suivante : 1) on partitionne les sommets en couches L_i à distance i d'un sommet arbitraire ; 2) le graphe H_i, $i > 0$ induit par les arêtes $\{uv \mid u \in L_i, v \in L_i \cup L_{i-1}\}$ a une largeur arborescente d'au plus 3 [40] ; 3) les deux graphes issus de l'union des H_i pour respectivement i pair et i impair sont de largeur arborescente bornée eux aussi car ils sont issus de l'union disjointes de graphes H_i.

En utilisant la décomposition décrite ci-dessus, il est clair qu'un schéma avec des étiquettes de λ bits pour les graphes de largeur arborescente k implique un schéma avec des étiquettes de 2λ bits pour les graphes planaires. En effet, il suffit d'assigner à chaque sommet une étiquette pour chaque sous-graphe et de tester l'adjacence dans les deux sous-graphes. En fait, il a été récemment prouvé que les arêtes des graphes planaires peuvent être partitionnées en deux graphes planaires extérieurs [54] (qui sont de largeur arborescente au plus deux). Un résultat similaire a été prouvé pour les graphes de genre d'Euler g ainsi que les graphes excluant un mineur fixé [35, 34] : ces graphes peuvent être partitionnés en deux sous-graphes dont la largeur arborescente dépend uniquement du genre ou du mineur exclu.

Notre schéma d'adjacence en $\log n + O(\log \log n)$ pour les graphes de largeur arborescente bornée implique donc un schéma en $2 \log n + O(\log \log n)$ pour les planaires mais aussi

les graphes excluant un mineur fixé. Avant de commencer la description de notre schéma, il convient de préciser qu'un schéma compact pour les arbres ne se traduit pas directement en un schéma compact pour les graphes ayant une bonne décomposition arborescente car les sacs de la décomposition arborescente peuvent s'intersecter de manière non-triviale.

La technique appliquée pour obtenir le schéma sur les k-arbres peut aussi être utilisée pour obtenir un schéma k-relationnel sur les arbres. Évidemment, les informations encodées et le test différent quelque peu mais les principes généraux restent les mêmes. Le schéma obtenu, qui permet entre autre le calcul exact de distance entre sommets à distance au plus k améliore un résultat récent de [6, 7] de manière non-triviale. Ces travaux ont fait l'objet de trois présentations dans des conférences internationales [46, 48, 47].

5.2 Schéma d'adjacence pour les graphes de largeur arborescente bornée

Dans cette section, on s'intéressera à prouver le résultat suivant :

Théorème 16. *La famille des graphes de largeur arborescente k ayant n sommets admet un schéma d'adjacence avec des étiquettes de $\log n + O(k \log\log(n/k))$ bits. De plus, l'adjacence peut être testée en temps constant et le calcul des étiquettes peut s'effectuer en temps $O(n \log n)$ pour k fixé.*

Le calcul des étiquettes est basé sur la construction d'un super-graphe triangulé dont la taille de clique maximale est minimale (c'est-à-dire $k+1$ puisque le graphe est de largeur arborescente k). La construction d'un tel super-graphe pour un graphe quelconque est un problème NP-complet [10]. Néanmoins, puisqu'on se restreint aux graphes de largeur arborescente bornée on peut alors utiliser des algorithmes linéaires [18].

Le schéma pour les graphes planaires est ainsi un corollaire direct du théorème 16.

Corollaire 4. *La famille des graphes planaires de n sommets admet un schéma d'adjacence avec des étiquettes de $2\log n + O(\log\log n)$ bits. De plus, l'adjacence peut être testée en temps constant et le calcul des étiquettes peut s'effectuer en temps $O(n \log n)$.*

Il est possible d'étendre cette construction au cas des graphes de genre d'Euler g puisqu'ils peuvent être partitionnés en deux graphes de largeur arborescente au plus $O(g)$ [35].

Corollaire 5. *La famille des graphes de genre d'Euler g ayant n sommets admet un schéma d'adjacence avec des étiquettes de $2\log n + O(g \log\log(n/g))$ bits. De plus, l'adjacence peut être testée en temps constant et le calcul des étiquettes peut s'effectuer en temps $O(n \log n)$ pour g fixé.*

De même, on peut déduire de la partition des arêtes des graphes n'ayant pas K comme mineur en deux graphes de largeur arborescente majorée par une fonction $f(K)$ dépendant de K [34], un schéma pour les graphes excluant un mineur fixé.

Corollaire 6. *Il existe une fonction f telle que la famille des graphes de n sommets n'ayant pas le graphe K comme mineur admet un schéma d'adjacence avec des étiquettes de $2\log n + O(f(K)\log\log(n/f(K)))$ bits. De plus, l'adjacence peut être testée en temps constant et le calcul des étiquettes peut s'effectuer en temps $O(n\log n)$ pour K fixé.*

5.2.1 (k, s)-triangulation et bidécomposition

Notre schéma repose sur les graphes triangulés. En effet, chaque graphe G de largeur arborescente k est un sous-graphe couvrant d'un graphe triangulé dont la clique maximale est de taille $k+1$ appelé *triangulation* de G. Les graphes triangulés de clique maximale $k+1$ ont deux propriétés importantes qui vont nous être utile pour notre schéma. Premièrement, dans ses graphes, il existe une sous-clique dont la suppression sépare le graphe en composantes connexes de taille au plus la moitié de la taille du graphe. Deuxièmement, il existe un schéma ordonné d'élimination de sommet appelles *schéma d'élimination simpliciel*, tel que le voisinage de chaque sommet enlevé est une clique dans le graphe restant.

Un graphe G est dit *s-séparable* si chacun de ses sous-graphes H admet un demi-séparateur de taille s, c'est-à-dire un ensemble de s sommets dont la suppression sépare H en composantes connexes de taille au plus $|V(H)|/2$ sommets. Un graphe est dit *k-clique orientable* si ses arêtes peuvent être orientées de telle sorte que le voisinage sortant de chaque sommet induise une clique de taille au plus k. Une telle orientation est appelée une k-clique orientation.

Définition 6. *Une* (k, s)-triangulation *est un graphe qui est k-clique orientable et s-séparable. Un* (k, s)-graphe *est un sous-graphe d'une (k, s)-triangulation.*

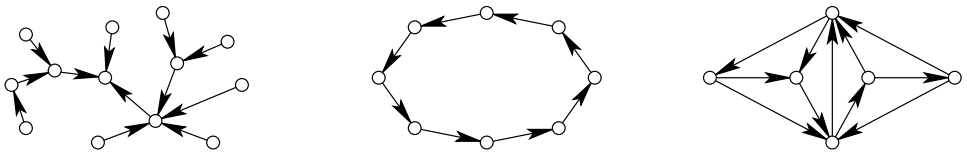


Figure 32 – Une $(1, 1)$-triangulation, une $(1, 2)$-triangulation, et une $(2, 2)$-triangulation.

D'après cette définition, les graphes de largeur arborescente k sont des $(k, k + 1)$-triangulations puisque leurs triangulations sont $(k + 1)$-séparables et que leur schéma d'élimination simpliciel nous donne une k-clique orientation. Cependant, les forêts sont des $(1, 1)$-graphes bien qu'elles soient de largeur arborescente 1. Les cycles (de largeur arborescente 2) sont 1-clique orientables et 2-séparables, et donc sont des $(1, 2)$-graphes (ce sont aussi $(1, 2)$-triangulations, voir la figure 32).

Les cliques de k sommets sont $(\lceil k/2\rceil, \lceil k/2\rceil)$-graphes bien qu'elles soient de largeur arborescente $k-1$. On en déduit que la famille des $(k, k+1)$-graphes contient strictement la famille des graphes de largeur arborescente k.

Un *k-complexe* d'un graphe G est un sous-graphe K isomorphe à une étoile d'au plus k feuilles. Sa *racine*, dénotée par racine(K), est l'unique sommet interne de K. Deux

complexes K et K' sont dits adjacents si racine$(K) \neq$ racine(K'), et soit racine$(K) \in V(K')$ ou racine$(K') \in V(K)$. Clairement, si deux complexes sont adjacents alors leurs racines sont adjacentes dans G.

Il est possible d'associer à chaque sommet u d'un (k, s)-graphe G un k-complexe K_u ayant u comme racine tel que toutes les arêtes soient couvertes par exactement un k-complexe. On procède de la manière suivante : K_u est composé de u et de ses voisins sortant dans la (k, s)-triangulation $\tilde{G}$ de G. On en déduit un schéma d'adjacence trivial consistant à étiqueter u par un code des sommets de $V(K_u)$. Il est clair que u et v sont adjacents si et seulement si leurs complexes associés K_u et K_v le sont. Ceci nous donne de manière triviale un schéma avec des étiquettes de $(k+1)\lceil \log n \rceil$ bits. En fait, il est possible d'obtenir un codage bien plus compact en utilisant le fait que dans la triangulation $\tilde{G}$, $V(K_u)$ induit une clique. Dans ce but, on a besoin d'une partition particulière des sommets de G.

Définition 7. *Une* bidécomposition *d'un graphe G est un arbre binaire enraciné T dont les sommets, appelés* parts*, forment une partition de $V(G)$ telle que les parts des extrémités de chaque arête de $E(G)$ sont liées*[1] *dans T.*

La part de T contenant le sommet u est notée T_u. Une observation directe que l'on peut faire à partir de la définition de T est que les parts des sommets d'un sous-graphe $K \subseteq G$ sont deux à deux liés si K est une clique.

5.2.2 Trouver une bidécomposition appropriée

Afin d'obtenir un codage succinct pour les étiquettes des sommets, il est nécessaire de calculer une bidécomposition de $\tilde{G}$ ayant des parts les plus petites possibles.

Lemme 13. *Soit $\tilde{G}$ un graphe s-séparable de n sommets. Alors, $\tilde{G}$ admet une bidécomposition telle que les parts à une profondeur h ont au plus $s\lfloor \log(2n/s) - h \rfloor$ sommets de T. De plus, une telle bidécomposition peut être calculée en temps $O(sn \log(n/s))$ si un demi-séparateur de taille s peut être calculé pour tout sous-graphe H de $\tilde{G}$ en temps $O(|V(H)| + |E(H)|)$.*

Démonstration. Soit $\tilde{G}$ un graphe s-séparable de n sommets. Un *biséparateur* B d'un graphe G est un sous-ensemble de $V(G)$ tel que l'on puisse partitionner $V(G) \setminus B$ en deux parts V_1 et V_2 sans arêtes $x_1x_2 \in E(G)$ avec $x_1 \in V_1$ et $x_2 \in V_2$. Le principe de l'algorithme de bidécomposition est de calculer un biséparateur du graphe, qui va constituer la racine de l'arbre de bidécomposition, et de réappliquer récursivement le processus sur les deux parts V_1 et V_2 afin d'obtenir les deux sous-arbres induits par les fils de la racine.

Le biséparateur B et la partition (V_1, V_2) de $V(\tilde{G}) \setminus B$ sont calculés grâce à la procédure suivante appelé BISÉPARATEUR (algorithme 1), où DEMI-SÉPARATEUR(H, s) est la sous-routine calculant un demi-séparateur de taille exactement s pour le graphe H. En effet, on va forcer à ce que chaque séparateur soit de taille s, à l'exception du dernier séparateur

1. Deux sommets d'un arbre enraciné sont *liés* si un sommet est l'ancêtre de l'autre.

qui peut être de taille moindre par manque de sommets restants, en rajoutant au besoin des sommets dans le séparateur.

