Aspects algorithmiques et combinatoires des réaliseurs des graphes plans maximaux

Nicolas Bonichon

N° d'ordre : 2627

THÈSE

PRÉSENTÉE À

L'UNIVERSITÉ BORDEAUX I

ÉCOLE DOCTORALE DE MATHÉMATIQUES ET D'INFORMATIQUE

Par **Nicolas BONICHON**

POUR OBTENIR LE GRADE DE

DOCTEUR

SPÉCIALITÉ : INFORMATIQUE

Aspects algorithmiques et combinatoires des réaliseurs des graphes plans maximaux

---

Soutenue le : 19 décembre 2002

Après avis des rapporteurs :

Jean-Marc Fédou ..	Professeur
Hubert de Fraysseix	Directeur de Recherche
Michel Habib	Professeur

Devant la commission d'examen composée de :

André Raspaud	Professeur	Président
Jean-Marc Fédou ..	Professeur	Rapporteur
Hubert de Fraysseix	Directeur de Recherche	Rapporteur
Michel Habib	Professeur	Rapporteur
Bertrand Le Saëc ..	Professeur	Examinateur
Mohamed Mosbah .	Maitre de Conférence HDR	Examinateur
Cyril Gavoille	Professeur	Invité

– 2002 –

Remerciements

Mes remerciements s'adressent en premier à mes directeurs de thèse : Bertrand Le Saëc et Mohamed Mosbah. Ils ont fait preuve d'un soutien sans faille. Ils ont su remarquablement me guider tout au long de ces années. Leurs choix et leurs jugements se sont toujours révélés être très justes. Je leur dois beaucoup de choses. Qu'ils trouvent ici toute l'expression de ma reconnaissance.

Je tiens à remercier chaleureusement Hubert de Fraysseix, Jean-marc Fédou et Michel Habib pour m'avoir fait l'honneur de lire attentivement mon mémoire. Leurs commentaires sur ce mémoire ont été très enrichissants. Je tiens à exprimer ma gratitude pour l'intérêt qu'ils ont porté à l'égard de mon travail.

Je remercie André Raspaud d'avoir accepté de présider mon jury.

Je suis très sensible à la présence de Cyril Gavoille dans ce jury. Ce fut également un réel plaisir de travailler avec lui et Nicolas Hanusse. Je suis heureux de pouvoir leur exprimer ici toute ma gratitude.

Les discutions avec Mireille Bousquet-Mélou, Olivier Guibert, Xavier Viennot et surtout Philippe Duchon m'ont apporté énormément de réponses.

Je tiens également à remercier :

- Guillaume, Irek, Davy, Valère, Pascalou, Akka, Laurent pour leur bonne humeur, leur amitié et soutien lors de ces trois années. Ils ont toujours été disponibles aussi bien pour prendre un p'tit café que pour me donner un coup de main quand j'en avais besoin. C'est toujours avec grand plaisir que je les retrouvais tous les jours.

- Yon, Fabrice, Yvan pour parce que se sont vraiment des types biens. Avec eux les missions sont toujours des grands moments.

- Pierre Ramet, Olivier Guibert, Christophe Paul, Gwenola Thomas, Julien Bernet, Jean-Pierre Jardry, Isabelle Dutour et Jean-Michel Lepine pour m'avoir

accompagné lors de mes premiers pas à l'IUT.

- Philippe Biais pour sa gentillesse et son efficacité.

- Pierre-André Wacrenier pour ses discussions souvent intéressantes.

- Marianne Campagnolle, Benoît Eloseguy, Michel Alfaro et Catherine Jaulent pour m'avoir donné le goût de la recherche.

- Paul et Glenda Ferguson pour leur soutien linguistique.

- Philippe Lagouarde pour ses encouragements lors de mon séjour à ICSF.

- Toute l'équipe des chasseurs de fautes (Virginie, Tantine, Françoise, Dudu, Stéph, Nico Lapin, Boulet) qui ont permis d'améliorer la qualité de ce document.

- Choune pour m'avoir initié à la pratique du voile-contact.

- Finot, Niche, Francky et Titus pour avoir su nous guider dans ces canyons aux météos imprévisibles.

- Les petits gars de la boulet-liste pour les nombreuses discutions "philosophiques" qu'ils ont su entretenir.

- Franck et Michèle pour avoir su créer à Voeuil-et-Giget un climat propice à l'avancement de cette thèse.

- Ma famille et plus particulièrement mes parents, mon frère, ma soeur et Annie pour m'avoir soutenu et épaulé durant toutes mes études.

S'il y a bien une personne que je dois remercier c'est bien sûr Virginie. Sans son amour et son soutien sans faille, rien n'aurait été possible. Discrètement, elle sait m'amener jusqu'au bout du meilleur de moi-même.

à mon parrain.

Aspects algorithmiques et combinatoires des réaliseurs des graphes plans maximaux

Résumé : Les réaliseurs, ou arbres de Schnyder, ont été introduits par Walter Schnyder à la fin des années 80 pour caractériser les graphes planaires, puis pour dessiner ces mêmes graphes sur des grilles $(n-2) \times (n-2)$.

Dans ce document nous proposons dans un premier temps une extension du théorème de Wagner aux réaliseurs, qui nous permet d'établir une relation entre le nombre de feuilles et le nombre de faces tricolores d'un réaliseur.

Ensuite, à l'aide d'une bijection entre les réaliseurs et les paires de chemins de Dyck qui ne se coupent pas, nous énumérons les réaliseurs. Un algorithme de génération aléatoire de p chemins de Dyck ne se coupant pas, est également présenté. Il permet en outre de générer aléatoirement des réaliseurs en temps linéaire.

Puis nous montrons que grâce aux réaliseurs, il est possible de dessiner, à l'aide de lignes brisées des graphes planaires sur des grilles de largeur et de surface optimales.

Enfin, nous proposons une généralisation des réaliseurs minimaux aux graphes planaires connexes : les arbres recouvrants bien-ordonnés. Grâce à cette généralisation ainsi qu'à une méthode de triangulation adaptée nous proposons un algorithme de codage des graphes planaires à n sommets en $5,007n$ bits.

Mots clés : Réaliseurs, Arbres de Schnyder, Dessin de graphes, Théorème de Wagner, flip diagonal, Pastèque, génération aléatoire uniforme, énumération graphes planaires

Discipline : Informatique

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

Algorithmic and combinatorial aspects of maximal plane graphs realizers

Abstract : The realizers, or Schnyder trees, have introduced by Walter Schnyder in the late 80's to give a characterization of planar graphs and to draw them on $(n-2) \times (n-2)$ grids.

In this document, we first give an extension of Wagner's theorem to realizers. Using this theorem we establish a relationship between the number of leaves and the number of 3-colored faces of a realizer.

A bijection between realizers and pairs of non-crossing Dyck path give us an enumeration of realizers. An algorithm generating p non-crossing Dyck paths, is also proposed. It allows us to generate randomly realizers in linear time.

Then, we show that thanks to realizers, we can draw plane graphs with polylines on grids of optimal width and area.

Finally, we propose a generalization of minimal realizers to connected planar graphs : well-orderly spanning trees. Using this generalization and with a particular triangulation algorithm, we present a new $5.007n$ bit planar graph encoding.

Keywords : Realizers, Schnyder's trees, graph drawing, polyline drawings, Wagner's theorem, diagonal flip, watermelon, uniform random generation, planar graph enumeration

Discipline : Computer-Science

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

Table des matières

Table des figures

Introduction

De nombreux problèmes, à commencer par les réseaux de communication, sont modélisés par des graphes. Parmi les graphes, certains peuvent être dessinés sur un plan sans que les arêtes ne se croisent. Ces graphes sont appelés graphes planaires. Lorsque l'ordre des arêtes autour des sommets est fixé et que la face externe est également fixée, on parle de carte planaire ou de graphe plan.

Bien que ces graphes soient très étudiés, ils conservent encore de nombreux problèmes ouverts. Considérons deux de ces problèmes. Quelle est la surface nécessaire et suffisante pour dessiner n'importe quel graphe plan ? Quel est le nombre de bits nécessaires et suffisants pour coder un graphe planaire ?

Fary [43], Stein [88] et Wagner [94] ont montré de façon indépendante que tout graphe plan admet un dessin où les arêtes sont représentées par des lignes droites. Depuis le début des années 90 on sait qu'il est possible de dessiner en temps linéaire un graphe plan à l'aide de lignes droites sur une grille $(n-2) \times (n-2)$ [87]. On sait également qu'une grille $(\lfloor \frac{2(n-1)}{3} \rfloor) \times (\lfloor \frac{2(n-1)}{3} \rfloor)$ est nécessaire, mais on ignore si cette grille est suffisante pour dessiner tous les graphes plans à n sommets.

Concernant le codage des graphes planaires, Bender, Gao et Wormald [8] ont montré qu'il faut au moins $4,71n$ bits pour coder un graphe planaire à n sommets. Cette borne inférieure est donnée par l'énumération des graphes planaires étiquetés biconnexes. Plus récemment, Osthus, Prömel et Taraz [80] ont montré qu'il était possible de coder un graphe planaire à n sommets avec $5,22n$ bits.

Pour essayer de répondre à ces deux questions on s'intéresse aux graphes plans maximaux. Ces graphes sont maximaux dans le sens où si on leur ajoute une arête quelconque, ils ne sont plus planaires.

Le fait de considérer les graphes planaires maximaux dans le domaine du dessin de graphes est naturel : en effet, comme tout graphe planaire est un sous-graphe d'un graphe planaire maximal, savoir dessiner tous les graphes planaires maximaux, permet de dessiner tous les graphes planaires. Comme maximiser un graphe planaire peut se faire en temps linéaire [48, 12, 9], tout algorithme de dessin qui est linéaire sur les graphes planaires maximaux induit un algorithme de dessin linéaire sur les graphes planaires.

Dans la problématique du codage, l'utilisation des graphes planaires maximaux est différente. Pour coder un graphe planaire G, on code un graphe planaire maximal G' qui admet G comme sous-graphe. Puis on code les arêtes de G' qu'il faut supprimer

pour reconstruire le graphe G. La difficulté d'une telle approche est double : dans un premier temps, il faut maximiser le graphe G de manière à conserver le maximum d'informations sur G et dans un second temps il faut coder efficacement G' et les arêtes de G' superflues.

Pour étudier et manipuler les graphes planaires maximaux, un outil s'est avéré indispensable : le *réaliseur*. Cet objet mathématique, a été introduit par Schnyder [86] pour caractériser les graphes planaires en terme de dimension d'un ensemble partiellement ordonné.

Un réaliseur d'un graphe plan maximal G est une partition des arêtes internes de G en trois arbres "recouvrants" (T_0, T_1, T_2) telle qu'autour de chaque sommet interne de G on rencontre, dans le sens anti-trigonométrique : l'arête vers le parent dans T_0, des arêtes vers les enfants dans T_2 (s'ils existent), l'arête vers le parent dans T_2, les arêtes vers les enfants dans T_0 (s'ils existent), l'arête vers le parent dans T_1 et enfin les arêtes vers les enfants dans T_2 (s'ils existent). La figure 1 montre un exemple de réaliseur.