Algorithme 1 : Biséparateur

Entrées : Un graphe s-séparable $\tilde{G}$
Sorties : Un biséparateur B pour $\tilde{G}$ et la partition de sommet associée (V_1, V_2)

$H := \tilde{G}$; $V_1 := V_2 := B := \varnothing$
tant que $|V(H)| > s$ **faire**
  $S :=$ Demi-Séparateur(H, s) ; $H := H \setminus S$; $B := B \cup S$
  pour chaque *composante connexe C de H exceptée la plus grande* **faire**
    $H := H \setminus C$;
    si $|V_1| > |V_2|$ **alors** $V_2 := V_2 \cup V(C)$ **sinon** $V_1 := V_1 \cup V(C)$
$B := B \cup H$

À chaque itération de la boucle principale de Biséparateur, la taille de H est divisé par au moins deux puisque les composantes connexes résultantes de $H \setminus S$ sont retirées de H, et la composante connexe restante, celle étant la plus grande, est de taille $|V(H)|/2$. De plus, chaque séparateur à un taille s. Par conséquent, il y a au plus $\log(n/s)$ itérations.

À chaque étape, la taille de B est augmentée d'au plus s. Durant l'étape finale, au plus s sommets sont ajoutés à B. Par conséquent, $|B| \leqslant s \cdot \lfloor \log(n/s) \rfloor + s = s \cdot \lfloor \log(2n/s) - h \rfloor$ avec $h = 0$.

Afin d'assurer de la correction de la bidécomposition, on montre l'invariant de boucle suivant.

$$(P) : \quad |V_1|, |V_2| \leqslant n/2 \quad \text{et} \quad V(H) \cup V_1 \cup V_2 \cup B = V(\tilde{G})$$

Il est facile de voir que (P) est vérifiée au début de l'exécution de la boucle principale. Il suffit de montrer que cet invariant de boucle reste vérifié à la fin de l'exécution de la boucle imbriquée. L'invariant reste clairement vérifié après le calcul du demi-séparateur et de la ligne $H := H \setminus S$ et $B := B \cup S$. On suppose sans perte de généralité que $|V_1| \geqslant |V_2|$. On obtient la relation suivante pour $|V_1|$, $|V_2|$ et $|B|$.

$$\begin{aligned}
|V(H)| + |V_1| + |V_2| &\leqslant n \\
|V(H)| &\leqslant n - |V_1| - |V_2| \leqslant n - 2|V_2| \quad (|V_1| \geqslant |V_2|) \\
|V(H)|/2 &\leqslant n/2 - |V_2| \\
|V(C)| &\leqslant n/2 - |V_2| \\
&\quad \text{puisqu'il existe une composante connexe plus grande dans } H \\
|V(C)| + |V_2| &\leqslant n/2
\end{aligned}$$

L'invariant (P) reste donc vérifié à la fin de la boucle imbriquée et donc à la fin de la boucle principale. On en déduit que $|V_1|, |V_2| \leqslant n/2$. De plus, il n'y a pas d'arêtes liant les

sommets de V_1 aux sommets de V_2 puisque les sommets ajoutés dans V_1 ou V_2 sont des composantes connexe de H.

Un arbre T ayant pour sommets des parties de $\tilde{G}$ est obtenue par l'application récursive de cette procédure sur les sous-graphes induits par V_1 et V_2. On relie à B (la racine de T) les arbres résultants de la procédure sur V_1 et V_2 si ceux-ci sont non vides. On cherche à prouver que T est une bidécomposition. Supposons par l'absurde que les deux extrémités d'une même arêtes soient dans des parts U et V qui ne sont pas ancêtre l'une de l'autre dans l'arbre T. Soit W le plus petit ancêtre commun à U et V. Par construction, W est un biséparateur et U et V sont dans des parts différentes par rapport à ce biséparateur. Par définition, aucune arête ne relie les sommets des deux parts d'une biséparation. On obtient donc une contradiction et T est une bidécomposition.

Puisque la taille V_1 et V_2 est inférieur ou égale à $n/2$, on en déduit que la taille de leurs biséparateurs (et donc des parts de profondeur $h = 1$ de T) est au plus $s \cdot \lfloor \log(2(n/2)/s)) \rfloor = s \cdot \lfloor \log(2n/s) - 1 \rfloor$. Plus généralement, pour une profondeur $h \geqslant 0$, les parts sont de taille $s \cdot \left\lfloor \log(2(n/2^h)/s)) \right\rfloor = s \cdot \lfloor \log(2n/s) - h \rfloor$.

Avant d'aborder le problème de la complexité en temps de l'algorithme, il est utile d'observer que chaque sous-graphe H de $\tilde{G}$ a $O(s|V(H)|)$ arêtes. En effet, il est connu [81] que les graphes s-séparables ont une largeur arborescente au plus $4s$. De surcroît, les graphes de largeur arborescente k ayant n sommets n'ont pas plus de kn arêtes à cause du schéma d'élimination simpliciel de leur triangulation. On en déduit que H a au plus $4s|V(H)|$ arêtes.

Il ne reste donc plus à prouver que le calcul du biséparateur prend un temps linéaire. A chaque itération de la boucle principale, un séparateur de H est calculé en temps $O(|V(H)| + |E(H)|)$ d'après les hypothèses du lemme. Les modifications des ensembles V_1, V_2, et H sont aussi déterminées en temps $O(|V(H)| + |E(H)|)$. La i-ème itération prend un temps $O(|V(H)| + |E(H)|) = O(sn/2^i)$. Le temps de calcul total de la bidécomposition est donc $t(n) \leqslant c \cdot sn + 2t(n/2)$ si $n > s$, et $t(n) = O(1)$ sinon. Au final, cela nous donne $t(n) = O(sn \log(n/s))$. □

5.2.3 Les étiquettes du schéma

On considère un (k, s)-graphe G de n sommets. Par définition, G est le sous-graphe d'une (k, s)-triangulation $\tilde{G}$. Soit T une bidécomposition de $\tilde{G}$ satisfaisant le lemme 13. Pour chaque sommet u de $V(G)$, on considère K_u le complexe enraciné en u dans G induit par les arêtes sortantes de u dans la k-clique orientation. Par construction, $V(K_u)$ induit une clique de taille au plus $k + 1$ dans $\tilde{G}$. Par conséquent, les parts des sommets de K_u sont deux à deux liées dans T.

On définit la fonction $\beta(h) = \sum_{i=0}^{h} s \lfloor \log(2n/s) - i \rfloor$. Intuitivement, $\beta(h)$ représente le nombre maximum de sommets de G contenus dans les parts d'une branche[2] de T de longueur h. On a aussi $\beta(h) = s(h + 1)(\lfloor \log(2n/s) \rfloor - h/2)$.

2. Un chemin ayant la racine de T comme extrémité.

Soit X une part de T de profondeur h, la racine de T étant de profondeur 0. On dénote par chemin(X) le mot binaire définissant le chemin unique allant de la racine de T à X. La longueur de chemin(X) est $|\text{chemin}(X)| = h$. À chaque sommet $u \in X$, on associe son *rang*, un entier unique dans $[0, |X|[$, et sa *position*, définie par $\text{pos}(u) = \text{rang}(u) + \beta(h-1)$. L'*apex* de u est le sommet a_u de K_u dont la position est maximum, c'est-à-dire, tel que $\text{pos}(a_u) = \max_{v \in V(K_u)} \text{pos}(v)$.

On peut remarquer que les positions sont relatives à une branche de T : n'importe quelle paire de sommets distincts dont les parts sont dans la même branche on des positions distinctes. Donc, les parts de sommets ayant la même position ne sont pas liées dans T.

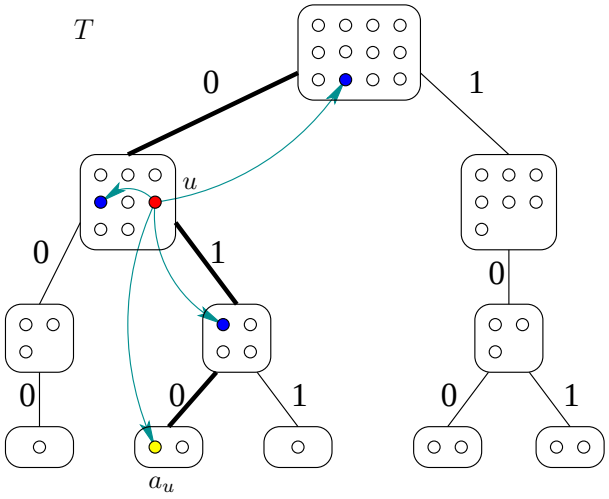


Figure 33 – Une bidécomposition avec un complexe K_u d'un sommet u ayant un apex a_u.

Soit u un sommet de G, et soit a_u l'apex de K_u dans T. L'étiquette $\ell(u)$ du sommet u est égale au quadruplet suivant :

$$\ell(u) = (\text{chemin}(T_{a_u}), \text{rang}(a_u), P_u, r_u)$$

où $P_u = \{\text{pos}(v) \mid v \in V(K_u), v \neq a_u\}$ et $r_u = |\{p \in P_u \mid p < \text{pos}(u)\}|$.

P_u est l'ensemble des positions de tous les sommets de K_u excepté celle de son apex, et r_u est le rang de la racine de K_u dans P_u. Donc $r_u = 0$ si $\text{pos}(u)$ est la plus petite position dans P_u et $r_u = |P_u|$ si $u = a_u$ est l'apex lui-même.

Soit $\text{pos}(K_u) = \{\text{pos}(v) \mid v \in V(K_u)\}$. Il est facile de voir que le complexe K_u est défini de manière unique par la paire $(\text{pos}(K_u), \text{chemin}(T_{a_u}))$, c'est-à-dire l'ensemble des positions et le chemin ayant l'apex comme extrémité. En effet, comme on l'a dit précédemment, les sommets contenus dans les parts d'une même branche ont des positions distinctes et les sommets de branches différentes ont bien évidemment des apex différents. L'ensemble $\text{pos}(K_u)$ n'est pas un champ de $\ell(u)$ puisqu'il manque $\text{pos}(a_u)$ à l'ensemble P_u. Néanmoins, il peut être calculé puisque $\text{pos}(a_u) = \text{rang}(a_u) + \beta(|\text{chemin}(T_{a_u})|)$. Finalement, $\ell(u)$ est un codage injectif de K_u. En particulier, si $u \neq v$ alors $K_u \neq K_v$ et donc $\ell(u) \neq \ell(v)$.

Lemme 14. *Les étiquettes ont* $\log n + 2k \log\log(n/s) + O(k \log(s/k))$ *bits.*

Démonstration. On suppose que n, k, s sont fixés. Soit $w = \lfloor \log(2n/s) \rfloor$, et soit $h = |\text{chemin}(T_{a_u})|$. Le mot binaire $\text{chemin}(T_{a_u})$ a pour taille h, et $\text{rang}(a_u) \in [0, \beta(h) - \beta(h-1)[$. On a $\beta(h) - \beta(h-1) = s \lfloor \log(2n/s) - h \rfloor = s(w-h)$.