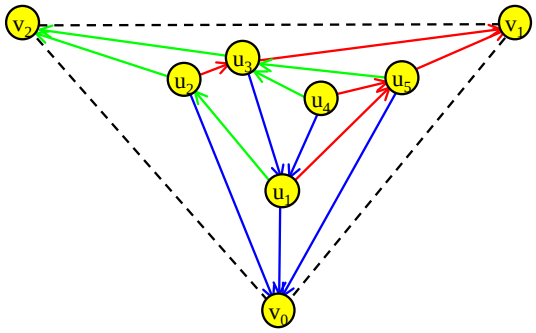

Figure 1 – Un exemple de réaliseur.

Avant d'aller plus loin dans les applications des réaliseurs, rappelons que tout graphe plan maximal admet au moins un réaliseur. De plus, on peut construire un tel réaliseur en temps linéaire [87].

Dans le domaine du dessin de graphes, cet objet a trouvé un grand essor. Citons 3 algorithmes de dessin s'appuyant sur les réaliseurs : dessin lignes droites [87], dessin de 2-visibilité [23], "floor-planning" [70]. Dans ce document nous présentons un nouvel algorithme de dessin de graphes qui utilise les réaliseurs fournissant un dessin avec des lignes brisées. Cet algorithme linéaire obtient des dessins de surface et de largeur optimales pour la classe des graphes planaires.

Fraysseix et Ossona de Mendez [48] ont établi une bijection entre les 3-orientations (i.e. les orientations d'un graphe plan maximal telles que tous les sommets internes ont un degré rentrant de 3) et les réaliseurs. Ils ont, par ailleurs, montré que l'ensemble des réaliseurs d'un graphe a une structure de treillis distributif. Bien plus tard, Chiang, Lin et Lu [23] ont montré que la somme des nœuds internes des trois arbres d'un réaliseur était strictement inférieure au nombre de nœuds du réaliseur : $\xi_0 + \xi_1 + \xi_2 \leq n - 1$ où ξ_i désigne le nombre de nœuds internes de l'arbre T_i. Ce dernier résultat permet

de réduire la taille des dessins obtenus par certains algorithmes utilisant les réaliseurs. Notre étude des réaliseurs nous a permis de relier le nombre de faces tricolores (i.e. faces qui possèdent une arête dans chacun des arbres) d'un réaliseur au nombre de nœuds internes des arbres du réaliseur. Pour montrer ce dernier résultat, nous avons été amené à proposer une extension du théorème de Wagner [94] aux réaliseurs. Le théorème de Wagner affirme qu'il est possible de transformer un graphe plan maximal en un autre graphe plan maximal à l'aide d'opérations appelées "flips diagonaux".

Deux généralisations des réaliseurs ont été également proposées. La première étend la notion de réaliseur aux graphes plans triconnexes. Dans de tels graphes, un réaliseur est encore constitué de 3 arbres recouvrants, où une arête peut appartenir à au plus 2 arbres [7, 44]. Les réaliseurs de graphes triconnexes sont utilisés dans des algorithmes de routage [93] ainsi que pour le calcul de dessins convexes (chaque face est un polygone convexe) [7, 44]. La seconde généralisation s'applique à certaines cartes planaires connexes. Il s'agit des arbres recouvrants ordonnés introduits par Chuang, Garg, He, Kao et Lu [26]. Grâce à cette généralisation, plusieurs algorithmes de codage furent proposés [26].

Le travail présenté ici s'inscrit dans la continuité des travaux précédemment cités. A savoir, dégager de nouvelles propriétés des réaliseurs, présenter un nouvel algorithme de dessin de graphes, mais aussi proposer un algorithme de codage des graphes planaires à l'aide des réaliseurs.

Plan du document

Partie I : Etude des réaliseurs

Chapitre 1 : Présentation des Réaliseurs

Dans un premier temps, nous rappelons quelques notions de théorie des graphes, des ensembles partiellement ordonnés.

Nous présentons également la définition de réaliseur ainsi que quelques propriétés fondamentales de ces objets que nous réutilisons tout au long de ce document. Nous présentons deux généralisations des réaliseurs : les réaliseurs de graphes triconnexes et les arbres recouvrants ordonnés.

Chapitre 2 : Extension du théorème de Wagner aux réaliseurs

Dans ce chapitre nous nous intéressons aux réaliseurs de taille n dans leur ensemble. Nous introduisons des opérations sur les réaliseurs appelés "flips diagonaux coloriés". Grâce à ces opérations nous proposons une extension du théorème de Wagner aux réaliseurs [17].

Ce résultat nous permet de trouver une relation entre le nombre de nœuds internes d'un réaliseur et le nombre de ses faces tricolores : $\xi_0 + \xi_1 + \xi_2 - \Delta = n - 1$ où ξ_i

désigne le nombre de nœuds internes de l'arbre T_i et Δ le nombre de faces tricolores du réaliseur.

Chapitre 3 : Bijection entre les réaliseurs et les paires de chemins de Dyck ne se coupant pas

Nous proposons ici une bijection entre les réaliseurs et les paires de chemins de Dyck qui ne se coupent pas [14]. Le codage d'un réaliseur par une paire de chemins de Dyck qui ne se coupent pas et le décodage se font en temps linéaire. Utilisant cette bijection, nous pouvons énumérer les réaliseurs de taille n et nous pouvons les générer exhaustivement de manière efficace. De plus, nous prouvons que le nombre de faces tricolores d'un réaliseur est asymptotiquement en moyenne de $n/2 + o(n)$.

Chapitre 4 : Génération aléatoire de pastèques

Dans le chapitre précédent nous avons considéré les paires de chemins de Dyck qui ne se coupent pas. Ici nous considérons le cas de plusieurs chemins de Dyck qui ne se coupent pas. Ces configurations sont également appelées pastèques. A l'aide de formules d'énumération [42, 69] sur les facteurs gauches de telles configurations nous proposons un algorithme de génération aléatoire de telles configurations. La complexité de cet algorithme est en $O(p^2 2^p l)$ où p est le nombre de chemins et l la longueur des branches. La bijection présentée dans le chapitre précédant nous permet donc de générer aléatoirement en temps linéaire des réaliseurs de taille n [18].

Quelques expérimentations ont été effectuées, montrant que la profondeur moyenne d'un arbre d'un réaliseur est environ de $0,97\sqrt{n}$.

Partie II : Utilisations des réaliseurs pour le dessin et le codage

Chapitre 5 : Un algorithme de dessin lignes brisées basé sur les réaliseurs

Nous observons dans un premier temps que les dessins lignes brisées, comme les dessins lignes droites [51], nécessitent une grille de surface (resp. largeur) au moins $\frac{4(n-1)^2}{9}$ (resp. $\lfloor\frac{2(n-1)}{3}\rfloor$). Nous présentons un algorithme linéaire de dessin lignes brisées utilisant les réaliseurs. Cet algorithme calcule, pour n'importe quel graphe planaire à n sommets, un dessin lignes brisées sur une grille de surface au plus $\frac{4(n-1)^2}{9}$. Les dessins obtenus ont au plus $n-2$ brisures et au plus 1 brisure par arête. Il permet d'affirmer que la surface (resp. largeur) de grille nécessaire et suffisante pour dessiner un graphe planaire à l'aide de lignes brisées est $\frac{4(n-1)^2}{9}$ (resp. $\lfloor\frac{2(n-1)}{3}\rfloor$) [16].

Chapitre 6 : Une majoration du nombre de graphes planaires à l'aide des réaliseurs

Ce chapitre présente un travail réalisé en collaboration avec Cyril Gavoille et Nicolas Hanusse [15].

Dans un premier temps nous proposons une généralisation de la notion de réaliseur minimal aux arbres recouvrants ordonnés [26].

Dans ce chapitre, nous montrons une borne supérieure de $2^{5,007n+O(\log(n))}$ sur le nombre de graphes planaires non étiquetés. Ce résultat implique que le nombre de graphes planaires étiquetés est d'au plus $n!2^{5,007n+O(\log(n))}$, améliorant ainsi la borne donnée dans [80].

Comme notre borne peut être paramétrée en fonction du nombre d'arêtes, nous pouvons montrer que la plupart des graphes planaires non étiquetés possèdent au moins $1,70n$ arêtes et au plus $2,54n$ arêtes, améliorant l'ancienne borne supérieure de $2,69n$ arêtes [80]. De plus, ce résultat est également vrai pour les graphes planaires étiquetés (ce qui améliore légèrement l'ancienne borne, $2,56n$ [80]), connexes étiquetés et connexes non étiquetés.

Mis à part l'aspect fondamental de l'énumération des graphes planaires, notre technique s'appuie sur une représentation *explicite*, relativement simple, et calculable en temps linéaire. De plus, nous donnons un algorithme linéaire de codage en $3,37n$ bits des graphes plans maximaux et en $5,03n$ bits pour les graphes planaires. Notre représentation des graphes plans maximaux améliore la compression "Edgebreaker" [67], et il ne fait pas de doute que notre construction explicite d'un graphe planaire peut être utilisée pour des problèmes de routage dans les réseaux, en particulier en améliorant le résultat de Lu [72].

Chapitre 1

Présentation des Réaliseurs

Dans ce chapitre, nous présentons les réaliseurs ainsi que de nombreuses propriétés utiles de ces objets mathématiques. Avant de pouvoir parler de réaliseurs, il est nécessaire de rappeler quelques notions classiques sur les graphes, les arbres et sur les ensembles partiellement ordonnés. Ceci fera l'objet de la première section. Dans la deuxième section, nous verrons la définition d'un réaliseur ainsi que quelques propriétés de cet objet. Dans la section 3, nous étudierons la structure de l'ensemble des réaliseurs d'un graphe plan maximal. Enfin, dans la dernière section nous verrons des généralisations de la notion de réaliseurs à des graphes plans triconnexes ainsi qu'à des graphes plans.

1.1 Préliminaires

Graphe

Un *graphe* $G = (V, E)$ est un ensemble de *sommets* V et un multi-ensemble d'*arêtes* E constituées de paires de sommets. Une arête (u, v) est une *boucle* si $u = v$. Une arête apparaissant plusieurs fois dans l'ensemble E est une *arête multiple*. Un graphe qui possède des boucles ou des arêtes multiples est un *graphe-multiple*. On dit que $G' = (V', E')$ est un *super-graphe* de $G = (V, E)$ (ou G est un *sous-graphe* de G') si $V \subseteq V'$ et $E \subseteq E'$. Le graphe $G' = (V', E')$ est un *sous-graphe induit* de G si pour tout couple de sommets (x, y) de V', $(x, y) \in E \Leftrightarrow (x, y) \in E'$. On dit que G' est le sous-graphe de G induit par V'.

Un sommet u est un *voisin* d'un sommet v s'il existe une arête $(u, v) \in E$. Le *degré* d'un sommet v, noté $\deg(v)$ désigne le nombre de ses voisins. On notera $\deg_{max}(G)$ le degré maximal du graphe G.

Un graphe peut être décrit par les listes d'adjacence de chacun de ses sommets. Une *carte* d'un graphe G est définie par l'ensemble de ces listes d'adjacence (i.e. listes des voisins de chaque sommet) où chacune de ces listes est ordonnée.