On a $\text{rang}(a_u) = x \cdot (w-h) + y$ où $x \in [0, s[$ et $y \in [0, w-h[: x = \lfloor \text{rang}(a_u)/(w-h) \rfloor$ et $y = \text{rang}(a_u) \bmod (w-h)$. On code les deux premiers champs $\text{chemin}(T_{a_u}), \text{rang}(a_u)$ par le mot binaire suivant :

$$W \;=\; \text{chemin}(T_{a_u}) \circ \text{bin}_s(x) \circ 10^y$$

où $\text{bin}_s(x)$ est la représentation binaire de x sur $\lfloor \lg(s) \rfloor + 1$ bits. Pour W donné (mais sans h), on peut extraire, y et $\text{bin}_s(x)$, et donc $\text{chemin}(T_{a_u})$, h, et $\text{rang}(a_u) = x \cdot (w-h) + y$. La longueur de W est $|S| \leqslant h + (\lfloor \log s \rfloor + 1) + 1 + w - h = w + \lfloor \log s \rfloor + 2 = \lfloor \log(2n/s) \rfloor + \lfloor \log s \rfloor + 2 \leqslant \log n + O(1)$.

Toutes les positions de P_u sont comprises dans $[0, \beta(h)[$. Il s'en suit qu'il y a au plus $\binom{\beta(h)}{|P_u|} \leqslant \binom{\beta(h)}{k}$ façons de sélectionner un ensemble P_u ayant au plus k positions. On peut donc coder cet ensemble avec $\log \binom{\beta(h)}{k} + O(1)$ bits. En fait, $r_u \in [0, |P_u|]$, et par conséquent il y a $k+1$ valeurs possibles. On peut coder r_u avec $\lfloor \lg(k+1) \rfloor + 1$ bits. Au total, la taille de $\ell(u)$ est au plus :

$$|\ell(u)| \;=\; \log n + \log \binom{\beta(h)}{k} + O(\log k) \tag{3}$$

D'après le lemme 13, la taille des parts de T de profondeur $\log(n/s)$ est de $s \cdot (\log(2n/s) - \log(n/s)) = s$. D'après la condition de la boucle principale de l'algorithme BISÉPARATEUR, si $|V(H)| \leqslant s$, alors le graphe n'est pas séparé, impliquant que la part est une feuille de T. Par conséquent $h \leqslant \log(n/s)$.

On a $\beta(h) \leqslant \beta(\log(n/s)) \leqslant (\log(n/s)+1) \cdot s \cdot \log(2n/s) = s \cdot \log^2(2n/s)$. Donc $\log \binom{\beta(h)}{k} \leqslant k \log(\beta(h)/k) + O(k) \leqslant k \log(s/k \cdot \log^2(2n/s)) + O(k) \leqslant 2k \log\log(n/s) + O(k \log(n/k))$. En combinant ceci avec l'équation (3), on obtient le résultat désiré. □

Donc, pour les k-arbres, qui sont des $(k, k+1)$-triangulations, on obtient des étiquettes de $\log n + 2k \log\log(n/k) + O(1)$ bits. Pour les $(1,1)$-triangulations (arbres), on a des étiquettes de $\log n + 2 \log\log n + O(1)$ bits.

5.2.4 Le test d'adjacence

Soit u et v deux sommets de $\tilde{G}$. On note $M = \beta(h)$ où h est la longueur du plus long préfixe commun à $\text{chemin}(T_{a_u})$ et $\text{chemin}(T_{a_v})$.

Lemme 15. *Les sommets u et v sont adjacents dans G si et seulement si $\ell(u) \neq \ell(v)$ et si soit* $\text{pos}(u) \in \text{pos}(K_v) \cap [0, M)$ *ou* $\text{pos}(v) \in \text{pos}(K_u) \cap [0, M)$.

Démonstration. Tout d'abord, il faut ramener le problème à l'adjacence entre deux k-complexes. On utilise la condition suivante. Les sommets u et v sont adjacents si et seulement si $K_u \neq K_v$ et $u \in V(K_v)$ ou $v \in V(K_u)$. Le test $K_u \neq K_v$ permet de s'assurer que

les k-complexes ont des racines différentes puisque à chaque sommet est associé un seul k-complexe. Le double test d'appartenance permet de déterminer si l'un des deux sommets est compris dans l'autre k-complexe auquel cas les deux sommets sont adjacents. La condition est nécessaire car si l'arête uv est orientée de u vers v dans la k-clique orientation alors $v \in V(K_u)$, et $u \in V(K_v)$ si l'arête est orientée dans l'autre sens.

Il reste donc à montrer que cette condition sur les k-complexes est équivalente à celle du lemme. On considère B_u la branche de T correspondant au k-complexe K_u et B_v celle correspondant au k-complexe K_v et B la branche allant de la racine de T au sommet défini par le préfixe commun de longueur h de chemin(T_{a_u}) et chemin(T_{a_v}). On a $B_u \cap B_v = B$ puisque B correspond au plus long préfixe commun des mots définissant B_u et B_v et donc à la partie commune des deux branches.

Les sommets communs aux deux k-complexes sont donc forcément dans B. Les sommets de B sont les sommets de $V(K_u)$ et $V(K_v)$ dont la position est inférieure strictement à M. Puisque ces sommets sont dans la même branche B, ils sont identifiés de manière unique par leurs positions. On en déduit que la condition d'adjacence est correcte puisqu'elle se résume à vérifier que $K_u \neq K_v$ avec la condition $\ell(u) \neq \ell(v)$ et à vérifier $u \in V(K_v)$ ou $v \in V(K_u)$ avec la condition $\text{pos}(u) \in \text{pos}(K_v) \cap [0, M)$ ou $\text{pos}(v) \in \text{pos}(K_u) \cap [0, M)$. □

On va maintenant décrire l'implémentation du test d'adjacence en temps constant utilisant le modèle RAM avec des mots $\Omega(\log n)$ bits. Premièrement, la valeur $\beta(h)$ peut être calculée en temps constant à partir de h en utilisant la formule suivante : $\beta(h) = s(h+1)(\lfloor s \log(2n/s) \rfloor - h/2)$. Par conséquent, M peut être calculé en temps constant car le préfixe commun à deux mots de $O(\log n)$ bits peut être calculé en utilisant un nombre constant d'opérations de recherche du bit de poids forts, de masquage et de décalage.

Il reste maintenant à tester si $\text{pos}(v) \in \text{pos}(K_u)$. Pour cela il suffit de tester si $\text{pos}(v) = \text{pos}(a_u)$ ou si $\text{pos}(v) \in P_u$. La position de l'apex peut être calculée d'après la formule suivante $\text{pos}(a_u) = \text{rang}(a_u) + \beta(|\text{chemin}(T_{a_u})|)$. La position de u peut être déterminée à partir de $\ell(u)$, puisque $\text{pos}(u)$ est égale à $\text{pos}(a_u)$ si $r_u = |P_u|$, et le r_u-ème élément de P_u sinon. Le test $\ell(u) \neq \ell(v)$ peut être effectué en vérifiant que $(\text{pos}(u), \text{chemin}(T_{a_u})) \neq (\text{pos}(v), \text{chemin}(T_{a_v}))$. Finalement, calculer l'appartenance de $\text{pos}(u)$ à l'ensemble P_v se résume à des requêtes d'appartenance et de sélection sur des ensembles d'entiers $P_u, P_v \subset \{0, \ldots, \beta(h)\}$.

En utilisant un dictionnaire statique, on peut implémenter ce genre de requêtes en temps constant avec une structure de données de $(1 + o(1)) \log \binom{\beta(h)}{|P_u|}$ bits [76]. On obtient donc un schéma d'adjacence en temps constant tout en conservant des étiquettes de $(1 + o(1)) \log \binom{\beta(h)}{k} \leqslant (2k + o(1)) \log \log(n/s)$ bits.

Le théorème 16 est déduit du fait que les graphes de largeur arborescente k sont des $(k, k+1)$-graphes et que leur $(k, k+1)$-triangulations peuvent être calculées en temps $O(n)$ pour k fixé [18].

5.3 Minorant pour la taille des étiquettes d'un schéma d'adjacence pour les graphes de largeur arborescente k

D'après [58], chaque schéma d'adjacence pour une famille de graphes ayant $F(n)$ graphes étiquetés de n sommets requiert des étiquettes d'au moins $\frac{1}{n}\log F(n)$ bits.

Bien qu'il soit trivial de montrer un minorant de $\max\{\log n, (k+1)/2\} = \Omega(\log n + k)$ bits pour les schémas des graphes de largeur arborescente k (en utilisant juste le fait que les n étiquettes d'un graphes sont nécessairement distinctes et qu'il existent $2^{\binom{k+1}{2}}$ graphes de largeur arborescente k ayant $k+1$ sommets), il est moins facile de prouver un minorant de $\log n + \Omega(k)$ pour la même famille.

Théorème 17. *Le nombre $T(n,k)$ de graphes étiquetés de largeur arborescente k ayant n sommets satisfait la relation suivante : $\frac{1}{n}\log T(n,k) = \log n + \Theta(k)$.*

On montre ce théorème en deux étapes, prouvant d'abord un minorant par le lemme 16 puis ensuite un majorant dans le lemme 17. Le minorant prouvé est en fait plus fort que le résultat nécessaire pour le théorème puisqu'il s'applique à une sous-famille stricte des graphes de largeur arborescente k, les graphes de largeur linéaire k. La *largeur linéaire* d'un graphe est le minimum des largeurs des décompositions arborescentes dont l'arbre de décomposition est un chemin.

Lemme 16. *Le nombre $P(n,k)$ de graphes connexes de largeur linéaire k ayant n sommets satisfait la relation suivante : $\frac{1}{n}\log P(n,k) = \log n + \Omega(k)$.*

Démonstration. L'idée de la preuve consiste à énumérer une sous-famille $\mathcal{F}_{n,k}$ de graphes étiquetés de largeur linéaire $k' = 2k$ ayant $n' = 2n+2$ sommets.

On suppose sans perte de généralité que n est divisible par k. On définit un *graphe ordonné* comme étant un graphe doté d'un ordre total sur ses sommets. Soit $\mathcal{S}_{n,k}$ l'ensemble des paires (S,P) où $S = (S_1, \cdots, S_{n/k})$ est une suite de n/k graphes ordonnés de k sommets et P est une permutation de $2n+2$ éléments. Les graphes de $\mathcal{F}_{n,k}$ sont construits à partir d'élément de $\mathcal{S}_{n,k}$ en utilisant la construction ci-après. Soit $(S,P) \in \mathcal{S}_{n,k}$. Le graphe G issu de (S,P) est composé d'un chemin $v_{0,k}v_{1,1}v_{1,2}\cdots v_{1,k}v_{2,1}v_{2,2}\cdots v_{n/k,k}$ de longueur $n+1$, d'un sommet x relié à tous les sommets $v_{i,j}$ pour tous $i \leqslant n/k$ et $j \leqslant k$ et des graphes de P. On note $u_{i,j}$ le j-ème sommet du graphe ordonné S_i. Pour tous $j \in \{1,\ldots,k\}$ et $i \in \{1,\ldots,n/k\}$, on relie les sommets $u_{i,j}$ et $v_{i,j}$. Finalement, on utilise la permutation P afin d'étiqueter les sommets de G. Le premier entier de P est l'étiquette de x, les $n+1$ entiers suivants servent à étiqueter les sommets du chemin $v_{0,k}\cdots v_{n/k,k}$ et les n entiers restants servent à étiqueter les sommets de $u_{1,1}$ à $u_{n/k,k}$. La figure 34 nous donne un exemple d'un tel graphe.