Un *graphe orienté* (ou *digraphe*) $G = (V, E)$ est un ensemble de *sommets* V et un

ensemble d'*arcs* E constitués de couples de sommets. Un arc $e = (u, v)$ est un *arc sortant* de u et un *arc rentrant* de v. Le sommet u est aussi appelé la *source* de e et v la *cible* de e. Le *degré rentrant* d'un sommet v, noté $deg^+(v)$, désigne le nombre d'arcs rentrants de v. De la même manière le *degré sortant* d'un sommet v, noté $deg^-(v)$, désigne le nombre d'arcs sortants de v.

Un *chemin* dans un graphe $G = (V, E)$ est une séquence $(v_1, v_2, \ldots, v_k)$ de sommets distincts de G telle que $\forall i / 1 \leq i \leq k-1, (v_i, v_{i+1}) \in E$. La *longueur* d'un chemin est le nombre d'arêtes du chemin. Un *cycle* est un chemin telle que $v_1 = v_k$. Un graphe sans cycle est dit *acyclique*. Un graphe est *connexe*, si pour tout couple de sommets (u, v), il existe un chemin allant de u à v. On appelle *sommet d'articulation* d'un graphe $G = (V, E)$ un sommet dont la suppression augmente le nombre de composantes connexes du graphe.

Un graphe à n sommets (avec $n \geq k+1$) est *k-connexe*, s'il faut supprimer au moins k sommets pour qu'il ne soit plus connexe. On utilise aussi le terme *biconnexe* pour désigner les graphes 2-connexes et le terme *triconnexe* pour désigner les graphes 3-connexes.

Un graphe $G = (V, E)$ est dit *complet* si pour toute paire de sommets (u, v) de G, $(u, v) \in E$. On note K_n le graphe complet à n sommets. Une *clique* d'un graphe G est un sous-graphe induit complet de G.

Par exemple, le graphe G_1 de la figure 2 possède 8 sommets et 14 arêtes. Les sommets $1, 6, 5, 7, 2, 8, 1$ forment un cycle de longueur 6. Ce graphe est de plus biconnexe, puisque si l'on supprime un sommet, le graphe reste connexe. En revanche ce graphe n'est pas triconnexe, puisque si l'on supprime les sommets 1 et 2, le graphe n'est plus connexe. Les voisins du sommet 3 sont les sommets 4,5,6 et 7. Le sommet 3 est donc de degré 4. Le graphe G_2 de la figure 2 est un exemple de graphe orienté. Dans ce graphe, on peut observer que le degré rentrant du sommet 3 est 1 et que le degré sortant du sommet 3 vaut 3. De plus, les sommets 1,6,3,5,4,1 forment un circuit de longueur 5.

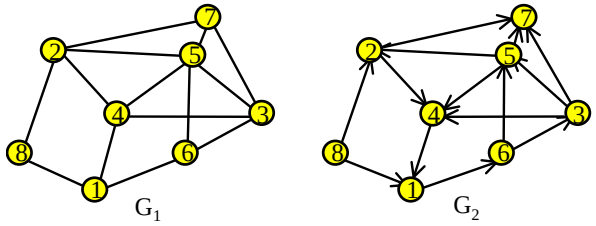

Figure 2 – A gauche, un graphe non orienté G_1. A droite un graphe G_2 qui est une orientation de G_1.

Graphes planaires et graphes plans

Un *dessin planaire* d'un graphe est un dessin sur le plan de ce graphe dont les arêtes joignant les sommets ne se croisent pas. Un graphe est *planaire* s'il admet un dessin

planaire. Une carte est *planaire*, s'il existe un dessin planaire respectant les ordres autour des sommets spécifiés par les listes d'adjacence.

Un *graphe plan* G est une carte planaire où une arête appartenant à la face extérieure est distinguée. Une telle arête est appelée *arête racine* du graphe plan G. Dans un dessin planaire d'un graphe, les arêtes partitionnent le plan en régions connexes appelées *faces*. On dit que deux faces sont *adjacentes*, si elles partagent au moins une arête. Une de ces régions est non bornée, elle est appelée *face extérieure*. Les autres faces sont dites *intérieure*.

On dit qu'une arête e (resp. un sommet v) appartient à la face f si elle (resp. il) appartient à la frontière de f. Les sommets de la face extérieure sont appelés *sommets externes*. De même, les arêtes de la face extérieure sont appelées *arêtes externes*. Une face qui ne contient pas d'arête externe est dite *face strictement intérieure*.

Théorème 1.1.1. *(**Whitney 1933**) [95]*
Soit G graphe planaire triconnexe et e une arête de G. Le graphe G admet un unique graphe plan ayant pour arête racine l'arête e.

La figure 3 représente le même graphe dessiné de deux manières différentes. On peut remarquer que ce graphe n'est pas triconnexe puisque si l'on supprime le sommet 5 le graphe se décompose en deux composantes connexes. L'une contenant uniquement le sommet 4 et l'autre contenant les sommets $2, 3, 1, 6$ et 7.

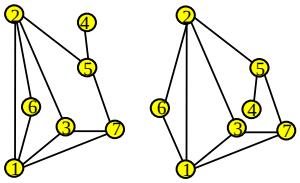

Figure 3 – Graphe planaire admettant plusieurs dessins planaires.

Un *graphe planaire maximal* G est un graphe planaire, avec au moins 3 sommets, qui si on lui ajoute une arête, il n'est plus planaire. Naturellement un *graphe plan maximal* est un graphe planaire maximal où l'on a distingué une des 3 arêtes de la face extérieure. Par la suite les 3 sommets de la face extérieure seront notés v_0, v_1, v_2 et l'arête distinguée sera l'arête (v_0, v_1).

La figure 4 représente un exemple de graphe planaire maximal à 8 sommets.

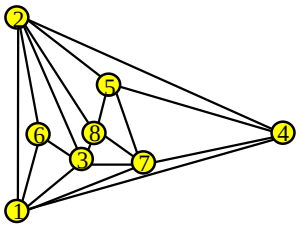

Figure 4 – Exemple de graphe planaire maximal

Le *graphe dual* G^* d'un graphe plan G est un graphe plan possédant un sommet par face de G et à chaque arête e de G, on associe une arête dans G^* entre les deux faces séparées par e.

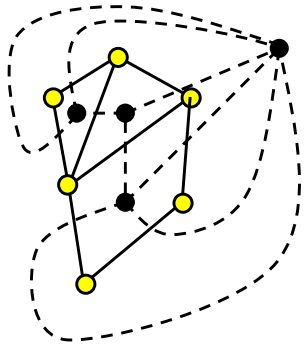

Figure 5 – Un graphe plan G ainsi que son dual G^*, représenté par les sommets noirs et les arêtes en pointillés.

Le graphe dual d'un graphe plan peut être un graphe multiple (voir figure 5). En revanche, le graphe dual d'un graphe plan triconnexe est un graphe plan simple et triconnexe.

Théorème 1.1.2. *(***Caractéristique d'Euler** *)*
Soit G un graphe plan connexe. Soient n le nombre de sommets de G, m le nombre d'arêtes de G et f le nombre de faces de G. Alors

$$n - m + f = 2$$

Si l'on considère les graphes plans de la figure 3, on peut observer qu'ils possèdent 7 sommets, 10 arêtes et 5 faces. Ils vérifient la caractéristique d'Euler : $7 - 10 + 5 = 2$.

Dans le cas d'un graphe plan maximal, le nombre d'arêtes vaut $3n - 6$, le nombre de faces vaut $2n - 4$ et le nombre de faces strictement intérieures vaut $2n - 8$.

Théorème 1.1.3. *(***Tutte 1962***) [91]*
Le nombre de graphes plans maximaux de taille n est :

$$T_n = \frac{2(4n-11)!}{(n-2)!(3n-7)!}$$

Arbres

Un *arbre* est un graphe connexe acyclique. Les sommets d'un arbre sont classiquement appelés les *nœuds* de l'arbre. Un arbre enraciné est un arbre où un sommet est distingué ; ce sommet est appelé *racine* de l'arbre. On dit que u est un *enfant* d'un sommet v si v est le successeur de u sur le chemin (un arbre étant acyclique, il y a un unique chemin entre deux sommets) entre u et la racine de l'arbre. On dit alors que v

est le *parent* de u. Si deux sommets ont le même parent, alors ils sont *frères*. On dit que v est un *ancêtre* de u s'il se trouve sur le chemin entre u et la racine de l'arbre. On dit alors que u est un *descendant* de v. Un nœud, autre que la racine, qui possède un ou plusieurs enfants est appelé *nœud interne*. A l'inverse un nœud qui ne possède pas d'enfant est une *feuille*. La *profondeur* d'un sommet est la longueur du chemin qui le sépare de la racine. La *profondeur* d'un arbre est la plus grande des profondeurs de ses nœuds.

On appelle *plus petit ancêtre commun* de u et v (ou plus proche ancêtre commun), noté $ppac(u, v)$, le sommet le plus loin de la racine étant à la fois ancêtre de u et de v. Par la suite nous considérerons que les arêtes d'un arbre enraciné sont orientées vers la racine.

On dit qu'un arbre est *ordonné enraciné* si l'ensemble des enfants de chaque sommet est ordonné.

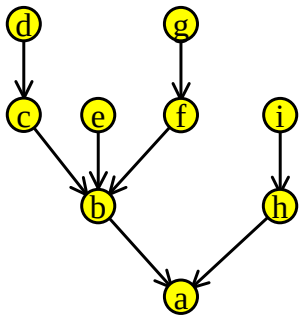

Figure 6 – Exemple d'arbre enraciné ordonné. Le nœud a est la racine de cet arbre.

Un parcours préfixe trigonométrique (resp. anti-trigonométrique) d'un arbre T consiste à parcourir l'arbre T à partir de la racine puis récursivement les sous-arbres issus de ses enfants dans l'ordre trigonométrique (resp. anti-trigonométrique). Un parcours postfixe trigonométrique (resp. anti-trigonométrique) consiste à parcourir récursivement les sous-arbres issus des enfants de la racine puis de la racine. L'*ordre postfixe trigonométrique*, l'*ordre postfixe anti-trigonométrique*, l'*ordre préfixe trigonométrique* et l'*ordre préfixe anti-trigonométrique* désignant respectivement l'ordre dans lequel les sommets sont parcourus suivant les différents parcours.

Soit $v_1, v_2, \ldots, v_n$ les sommets d'un arbre T classés dans l'ordre préfixe anti-trigonométrique. Une *branche gauche d'un arbre* T est un chemin $v_i, v_{i-1}, \ldots, v_j$ (i.e. v_{i+1} est le premier enfant de v_i), tel que $i - j$ soit maximal. Clairement les branches gauches partitionnent les sommets de T et il y a exactement une branche gauche par feuille de T. De manière symétrique on définit une *branche droite d'un arbre* en considérant les sommets $v_1, v_2, \ldots, v_n$ classés dans l'ordre préfixe trigonométrique de T. La *branche droite d'un sommet* (resp. *branche gauche d'un sommet*) dans un arbre T est l'unique branche droite (resp. gauche) de T qui contient ce sommet.