Les graphes de $\mathcal{F}_{n,k}$ ont une largeur linéaire inférieure ou égale à $2k$. En effet, on peut construire une décomposition linéaire de largeur $2k$ en prenant pour sacs B_i et D_i décrits comme suit : $B_i = \bigcup_{j=1}^{k}\{x, v_{i,j}, u_{i,j}\}$ et $D_i = \{x, v_{i-1,k}, v_{i,1}\}$. Le chemin de la décomposition

est $D_1B_1D_2\cdots B_{n/k}$. Il est facile de vérifier que la décomposition ainsi obtenue respecte les trois conditions énoncées par la définition de la décomposition linéaire.

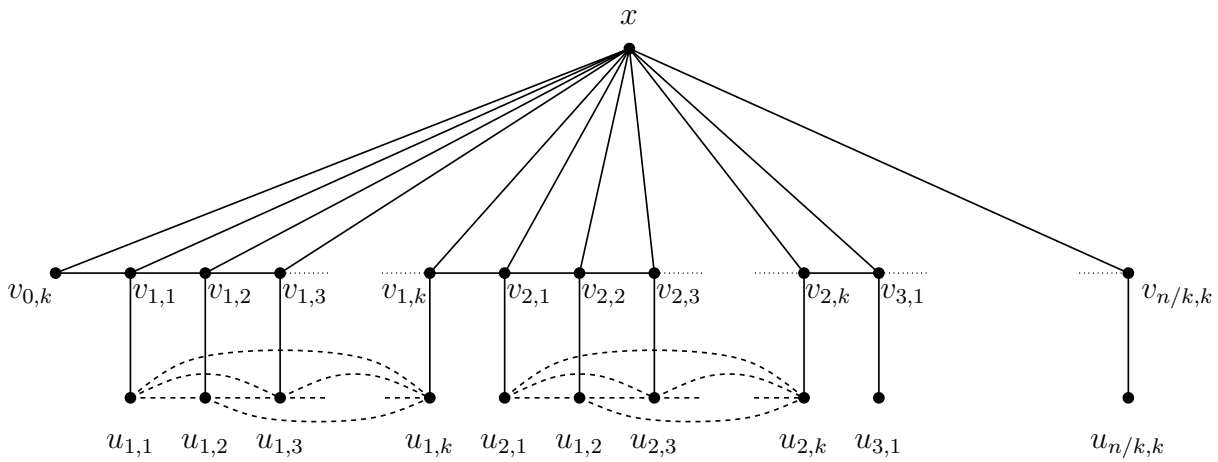


Figure 34 – Un graphe de $\mathcal{F}_{n,k}$.

La partie principale de la preuve consiste à prouver que la construction de $\mathcal{S}_{n,k}$ est injective. Afin d'assurer cette propriété on utilise le fait suivant.

Fait 5. *Soit G un graphe de $\mathcal{F}_{n,k}$. Il existe une manière unique d'identifier les sommets de G, c'est-à-dire de les identifier comme étant les sommets x, $u_{i,j}$ ou $v_{i,j}$ décrit dans la construction.*

A partir de G, il est possible d'identifier le sommet x puisque celui-ci est le seul sommet de degré supérieur à k. Le voisinage de x correspond par construction au chemin $v_{0,k}v_{1,1}v_{1,2}\cdots v_{1,k}v_{2,1}v_{2,2}\cdots v_{n/k,k}$. Afin d'identifier ses sommets dans le bon ordre, il suffit de se rappeler que $v_{0,k}$ est le seul sommet du chemin ayant un degré 2 dans G. Pour tous i,j, le seul sommet relié à $v_{i,j}$, qui n'est pas x ni un sommet du chemin, est le sommet $u_{i,j}$.

Il reste à montrer que notre construction est injective. Pour cela, on montre que deux paires distinctes (S,P) et (S',P') gênèrent des graphes distincts G et G'. D'après le fait 5, c'est évident. En effet, si les suites de graphes S et S' différent alors il existe une arête dans G qui n'est pas dans G' ou inversement. De même pour $P \neq P'$, les étiquetages de G et G' différent. Puisqu'on peut identifier les sommets, il est clair que les deux graphes différent. On en déduit que $|\mathcal{S}_{n,k}| \leqslant |\mathcal{F}_{n,k}|$. Il ne reste donc plus qu'à énumérer les éléments de $\mathcal{S}_{n,k}$.

Fait 6. $|\mathcal{S}_{n,k}| = (2n+2)!2^{\binom{k}{2}\cdot n/k}$

Il y a $(2n+2)!$ permutations de $2n+2$ éléments. Choisir un graphe parmi tous les graphes ordonnés de n sommets consiste à choisir pour chaque paire de sommets de mettre ou pas une arête reliant les deux sommets. Par conséquent, il y a $2^{\binom{k}{2}}$ exactement graphes

de ce type. Il existe $\left(2^{\binom{k}{2}}\right)^{n/k}$ suites de n/k graphes de ce type. Au final, il y a donc $(2n+2)!2^{\binom{k}{2}\cdot n/k}$ paires (S,P) dans $\mathcal{S}_{n,k}$.

Pour $\mathcal{F}_{n,k}$, on en déduit que :

$$\begin{aligned}
|\mathcal{F}_{n,k}| &\geqslant |\mathcal{S}_{n,k}| \\
|\mathcal{F}_{n,k}| &\geqslant (2n+2)!2^{\binom{k}{2}n/k} \\
P(2n+2,2k) &\geqslant (2n+2)!2^{\binom{k}{2}n/k} \\
P(2n+2,2k) &\geqslant \left(\frac{2n+2}{e}\right)^{2n+2} 2^{(k-1)n/2} \\
\log P(2n+2,2k) &\geqslant (2n+2)\log\left(\frac{2n+2}{e}\right) + (k-1)n/2 \\
\log P(n',k') &= n'\log n' + \Omega(k'n') \\
&\quad \text{où } n' = 2n+2 \text{ et } k' = 2k \\
\frac{1}{n'}\log P(n',k') &= \log n' + \Omega(k')
\end{aligned}$$

□

Lemme 17. *Le nombre $T(n,k)$ de graphes étiquetés de largeur arborescente k ayant n sommets satisfait la relation suivante : $\frac{1}{n}\log T(n,k) = \log n + O(k)$.*

Démonstration. Un graphe de largeur arborescente k est un sous-graphe d'un k-arbre [94]. Le nombre $C(n,k)$ de k-arbres étiquetés de n sommets est donnés par $C(n,k) = \binom{n}{k}(kn - k^2 + 1)^{n-k-2}$ [13, 39]. Soit G un k-arbre. Il y a au plus $2^{|E(G)|}$ sous-graphes différents de G car il suffit pour chaque arête de la mettre ou non dans le sous-graphe. Puisque les k-arbres de n sommets ont au plus kn arêtes, on en déduit que :

$$\begin{aligned}
T(n,k) &\leqslant C(n,k)\cdot 2^{nk} \\
T(n,k) &\leqslant \binom{n}{k}\cdot(kn-k^2+1)^{n-k-2}\cdot 2^{nk} \\
T(n,k) &\leqslant n^k\cdot(kn)^n\cdot 2^{nk} \\
\log T(n,k) &\leqslant k\log n + n(\log k + \log n) + nk \\
\frac{1}{n}\log T(n,k) &\leqslant \log n + k\left(\frac{\log n}{n} + \frac{\log k}{k} + 1\right) \leqslant \log n + O(k)
\end{aligned}$$

□

5.4 Schéma k-relationnel

Le schéma d'adjacence pour les graphes de largeur arborescente bornée décrit dans la section précédente peut être utilisé sur les arbres puisque ceux-ci sont de largeur arborescente 1. On obtient de cette manière un schéma avec des étiquettes de $\log n + O(\log \log n)$ bits. L'avantage par rapport au schéma en $\log n + O(\log^* n)$ bits [9] est le fait qu'on puisse tester si deux sommets sont adjacents ou s'ils sont frères. En effet, notre technique consiste pour chaque sommet v de l'arbre à stocker les identifiants des sommets du 1-complexe de v. En choisissant comme 1-clique orientation l'orientation du fils vers le père, le 1-complexe de chaque sommet v est constitué de v et de son père. On peut donc effectivement tester si deux sommets sont frères en vérifiant s'ils ont le même père ce qui est possible car chaque sommet code un identifiant de son père en plus du sien.

En fait, il est possible de construire un schéma qui permet le calcul de requêtes plus complexes. On considère un arbre enraciné T. Deux sommets u et v de T ayant comme plus petit ancêtre commun w sont dit (k_1, k_2)-*reliés* si la distance de u à w est k_1 et la distance de v à w est k_2. Pour un entier k, on définit un *schéma k-relationnel* comme étant une représentation distribuée d'arbres supportant des requêtes afin de déterminer si u et v sont (k_1, k_2)-reliés pour $k_1, k_2 \leqslant k$.

Par exemple, un schéma 1-relationnel permet de tester si deux sommets sont $(0, 0)$, $(1, 0)$, $(0, 1)$ ou $(1, 1)$-reliés, et donc supporte des requêtes de parenté (test afin de déterminer si un des deux sommets et le père de l'autre) et de fraternité (test afin ce déterminer si les deux sommets sont frères). Plus généralement, un schéma k-relationnel permet le calcul des distances entre deux sommets à distance au plus k.

En utilisant une technique similaire à celle utilisée pour les graphes de largeur arborescente k, il est possible de construire un schéma k-relationnel pour les arbres de n sommets avec des étiquettes de $\log n + O(k \log(k \log(n/k)))$ bits. On améliore ainsi le meilleur schéma connu qui utilise des étiquettes de $\log n + O(k^2 \log(k \log n))$ bits [6, 7]. De plus, ce schéma est simple et donc facile à implémenter. Notre schéma peut être utilisé pour calculer efficacement des distances entre des sommets à distance bornée. Plus précisément, il est possible de calculer la distance exacte entre des sommets à distance au plus $o(\log n / \log \log n)$ avec des étiquettes de $\log n + o(\log n)$ bits. Sachant que $\Omega(\log^2 n)$ bits par sommets sont nécessaire pour le calcul des distances entre n'importe quels sommets, il peut parfois être utile d'utiliser notre schéma qui est plus adapté pour certains cas particuliers comme les arbres de faible diamètre.