Considérant l'arbre de la figure 6. L'ordre préfixe trigonométrique est : $a, h, i, b, f, g, e, c, d$. L'ordre préfixe anti-trigonométrique est $a, b, c, d, e, f, g, h, i$. L'ordre

postfixe trigonométrique est : $i, h, g, f, e, d, c, b, a$. L'ordre postfixe anti-trigonométrique est : $d, c, e, g, f, b, i, h, a$. La branche droite du sommet b est constituée des sommets g, f, b. Le plus petit ancêtre commun de d et f est le sommet b.

Définissons maintenant deux relations d'ordre sur les arbres, $\leq_{cw}$ et $\leq_{ccw}$:

Définition 1.1.1. *Soient T et T' deux arbres (enracinés et ordonnés) ayant k nœuds. Soient $n_1, n_2, \ldots, n_k$ et $m_1, m_2, \ldots, m_k$, les nœuds de T et T' dans l'ordre préfixe anti-trigonométrique (resp. trigonométrique). Si $T = T'$ alors $T \leq_{cw} T'$ et $T \leq_{ccw} T'$. Sinon, soit i le premier indice tel que $\deg(n_i) \neq \deg(m_i)$. Si $\deg(n_i) < deg(m_i)$ alors $T \leq_{cw} T'$ (resp. $T \leq_{ccw} T'$).*

Naturellement, $T \geq_{cw} T'$ est une autre notation pour $T' \leq_{cw} T$. De même, $T <_{cw} T'$ signifie $T \leq'_{cw} T$ et $T \neq T'$. Les notations *cw* et *ccw* viennent de l'anglais "clockwise" et "counter-clockwise".

Si l'on considère l'exemple de la figure 7, on peut remarquer que les deux sommets gris sont respectivement les premiers nœuds (dans l'ordre préfixe anti-trigonométrique) de T et T' qui n'ont pas le même nombre d'enfants. Le sommet gris de l'arbre T possède plus d'enfants que celui de l'arbre T' donc $T >_{cw} T'$. De manière similaire, les deux nœuds noirs sont les premiers (dans l'ordre préfixe trigonométrique) qui n'ont pas le même nombre d'enfants. Le sommet noir de l'arbre T' possède plus d'enfants que celui de l'arbre T donc $T <_{ccw} T'$.

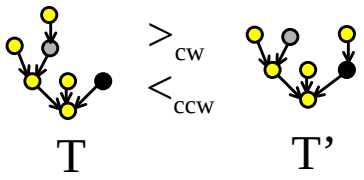

Figure 7 – Illustration de la relation d'ordre entre les arbres.

Ensemble partiellement ordonné

Définition 1.1.2. *Un ordre partiel $\leq$ sur un ensemble E est une relation binaire qui vérifie les propriétés suivantes :*

- *réflexivité : pour tout $x \in E, x \leq x$.*
- *anti-symétrie : pour tous $x, y \in E$, si $x \leq y$ et $y \leq x$ alors $x = y$.*
- *transitivité : pour tous $x, y, z \in E$, si $x \leq y$ et $y \leq z$ alors $x \leq z$.*

Un *ensemble partiellement ordonné* (ou *EPO*) $P = (E, \leq)$ est une paire constituée d'un ensemble E, appelé *domaine*, et d'un ordre partiel $\leq$ sur E. Souvent on écrit $x \in P$ au lieu de $x \in E$. L'ordre $\leq$ est dit *total* si pour tout couple (x, y) d'éléments de E, $x \leq y$ ou $y \leq x$. On dit qu'un ordre total $\leq_L$ sur E est une *extension linéaire* d'un EPO $(E, \leq)$ si pour tout couple d'élément $(x, y) \in E^2$, $x \leq y$ implique $x \leq_L y$. On dit que $(\leq_1, \leq_2, \ldots, \leq_k)$ est un *réaliseur* de l'ordre $\leq$ si pour tout couple d'élément (x, y)

$x \leq y$ est équivalent à $x \leq_i y$ pour tout $i \leq k$. La *dimension* d'un EPO $P = (E, \leq)$ est le plus petit nombre k tel que $(\leq_1, \leq_2, \ldots, \leq_k)$ soit un réaliseur de P.

On dit que x *couvre* y si $y < x$ et s'il n'existe pas d'élément z tel que $y < z < x$. Le *diagramme de Hasse* d'un EPO P est un graphe H orienté dont les sommets sont les éléments de P et dont les arcs sont les couples (y, x) tels que x couvre y. Les sommets sont positionnés dans le plan de telle sorte que toutes les arêtes soient orientées vers le haut.

Si l'on considère l'ensemble $X = \{a, b, c, d, e\}$ et la relation d'ordre R définie par les relations de couverture suivantes : e couvre d, c et d couvrent b et b couvre a. La partie gauche de la figure 10 montre le Diagramme de Hasse de l'EPO $P = (X, R)$.

Les ordres $<_1$: $abdcef$ et $<_2$: $cbfaed$ constituent un réaliseur de l'EPO P de la figure 8. Comme l'EPO P n'est clairement pas de dimension 1, il est donc de dimension 2.

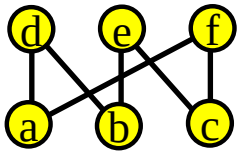

Figure 8 – Exemple d'un EPO P de dimension 2.

Définition 1.1.3. *Soit $G = (V, E)$ un graphe. L'ordre partiel $<_G$ sur $V \cup E$ est défini de la manière suivante : $a <_G b \equiv a \in V, b \in E$ et a est une extrémité de b. $P(G) = (V \cup E, <_G)$ est appelé EPO d'incidence de G*

La figure 9 illustre la définition précédente.