Le résultat principal de cette section peut se résumer en le théorème suivant :

Théorème 18. *La famille des arbres enracinés de n sommets admet un schéma k-relationnel avec des étiquettes de $\log n + O(k \log(k \log(n/k)))$ bits. De plus les requêtes s'effectuent en temps constant et l'étiquetage prend un temps $O(n \log n)$ pour k fixé.*

Pour $k = 1$, on montre que les étiquettes sont de longueur $\log n + 2 \log \log n + O(1)$, améliorant ainsi les étiquettes de $\log n + 5 \log \log n + O(1)$ bits du schéma de [7]. De plus, on déduit du théorème 18 que :

Corollaire 7. *La famille des arbres enracinés de n sommets admet une représentation distribué avec des étiquettes $\log n + o(\log n)$ bits, qui supporte des requêtes de distance pour les sommets à distance au plus $o(\log n / \log\log n)$.*

5.4.1 Principe général du schéma

Le principe du schéma est tout simplement de stocker dans l'étiquette de chaque sommet u des identifiants de u et de ses k ancêtres les plus proches. En effet, afin de tester si u et v sont (k_1, k_2)-reliés, il suffit de tester si l'ancêtre à distance k_1 de u est égal à l'ancêtre à distance k_2 de v et si l'ancêtre à distance $k_1 - 1$ de u est différent à l'ancêtre à distance $k_2 - 1$ de v. Le premier test permet de vérifier si le sommet est bien un ancêtre commun et le deuxième permet de s'assurer que c'est bien le plus petit ancêtre commun. Une implémentation naïve de ce schéma aboutirait à des étiquettes de $O(k \log n)$-bits. On peut diminuer significativement la taille des étiquettes en utilisant une technique similaire à celle utilisée précédemment pour encoder les identifiants des sommets des k-complexes dans une (k, s)-triangulation.

On considère un arbre enraciné T de n sommets. On définit la *k-ascendance* d'un sommet u comme l'ensemble des ancêtres de u à distance au plus k, u inclus. Afin de coder la k-ascendance de chaque sommet, on a besoin d'une décomposition similaire à la bidécomposition. La figure 35 donne un exemple d'une telle décomposition.

Définition 8. *Une* décomposition k-ascendante *d'un arbre enraciné T est un arbre binaire enraciné B dont les sommets, appelés* parts*, forment une partition de $V(T)$ telle que les sommets d'une même k-ascendance de T soient dans les parts d'une seule branche de B.*

5.4.2 Trouver une bonne décomposition k-ascendante

Lemme 18. *Chaque arbre enraciné T de n sommets admet une décomposition k-ascendante telle que les parts à une profondeur h ont au plus $(k+1)\lfloor \log(2n/(k+1)) - h \rfloor$ sommets de $V(T)$. De plus cette décomposition se calcule en temps $O(n \log n)$ pour k fixé.*

On dit que deux sommets u et v sont *k-reliés* s'il sont liés (l'un des sommets est ancêtre de l'autre) et que la distance entre les deux sommets est inférieure ou égale à k. L'idée de base de la preuve est de ramener le problème à un calcul de bidécomposition d'une $(k, k+1)$-triangulation. Pour cela, on construit une *k-augmentation de T*, obtenue en ajoutant une arête entre toutes les paires de sommets distincts k-reliés. On montre tout d'abord que le graphe obtenu est une $(k, k+1)$-triangulation.

Propriété 5. *Soit G une k-augmentation de d'un arbre enraciné T. Alors G est une $(k, k+1)$-triangulation.*

Démonstration. Il suffit de prouver que G est $k+1$-séparable et k-clique orientable. Une k-clique orientation simple consiste à orienter les arêtes des sommets vers leurs ancêtres.

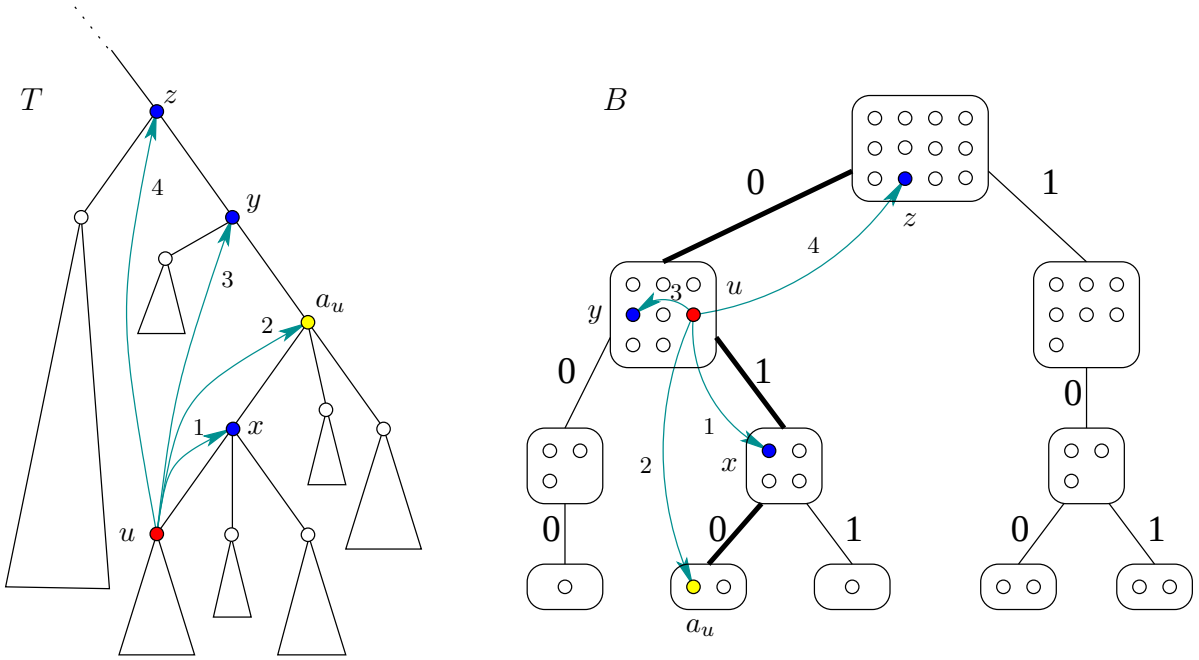


Figure 35 – Un exemple d'une décomposition k-ascendante d'un arbre T où la k-ascendance A_u (l'ensemble des cinq sommets colorés) sont sur la branche "010" de B.

Les voisins sortants de chaque sommet sont dans ce cas ses k ancêtres les plus proches. Ces k sommets sont clairement deux-à-deux k-reliés et donc forment une clique dans G.

Il est connu [51] que chaque graphe triangulé G est k-séparable où k est sa taille de clique maximale. Afin de prouver que G est $k+1$-séparable, il suffit donc de prouver que G ne contient pas de clique de taille $k+2$ et de cycle induit de taille supérieur à 3. Les sommets reliés par une arête dans G sont des sommets k-reliés dans T. Une clique K de G correspond donc à un ensemble de sommets deux à deux k-reliés dans T. Les sommets de K sont donc dans une même branche et donc un chemin dans T. De plus, ils sont à distance au plus k. Par conséquent, la clique K correspond à un chemin de longueur k dans T et contient donc au plus $k+1$ sommets.

Il reste à prouver que G est triangulé. On suppose maintenant par l'absurde que G a un cycle induit C de taille supérieure à 3. Soit u, v, w trois sommets de C tel que u est un ancêtre de v, et w est un ancêtre de v. De tels sommets existent car sinon C serait un chemin. Soit w est un ancêtre de u soit u est un ancêtre de w. Dans les deux cas, u et w sont liés et à distance au plus k dans l'arbre et donc sont adjacents dans G. On obtient une contradiction est donc G est bien triangulé. □

Démonstration du lemme 18. Soit T un arbre enraciné de n sommets. On construit G la k-augmentation de T. Puisque G est une $(k, k+1)$-triangulation, on peut calculer une bidécomposition B de G dont les parts à une profondeur h ont au plus $s\lfloor \log(2n/s) - h \rfloor$ sommets de $V(G)$. Cette bidécomposition B est une décomposition k-ascendante de T. En

effet, les sommets k-reliés dans T sont reliés par une arête dans G et sont donc dans une même branche de B. Par conséquent, les sommets d'une même k-ascendance de T sont bien dans les parts d'une seule branche de B. Le temps de calcul de B est de $O(n \log n)$ car un demi-séparateur d'un sous-graphe H de G se calcule en temps $O(|V(H)|)$ pour k fixé. □

5.4.3 Les étiquettes du schéma

On réutilise dans le codage les mêmes notions d'apex, de codage de branche, de position et de rang que la section 5.2.3. La seule différence se situant dans le fait que l'on considère les k-ascendances des sommets au lieu de considérer leurs k-complexes. Les notions sont en effet les mêmes car tout comme les sommets d'un k-complexe, les sommets d'une k-ascendance sont dans une même branche dans la décomposition considérée.

On considère T un arbre enraciné de n sommets et B sa décomposition k-ascendante. Soit u un sommet de T, A_u sa k-ascendance, a_u l'apex de A_u, et B_{a_u} la part de B contenant a_u. L'étiquette de u est définie comme étant le quadruplet suivant :

$$\ell(u) = (\text{chemin}(B_{a_u}), \text{rang}(a_u), d_{a_u}, P_u)$$

où :
- d_{a_u} est la distance de T de u à a_u ; et
- $P_u = \{\text{pos}(v) \mid v \in A_u, v \neq a_u\}$.

Afin d'optimiser la complexité du temps de réponse des requêtes, on suppose que P_u est stocké sous forme d'un tableau dont les éléments sont ordonnés selon l'ordre croissant des distances à u dans T des sommets correspondants. Concrètement, l'étiquette u de l'exemple de la figure 35 est [3] :

$$\ell(u) = (\text{"010"}, 0, 2, \{(1,5), (2,0), (1,3), (0,9)\}) \ .$$

Lemme 19. *Les étiquettes ont* $\log n + O(k \log(k \log(n/k)))$ *bits. Pour* $k = 1$*, leurs longueurs sont de* $\log n + 2 \log\log n + O(1)$.

Démonstration. On considère un sommet u avec h la profondeur de sa part dans B_{a_u}. Le mot binaire chemin(B_{a_u}) est de longueur égale à h. Soit $M = \lfloor (k+1) \log(2n/(k+1)) \rfloor$. On a $\text{rang}(a_u) \in [0, M)$, et donc $\log M + O(1)$ bits suffisent car d'après le lemme 18, les parts de B ont au plus M sommets. On a $d_{a_u} \in [0, k]$, et donc $O(\log k)$ bits suffisent. Chaque position $(h', r') \in P_u$ peut être stockée avec $\log h + \log M + O(1)$ bits puisque $h' \leqslant h$ et $r' < M$. De plus, $|P_u| = k$, donc $k \cdot (\log h + \log M) + O(k)$ bits suffisent pour P_u.

Au final, pour un h donné, la longueur de $\ell(u)$ est d'au plus :

$$|\ell(u)| \ \leqslant \ h + \log M + k \cdot (\log h + \log M) + O(k) \ .$$

3. la position est codée sous forme du couple (profondeur,rang) et le rang des sommets est ordonné par lignes de gauche à droite puis de bas en haut

On va maintenant majorer la longueur de chaque terme par une fonction dépendant uniquement des paramètres du problème (ici n et k) et non de valeurs spécifiques au sommet, comme h. Ceci va nous permettre découper le mot binaire afin d'extraire ses différents champs sans ajouter d'information supplémentaire.

D'après la preuve du lemme 14 un arbre de bidécomposition d'une $(k, k+1)$-triangulation a une profondeur au plus $k\log(n/k+1)$. Par conséquent B a une profondeur au plus $h_0 \leqslant \log(n/(k+1)) < \log(n/k)$. En majorant h par h_0, on obtient :

$$\begin{aligned}
|\ell(u)| &\leqslant h_0 + \log M + k\cdot(\log h_0 + \log M) + O(k) && (4)\\
&\leqslant h_0 + k\log h_0 + (k+1)\log M + O(k) && (5)\\
&\leqslant \log(n/k) + k\log\log(n/k) + (k+1)\log((k+1)\log(2n/(k+1))) + O(k) && (6)\\
&\leqslant \log(n/k) + (2k+1)\log\log(n/k) + O(k\log k) && (7)\\
&\leqslant \log(n/k) + O(k)\cdot(\log\log(n/k) + \log k) && (8)\\
&\leqslant \log n + O(k\log(k\log(n/k)))\ . && (9)
\end{aligned}$$

Pour $k=1$, cette formule donne $\log n + 3\log\log n + O(1)$ d'après l'équation (7). On peut légèrement améliorer cette analyse en observant que les deux premiers champs de $\ell(u)$: $\text{chemin}(B_{a_u})$ et $\text{rang}(a_u)$ peuvent être encodés $\log n + O(1)$ au lieu de $h_0 + \log M \sim \log(n/k) + \log\log(n/k)$.

Pour $k=1$, la k-augmentation G de T est T lui-même. Par conséquent, la taille de son demi-séparateur de 1 plutôt que 2. En effet, il est connu que pour chaque forêt contient un sommet dont la suppression la découpe en composantes connexes de taille moitié moins grande que le graphe de départ. Il en suit que les parts à profondeur h contiennent au plus $\alpha = \log n - h + O(1)$ sommets au lieu de $(k+1)(\log(2n/(k+1)) - h) = 2(\log n - h)$. Une conséquence directe est que les deux premiers champs de $\ell(u)$ peuvent être encodés conjointement en un seul mot W de $\log n + O(1)$ bits :

$$W \ = \ \text{chemin}(B_{a_u}) \circ \text{code}_0(\text{rang}(a_u))$$

Au final, on obtient que $|W| = h + 1 + \alpha = \log n + O(1)$. Les deux champs peuvent être extraits en temps constant puisque $\text{rang}(a_u)$ est codé grâce à un code suffixe. Les deux autres champs restants ont une longueur bornée par $2k\log\log(n/k) + O(k\log k)$. On obtient donc bien une longueur totale de $\log n + O(k\log(k\log(n/k)))$ bits pour les étiquettes. Pour le cas $k=1$, on peut même affiner ce majorant de manière à avoir des étiquettes d'au plus $\log n + 2\log\log n + 2$ pour tout $n \geqslant 16$. □

5.4.4 Test relationnel

Soit u, v deux sommets de T ayant pour k-ascendances respectives A_u et A_v. On considère une paire (k_1, k_2) d'entiers inférieurs ou égaux à k. On rappelle que vérifier si u et v sont (k_1, k_2)-reliés revient à vérifier si leur plus petit ancêtre commun w est à distance k_1 de u et à distance k_2 de v. Si u et v sont (k_1, k_2)-reliés alors w est contenu

dans $A_u \cap A_v$ puisque w est dans ce cas un ancêtre de u et v à distance au plus k des deux sommets.

Il est clair qu'il a exactement un seul ancêtre $w_u \in A_u$ de u à distance k_1 de u, et un seul ancêtre $w_v \in A_v$ de v à distance k_2 de v. Il suffit donc de tester si $w_u = w_v$, et de s'assurer que c'est bien le plus petit ancêtre commun en testant la présence d'un éventuel ancêtre commun $w' \in A_u \cap A_v$ à distance $k_1 - 1$ de u et $k_2 - 1$ de v.

On considère maintenant les étiquettes de u et v : $\ell(u) = (\text{chemin}(B_{a_u}), \text{rang}(a_u), d_{a_u}, P_u)$ et $\ell(v) = (\text{chemin}(B_{a_v}), \text{rang}(a_v), d_{a_v}, P_v)$. On note $\text{chemin}(X)[0..h]$ le préfixe de longueur h [4] de $\text{chemin}(X)$.

Le lemme suivant indique comment tester si $z \in A_u \cap A_v$ à partir de la position de z et des étiquettes de u et v. On rappelle que les positions ne sont des identifiants que pour les sommets d'une même branche. Deux sommets de deux branches différentes peuvent en effet avoir des positions identiques. On ne peut donc pas seulement utiliser que les positions des sommets pour le test.

Propriété 6. *Soit $z \in A_u$, et $\text{pos}(z) = (h, r)$. Alors $z \in A_v$ si et seulement si $\text{pos}(z) \in \text{pos}(A_v)$ et $\text{chemin}(B_{a_u})[0..h] = \text{chemin}(B_{a_v})[0..h]$.*

Démonstration. Soit $z \in A_u$, et soit B_z sa part dans la décomposition k-ascendante B. On sait que $\text{chemin}(B_z)$ est un préfixe de $\text{chemin}(B_{a_u})$ puisque $z \in A_u$. Si la profondeur de z est h, alors $\text{chemin}(B_z) = \text{chemin}(B_{a_u})[0..h]$. De même, $z \in A_v$ implique que $\text{chemin}(B_z) = \text{chemin}(B_{a_v})[0..h]$. Évidemment, $z \in A_v$ implique que $\text{pos}(z) \in \text{pos}(A_v)$. Par conséquent, on a montré que $z \in A_v$ implique que $\text{pos}(z) \in \text{pos}(A_v)$ et $\text{chemin}(B_{a_u})[0..h] = \text{chemin}(B_{a_v})[0..h] = \text{chemin}(B_z)$.

Réciproquement, si $\text{pos}(z) = (h, r) \in \text{pos}(A_v)$ et $\text{chemin}(B_{a_u})[0..h] = \text{chemin}(B_{a_v})[0..h]$ alors la part de z, B_z a pour identifiant le mot $\text{chemin}(B_z) = \text{chemin}(B_{a_u})[0..h] = \text{chemin}(B_{a_v})[0..h]$ puisque $z \in A_u$. Il en suit que la position (h, r) correspond à un sommet de B_z qui est dans A_v. Or, dans B_z, il n'existe qu'un unique sommet dont le rang est r : le sommet z. Au final, on obtient bien que $z \in A_v$. □

Pour chaque distance $i \in \{0, \ldots, k\}$, on note $\text{pos}(A_u)[i]$ la position du i-ème ancêtre de u, c'est-à-dire l'ancêtre à distance i de u. La position de cet ancêtre $\text{pos}(A_u)[i]$ peut être extrait de $\ell(u)$ en temps constant en utilisant la procédure suivante (on rappelle qu'on suppose que les éléments de P_u sont ordonnés suivant leurs distance à u) :

EXTRACT $\text{pos}(A_u)[i]$ (étant donnée $\ell(u)$) :
1. Si $i = d_{a_u}$ alors retourne $\text{pos}(a_u)$, i.e., $(|\text{chemin}(B_{a_u})|, \text{rang}(a_u))$.
2. Si $i > d_{a_u}$ alors $i = i - 1$.
3. Retourne $P_u[i]$.

D'après le lemme 6, afin de vérifier si $z \in A_u \cap A_v$ il est nécessaire de vérifier que $\text{pos}(z) = (h, r) \in \text{pos}(A_u) \cap \text{pos}(A_v)$ et que les préfixes de longueurs h des deux mots correspondent. On utilise donc la procédure suivante :

4. De tels préfixes peuvent être extrait en temps constant grâce à des décalage de mots dans le modèle RAM puisque $\text{chemin}(X)$ est un mot binaire de longueur $\leqslant \log n$.

Test Final (Test pour déterminer si u et v sont (k_1, k_2)-reliés étant donné $\ell(u)$ et $\ell(v)$) :

1. Extraire $(h_u, r_u) = \text{pos}(A_u)[k_1]$ et $(h_v, r_v) = \text{pos}(A_v)[k_2]$.
2. Si $(h_u, r_u) \neq (h_v, r_v)$ alors retourne Faux.
3. Si $\text{chemin}(B_{a_u})[0..h_u] \neq \text{chemin}(B_{a_v})[0..h_v]$alors retourne Faux.
4. SI $k_1 = 0$ or $k_2 = 0$ alors retourne Vrai.
5. Extraire $(h'_u, r'_u) = \text{pos}(A_u)[k_1 - 1]$ et $(h'_v, r'_v) = \text{pos}(A_v)[k_2 - 1]$.
6. Si $(h'_u, r'_u) \neq (h'_v, r'_v)$ alors retourne Vrai.
7. Si $\text{chemin}(B_{a_u})[0..h'_u] \neq \text{chemin}(B_{a_v})[0..h'_v]$ alors retourne Vrai.
8. Retourne Faux.

Lemme 20. *Le test k-relationnel peut s'effectuer en temps constant dans le modèle RAM.*

Démonstration. La procédure ci-dessus s'effectue en temps constant, et sa validité est déduite de la propriété 6. □

En combinant les lemmes 13, 19, 20, on prouve le théorème 18.

5.5 Conclusion

Dans la première partie de ce chapitre, on a décrit une représentation implicite pour les graphes planaires, et plus généralement pour les graphes excluant un mineur fixé, utilisant asymptotiquement $2 \log n$ bits par sommet. Une amélioration significative de ce schéma demanderait une approche totalement différente puisque une partition des arêtes en au moins deux parts mène automatiquement à des étiquettes d'au moins $2 \log n$ bits.

D'après [75], le nombre de graphes étiquetés de n sommets n'ayant pas le graphe K comme mineur est égal à $\psi(n, H) = n! \cdot 2^{hn+o(n)}$, où h dépends seulement de H. Par conséquent, le minimum théorique d'information à stocker par sommet pour l'adjacence est de $\frac{1}{n} \log \psi(n, H) \sim \log n + h$ d'après [58]. Cela conduit naturellement à poser la conjecture suivante :

Conjecture 5. *La famille des graphes de n sommets n'ayant pas le graphe K comme mineur admet un schéma d'adjacence avec des étiquettes de* $\log n + h$ *bits avec* $h = h(H)$.

On pense naturellement que cette conjecture est vraie et la résoudre même pour les cas les plus simples comme par exemple $K = K_3$ (la famille des forêts) serait une avancée considérable pour le domaine.

Dans la deuxième partie de ce chapitre, on a construit un schéma k-relationnel avec des étiquettes de $\log n + O(k \log(k \log(n/k)))$ bits. Ce schéma implique que les distances dans les arbres peuvent être calculées avec des étiquettes de $\log n + o(\log n)$ pour des sommets à distance au plus $o(\log n / \log \log n)$. Cela laisse ouverte la question de savoir s'il est possible de :

– Construire un schéma de distance pour les arbres avec des étiquettes de $\log n + o(\log n)$ bits pour des sommets à distance au plus $\log n$.

– Construire un schéma de distance pour les graphes de largeur arborescente bornée avec des étiquettes de $\log n + o(\log n)$ bits pour des sommets à distance au plus $\log n$.

Conclusion générale

Le travail effectué lors de cette thèse porte essentiellement sur deux domaines : les partitions d'arêtes des graphes sur surface et la représentation implicite de graphes.

Notre principale contribution pour les partitions d'arêtes est la partition des arêtes des graphes de genre d'Euler g en trois forêts plus un ensemble d'au plus $3g - 3$ arêtes. La preuve de l'existence d'une telle partition repose sur une généralisation d'un résultat de Kundu [68] sur la densité arborescente des graphes toriques. Pour la preuve, on utilise donc une technique similaire.

Pour le cas des graphes toriques, on a donné un algorithme linéaire réalisant cette partition. L'algorithme se base essentiellement sur deux algorithmes existants. Le premier algorithme utilisé est celui de [36] pour obtenir un cycle non-contractible non-séparant C dans G. Ceci permet ensuite de partitionner le graphe torique en un graphe planaire et un tambourin. Ensuite on utilise l'algorithme de Schnyder [86] qui permet de calculer un réaliseur de Schnyder du graphe planaire obtenu. Toute la complexité de notre algorithme repose sur une planarisation spéciale des graphes toriques.

Pour le cas des graphes universels induits, notre principale contribution est l'amélioration du graphe universel induit pour la famille des graphes de degré borné. Pour le cas où k est pair, le graphe obtenu est quasiment optimal. L'un des principaux avantages de notre construction est que le graphe est simple et mène donc directement à une représentation implicite efficace.

Un autre résultat important, même si la preuve reste assez simple, est résultat donnant le lien entre graphes universels induits et graphes universels induits orientés grâce aux graphes universels pour l'homomorphisme. En effet, cette construction permet de faire le lien entre le cas non-orienté et orienté pour de nombreuses familles de graphes. En fait, il permet de profiter de nombreux travaux effectués sur la coloration orientée.

On a aussi décrit un schéma d'adjacence pour les arbres de degré interne borné par $\hat{d}$ ayant n sommets avec des étiquettes de $\log n + O(\log \hat{d})$ bits. Plus que le résultat obtenu, c'est la technique utilisée de parcours bondissant qui est intéressante. En effet, elle est relativement générale et peut sans doute être utilisée sur d'autres familles de graphes.

L'un des principaux problèmes qu'on a résolu pour la représentation implicite de graphes était de savoir s'il était possible d'améliorer significativement le schéma d'adjacence en

$3 \log n + O(\log^* n)$ bits des graphes planaires. On a montré qu'il était possible de réaliser un schéma en $(2 + o(1)) \log n$ bits en utilisant un schéma en $(1 + o(1)) \log n$ les graphes de largeur arborescente bornée. La technique de codage est générique. En effet, elle repose essentiellement sur la bidécomposition et sur l'existence de séparateurs du graphe. Il est donc possible d'utiliser cette technique pour d'autres schémas et d'ailleurs, on a amélioré le meilleur schéma k-relationnel connu pour les arbres grâce à cette technique.

Perspectives

Les suites possibles que l'on peut donner à ces recherches sont diverses. Pour les partitions des arêtes, on a déjà décrit dans la partie 2.5 de nombreux problèmes ouverts. Fondamentalement, les conjectures proposées sont des généralisations des résultats connus pour les graphes planaires. La partition des graphes de genre en deux graphes g-externes semble particulièrement intéressante car elle est une généralisation assez naturelle de la partition des graphes planaires en deux graphes planaires extérieurs.

Un autre axe de recherche possible est la généralisation des réaliseurs de Schnyder pour les graphes de genre g. En effet, les réaliseurs de Schnyder ont de nombreuses applications pour les graphes planaires notamment en termes de dessins de graphes, d'énumération et d'algorithmes de partition. Trouver une notion similaire pour les graphes sur surfaces aurait donc de nombreuses applications.

Pour le cas des représentations implicites, on a décrit dans les conclusions des trois chapitres correspondants de nombreux problèmes ouverts liés aux résultats que l'on a obtenus. On peut résumer ces problèmes en la question suivante :

Est-il possible d'atteindre la borne déduite de l'énumération de la famille à $O(1)$ bits près ?

Ce problème est la plupart du temps très difficile. En effet, même pour une famille aussi simple que celle des arbres, on ne connaît la réponse à ce problème. On a néanmoins prouvé que la réponse à cette question est positive pour les arbres de degré interne borné ainsi que les graphes de degré pair. Ce problème revient grossièrement à se poser la question de savoir s'il est possible de distribuer la représentation du graphe parmi ses sommets sans duplication d'information. En effet, on peut imaginer que dans certains cas l'information du graphe ne peut pas être répartie équitablement entre ses sommets et que l'on soit obliger de coder une même information dans plusieurs sommets afin de pouvoir réaliser le test d'adjacence. On conjecture néanmoins que pour les familles que l'on considère (arbres, graphes de genre borné, ...), une répartition quasi-optimale de la représentation du graphe sur ses sommets est possible.

Les différentes représentations distribuées de graphes que l'on a décrites ne permettent pas de modifier de manière efficace l'information à stocker en cas de modification du graphe de départ. En cas de modification du graphe, le seul moyen d'obtenir la nouvelle représentation est de réappliquer l'étiquetage sur tout le graphe. On peut se poser la question de savoir s'il est possible de traiter de manière efficace le modèle dynamique

pour les familles de graphes que l'on a considéré. Cette amélioration semble essentielle car de nombreux réseaux distribués tel que les réseaux pair-à-pair sont dynamiques. La principale difficulté concernant le passage au cadre dynamique de notre schéma pour les graphes de largeur arborescente k semble être le maintien de la bidécomposition. Un tel maintien dynamique efficace de la bidécomposition semble être possible en s'inspirant de la méthode décrite dans [67]. Dans ce papier, les auteurs décrivent un schéma de routage dynamique dans les arbres utilisant des étiquettes de taille optimale et dont la complexité amortie par ajout et suppression de sommet est de $O(\log^3 n)$ messages. Leur schéma se base sur une décomposition en séparateur de l'arbre qui est maintenu dynamiquement. Cette décomposition en séparateurs étant très proche de notre notion de bidécomposition, il semble possible d'adapter leur méthode pour notre schéma pour les graphes de largeur arborescente k afin de maintenir dynamiquement la bidécomposition avec une complexité amortie par ajout et suppression de sommet faible.

Bibliographie

[1] S. Abiteboul, S. Alstrup, H. Kaplan, T. Milo, and T. Rauhe. Compact labeling schemes for ancestor queries. *SIAM J. on Computing*, 35(6) :1295, 2006.

[2] S. Abiteboul, H. Kaplan, and T. Milo. Compact labeling schemes for ancestor queries. In 12^{th} *Symposium on Discrete Algorithms (SODA)*, pages 547–556. ACM-SIAM, Jan. 2001.

[3] M. O. Albertson and J. Hutchinson. On the independence ratio of a graph. *J. Graph Theory*, 2 :1–8, 1978.

[4] N. Alon and M. Capalbo. Sparse universal graphs for bounded degree graphs. *Random Structures and Algorithms*, 31 :123–133, 2007.

[5] N. Alon, B. Mohar, and D. P. Sanders. On acyclic colorings of graphs on surfaces. *Israel J. Math.*, 1996.

[6] S. Alstrup, P. Bille, and T. Rauhe. Labeling schemes for small distances in trees. In 14^{th} *Symposium on Discrete Algorithms (SODA)*, pages 689–698. ACM-SIAM, Jan. 2003.

[7] S. Alstrup, P. Bille, and T. Rauhe. Labeling schemes for small distances in trees. *SIAM Journal on Discrete Mathematics*, 19(2) :448–462, 2005.

[8] S. Alstrup, C. C. Gavoille, H. Kaplan, and T. Rauhe. Nearest common ancestors : A survey and a new algorithm for a distributed environment. *Theory of Computing Systems*, 37 :441–456, 2004.

[9] S. Alstrup and T. Rauhe. Small induced-universal graphs and compact implicit graph representations. In 43^{rd} *Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 53–62. IEEE Computer Society Press, Nov. 2002.

[10] S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM J. on Algeb. and Disc. Meth.*, 8 :277–284, 1987.

[11] J. Battle, F. Harary, Y. Kodama, and J. W. T. Youngs. additivity of the genus of a graph. *Bull. Amer. Math. Soc.*, 68 :565–568, 1962.

[12] F. Bazzaro and C. Gavoille. Localized and compact data-structure for comparability graphs. In 16^{th} *Annual International Symposium on Algorithms and Computation (ISAAC)*, volume 3827 of Lecture Notes in Computer Science, pages 1122–1131. Springer, Dec. 2005.

[13] L. W. Beineke and R. E. Pippert. The number of labeled k-dimensional trees. *Journal of Combinatorial Theory*, 6 :200–205, 1969.

[14] S. Bhatt, F. Chung, F. Leighton, and A. Rosenberg. Optimal simulations of tree machines. In *27th IEEE Foundations of Computer Science*, pages 274–282, Toronto, 1986.

[15] S. N. Bhatt, F. R. K. Chung, F. T. Leighton, and A. L. Rosenberg. Universal graphs for bounded-degree trees and planar graphs. *SIAM Journal on Discrete Mathematics*, 2(2) :145–155, May 1989.

[16] S. N. Bhatt, F. R. K. Chung, F. T. Leighton, and A. L. Rosenberg. Universal graphs for bounded-degree trees and planar graphs. *SIAM J. Discret. Math.*, 2(2) :145–155, 1989.

[17] S. N. Bhatt and C. E. Leiserson. *How to assemble tree machines*. F. Preparata, 1984.

[18] H. L. Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. *SIAM J. on Computing*, 25(6) :1305–1317, 1996.

[19] N. Bonichon, C. Gavoille, and A. Labourel. Edge partition of toroidal graphs into forests in linear time. In Elsevier, editor, 7^{th} *International Conference on Graph Theory (ICGT)*, volume 22, pages 421–425. Electronic Notes in Discrete Mathematics, Sept. 2005.

[20] N. Bonichon, C. Gavoille, and A. Labourel. Short labels by traversal and jumping. In 13^{th} *Int'l Colloquium on Structural Information & Communication Complexity (SIROCCO)*, volume 4056 of LNCS, pages 143–156. Springer, July 2006.

[21] N. Bonichon, C. Gavoille, and A. Labourel. Short labels by traversal and jumping. In *SIROCCO*, pages 143–156, 2006.

[22] O. V. Borodin, A. V. Kostochka, J. N. setřil, A. Raspaud, and E. Sopena. On universal graphs for planar oriented graphs of a given girth. *Discrete Math.*, 188 :73–85, 1998.

[23] M. A. Breuer. Coding the vertexes of a graph. *IEEE Transactions on Information Theory*, IT-12 :148–153, 1966.

[24] S. Butler. Induced-universal graphs for graphs with bounded maximum degree. preprint, 2006.

[25] M. R. Capalbo. Small universal graphs for bounded-degree planar graphs. *Combinatorica*, 22(3) :345–359, July 2002.

[26] G. Cherlin and P. Komjath. There is no universal countable pentagon-free graph. *Journal of Graph Theory*, 18(4) :337–341, 1994.

[27] F. R. K. Chung. Universal graphs and induced-universal graphs. *Journal of Graph Theory*, 14 :443–454, 1990.

[28] F. R. K. Chung and R. L. Graham. On universal graphs for spanning trees. *J. London Math. Soc.*, 27 :203–211, 1983.

[29] C. Cortés, C. Grima, A. Marquez, and A. Nakamoto. Diagonal flips in outer-triangulations on closed surfaces. *Discrete Mathematics*, 254 :63–74, 2002.

[30] B. Courcelle and A. Twigg. Compact forbidden-set routing. In *Proceedings STACS*, volume 4393 of *Lec. Notes. Comput. Sci.*, pages 37–48, 2006.

[31] B. Courcelle and R. Vanicat. Query efficient implementation of graphs of bounded clique-width. *Discrete Appl. Math.*, 131(1) :129–150, 2003.

[32] S. P. Cyril Gavoille, David Peleg and R. Raz. Distance labeling in graphs. *Journal of Algorithms*, 53(1) :85–112, 2004.

[33] C. L. D. Archdeacon, N. Hartsfield and B. Mohar. Obstruction sets for outer-projective-planar graphs. *Ars Combin.*, 49 :113–127, 1998.

[34] M. DeVos, G. Ding, B. Oporowski, D. P. Sanders, B. Reed, P. Seymour, and D. Vertigan. Excluding any graph as a minor allows a low tree-width 2-coloring. *J. Comb. Theory Ser. B*, 91(1) :25–41, 2004.

[35] G. Ding, B. Oporowski, D. P. Sanders, and D. Vertigan. Surfaces, tree-width, clique-minors, and partitions. *J. Comb. Theory Ser. B*, 79(2) :221–246, 2000.

[36] H. Djidjev and S. Venkatesan. Planarization of graphs embedded on surfaces. In *Proc. 21st Workshop on Graph-Theoretic Concepts in Computer Science (WG '95)*, volume 1017, pages 62–72, 1995.

[37] F. F. Dragan and I. Lomonosov. New routing schemes for interval graphs, circular-arc graphs, and permutation graphs. In 14^{th} *IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS)*, pages 78–83, Nov. 2002.

[38] F. F. Dragan and I. Lomonosov. On compact and efficient routing in certain graph classes. In 15^{th} *Annual International Symposium on Algorithms and Computation (ISAAC)*, volume 3341 of Lecture Notes in Computer Science, pages 402–414. Springer, Dec. 2004.

[39] O. Egecioglu and L.-P. Shen. A bijective proof for the number of labeled q-trees. *Ars Combinatoria*, 25B :3–30, 1988.

[40] E. S. Elmallah and C. J. Colbourn. Partitioning the edges of a planar graph into two partial k-trees. *Congressus Numerantium*, 66 :69–80, 1988.

[41] P. Erdös, L. Gerencsér, and A. Máté. Problems of graph theory concerning optimal design. In P. Erdös, A. Rényi, and V. T. Sós, editors, *Colloq. Math. Soc. Janos Bolyai* 4 *: Combinatorial theory and its applications*, volume 1, pages 317–325. North-Holland, 1970.

[42] L. Esperet, A. Labourel, and P. Ochem. On induced-universal graphs for the class of bounded-degree graphs. Technical Report RR-1424-07, LaBRI, Université Bordeaux 1, march 2007.

[43] R. L. G. F. R. K. Chung and J. Shearer. Universal caterpillars. *J. Comb. Theory, Ser. B*, 31, 1981.

[44] P. Fraigniaud and C. Gavoille. Routing in trees. In F. Orejas, P. G. Spirakis, and J. v. Leeuwen, editors, 28^{th} *International Colloquium on Automata, Languages and Programming (ICALP)*, volume 2076 of Lecture Notes in Computer Science, pages 757–772. Springer, July 2001.

[45] C. Gavoille, M. Katz, N. A. Katz, C. Paul, and D. Peleg. Approximate distance labeling schemes. In *Algorithms â ESA 2001*, volume 2161 of *Lecture Notes in Computer Science*. Springer, 2001.

[46] C. Gavoille and A. Labourel. Brief annoncement : On local representation of distances in trees. In *PODC*, pages 352–353, 2007.

[47] C. Gavoille and A. Labourel. Distributed relationship schemes for trees. In *ISAAC*, 2007.

[48] C. Gavoille and A. Labourel. Shorter implicit representation for planar graphs and bounded treewidth graphs. In *ESA*, 2007.

[49] C. Gavoille and C. Paul. Optimal distance labeling schemes for interval and circular-arc graphs. In G. D. Battista and U. Zwick, editors, 11^{th} *Annual European Symposium on Algorithms (ESA)*, volume 2832 of Lecture Notes in Computer Science, pages 254–265. Springer, Sept. 2003.

[50] C. Gavoille and D. Peleg. Compact and localized distributed data structures. *Journal of Distributed Computing*, 16 :111–120, May 2003. PODC 20-Year Special Issue.

[51] J. R. Gilbert, D. J. Rose, and A. Edenbrandt. A separator theorem for chordal graphs. *SIAM J. on Algebraic and Discrete Methods*, 5(3) :306–313, 1984.

[52] M. Goldstern and M. Kojman. Universal bridge-free graphs, 1994.

[53] D. Gonçalves. Covering planar graphs with 3 forests, one being of maximum degree 4. submited.

[54] D. Gonçalves. Edge partition of planar graphs into two outerplanar graphs. In 37^{th} *Annual ACM Symp. on Theory of Computing (STOC)*, pages 504–512. ACM Press, May 2005.

[55] D. Gonçalves. *Étude de différents problèmes de partition de graphes*. PhD thesis, Université Bordeaux 1, 2006.

[56] R. Halin. s-functions for graphs. *J. Geometry*, 8 :171–186, 1976.

[57] M. Jovan, J. Marincek, and B. Mohar. Embedding graphs in the torus in linear time. In *Proceedings of the 4th International IPCO Conference on Integer Programming and Combinatorial Optimization*, pages 360–363, London, UK, 1995. Springer-Verlag.

[58] S. Kannan, M. Naor, and S. Rudich. Implicit representation of graphs. In 20^{th} *Annual ACM Symposium on Theory of Computing (STOC)*, pages 334–343. ACM Press, May 1988.

[59] H. Kaplan and T. Milo. Short and simple labels for small distances and other functions. In 7^{th} *International Workshop on Algorithms and Data Structures (WADS)*, volume 2125 of Lecture Notes in Computer Science, pages 32–40. Springer, Aug. 2001.

[60] M. Katz, N. A. Katz, A. Korman, and D. Peleg. Labeling schemes for flow and connectivity. Technical Report MCS01-01, The Weizmann Institute of Science, Aug. 2001.

[61] K. S. Kedlaya. Outerplanar partitions of planar graphs. *Theory Ser. B*, 67(2) :238–248, 1996.

[62] P. Komjath, A. Mekler, and J. Pach. Some universal graphs. *Mathematics*, 64 :158–168, 1988.

[63] P. Komjath and J. Pach. Universal elements and the complexity of certain classes of infinite graphs. *Discrete Mathematics*, 95 :255–270, 1991.

[64] P. Komjath and J. Pach. The complexity of a class of infinite graphs. *Combinatorica*, 14(1) :121–125, 1994.

[65] A. Korman. Labeling schemes for vertex connectivity. In 34^{th} *International Colloquium on Automata, Languages and Programming (ICALP)*, volume 4596 of Lecture Notes in Computer Science, pages 102–109. Springer, July 2007.

[66] A. Korman, S. Kutten, and D. Peleg. Proof labeling system. In 24^{th} *Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 9–18. ACM Press, July 2005.

[67] A. Korman and D. Peleg. Compact separator decompositions and routing in dynamics trees. In 34^{th} *Int'l Colloquium on Automata, Languages and Programming (ICALP)*, volume 4596 of LNCS. Springer, July 2007.

[68] S. Kundu. Bounds on number of disjoint spanning trees. *J. Combinatorial Theory B*, 17 :199–203, 1974.

[69] R. Libeskind-Hadas, D. Mazzoni, and R. Rajagopalan. Tree-based multicasting in wormhole-routed irregular topologies. In *Proc. Merged 12th Internat. Parallel Processing Symp. and the 9th Symp. on Parallel and Distributed Processing.*, pages 244–249, April 1998.

[70] B. Mohar and C. Thomassen. *Graphs on Surfaces*. Johns Hopkins University Press, 2001.

[71] J. H. Muller. *Local structure in graphs*. PhD thesis, School of information and computer science, Georgia institute of technology Atlanta, 1988.

[72] C. Nash-Williams. Decomposition of finite graphs into forests. *J. London Math. Soc*, 39, 1964.

[73] C. S. J. A. Nash-Williams. Edge-disjoint spanning trees of finite graphs. *J. London Math. Soc.*, 36 :445–450, 1961.

[74] L. M. Ni and P. K. McKinley. A survey of wormhole routing techniques in direct networks. In *IEEE Computer*, volume 2, pages 62–76, 1993.

[75] S. Norine, N. Robertson, R. Thomas, and P. Wollan. Proper minor-closed families are small. *J. of Combinatorial Theory, Series B*, 96(5) :754–757, 2006.

[76] R. Pagh. Low redundancy in static dictionaries with constant query time. *SIAM J. on Computing*, 31(2) :353–363, 2001.

[77] J. Petersen. Die theorie der regulären graphs. *Acta Math.*, 15 :193–220, 1891.

[78] R. Rado. Universal graphs and universal functions. *Acta Arithmetica*, 9 :331–340, 1964.

[79] R. Rado. Universal graphs. In *A Seminar in Graph Theory*, pages 83–85, 1967.

[80] A. Raspaud and E. Sopena. Good and semi-strong colorings of oriented planar graphs. *Inform. Processing Letters*, 51, 1994.

[81] B. Reed. Finding approximate separators and computing treewidth quickly. In 24^{th} *Annual ACM Symp. on Theory of Comp. (STOC)*, pages 221–228. ACM Press, 1992.

[82] G. Ringel. *Map Color Theorem*. Springer, New York, 1974.

[83] N. Robertson and P. Seymour. Graph minors. ii. algorithmic aspect of treewidth. *J. Algorithm*, 7 :309–322, 1986.

[84] J. Roskind and R. E. Tarjan. A note on finding minimum-cost edge-disjoint spanning trees. *Math. Oper. Res.*, 10(4) :701–708, 1985.

[85] W. Schnyder. Planar graphs and poset dimension. *Order*, pages 323–343, 1989.

[86] W. Schnyder. Embedding planar graphs on the grid. In 1^{st} *Symp. on Discrete Algorithms (SODA)*, pages 138–148. ACM-SIAM, Jan. 1990.

[87] E. Sopena. The chromatic number of oriented graphs. *J. Graph Theory*, 25 :191–205, 1997.

[88] J. P. Spinrad. *Efficient Graph Representations*. American Mathematical Society, 2003.

[89] S. Stahl and L. W. Beineke. Blocks and the nonorientable genus of a graph. *J. Graph Theory*, 1 :75–78, 1977.

[90] R. Thomas and X. Yu. Five-connected toroidal graphs are hamiltonian. *J. Comb. Theory Ser. B*, 69(1) :79–96, 1997.

[91] C. Thomassen. The jordan-schönflies theorem and the classification of surfaces. *Am. Math. Monthly*, 99(2) :116–130, 1992.

[92] M. Thorup and U. Zwick. Compact routing schemes. In 13^{th} *Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 1–10. ACM Press, July 2001.

[93] W. T. Tutte. On the problem of decomposing a graph into n connected factors. *J. London Math. Soc.*, 36 :221–230, 1961.

[94] J. van Leeuwen. *Graph Algorithms*. Handbook of Theoretical Computer Science. A : Algorithm and Complexity Theory. Elsevier, North Holland, 1990.

[95] A. Y. Wu. Embedding of tree networks into hypercubes. *Journal of Parallel and Distributed Computing*, 1985.