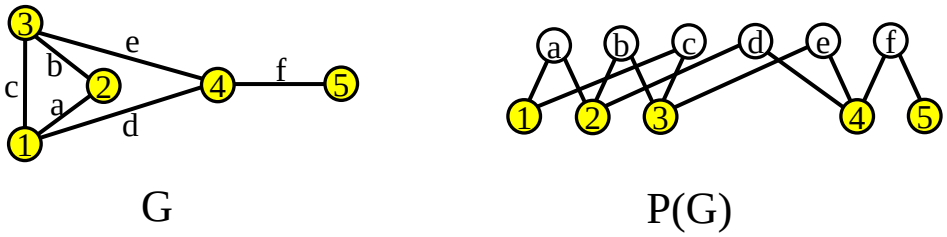

Figure 9 – A gauche un graphe G. A droite le diagramme de Hasse l'EPO d'incidence de G.

Un *idéal* I d'un EPO P est un sous-EPO de P vérifiant $\forall x \in I, y < x$ et $y \in I$. On note classiquement $L(P)$ l'ensemble des idéaux de P muni de la relation d'inclusion. Un élément x d'un EPO P est *minimal* si P ne contient pas d'élément plus petit que x. De même, un élément x d'un EPO P est *maximal* si P ne contient pas d'élément plus grand que x. Un EPO est *borné* s'il possède un unique élément maximal et un unique élément minimal. Une *borne supérieure* d'une paire éléments x, y d'un EPO est un élément z tel que $x \leq z$ et $y \leq z$. Une *plus petite borne supérieure* (ou *lub*, pour

least upper bound) pour les éléments x, y d'un EPO est une borne supérieure z de x, y telle que toute autre borne supérieure z' de x, y vérifie l'inégalité suivante : $z \leq z'$. Un tel élément, s'il existe est noté $x \vee y$. De manière similaire, une *borne inférieure* d'un paire éléments x, y d'un EPO est un élément z tel que $z \leq x$ et $z \leq y$. Une *plus grande borne inférieure* (ou *glb*, pour greatest lower bound) pour les éléments x, y d'un EPO est une borne inférieure z de x, y telle que toute autre borne inférieure z' de x, y vérifie l'inégalité suivante : $z' \leq z$. Un tel élément, s'il existe, est noté $x \wedge y$.

Un *treillis* est un EPO dans lequel chaque paire d'éléments possède une unique plus petite borne supérieure et une unique plus grande borne inférieure. Un *treillis distributif* L est un treillis qui vérifie la condition suivante : pour tous $x, y, z \in L$, $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$ et $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$.

Théorème 1.1.4. *(***Birkhoff 1933** *: théorème fondamental des treillis distributifs finis) [10]*
Pour chaque treillis distributif fini L, il existe un unique EPO P tel que $L = L(P)$.

La figure 10 illustre le théorème précédent. A gauche le diagramme de Hasse d'un EPO P et à droite le diagramme de Hasse du treillis des idéaux de P.

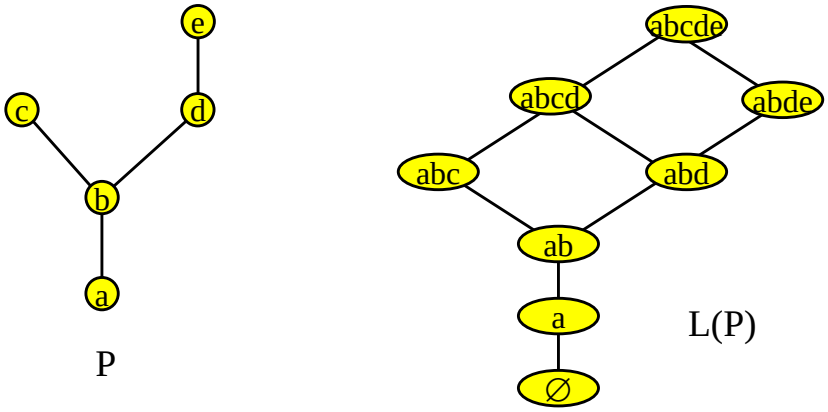

Figure 10 – Diagramme de Hasse d'un EPO P ainsi que celui de son treillis distributif de ses idéaux $L(P)$.

1.2 Réaliseur d'un graphe plan maximal : définition et propriétés

Définition 1.2.1. *(***Schnyder 1989***) [86]*
Un réaliseur d'un graphe plan maximal G est une partition des arêtes internes de G en trois ensembles T_0, T_1 et T_2 d'arêtes orientées, telle que chaque sommet interne v les conditions suivantes sont respectées :

1. *Le sommet v possède exactement une arête sortante dans chacun des ensembles T_0, T_1 et T_2.*

2. **Condition locale** *: les arêtes incidentes à v apparaissent dans le sens trigonométrique de la manière suivante : une arête sortante dans T_0, éventuellement des arêtes rentrantes dans T_2, une arête sortante dans T_1, éventuellement des arêtes rentrantes dans T_0, une arête sortante dans T_2 et éventuellement des arêtes rentrantes dans T_1 (voir figure 11).*

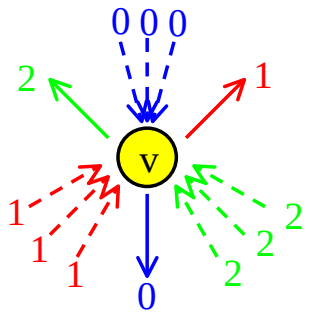

Figure 11 – **Condition locale** : orientation et coloration des arêtes autour de chaque sommet interne.

Théorème 1.2.1. *(**Schnyder 1989**) [86]*
Soit G un graphe plan maximal possédant au moins 3 sommets. Soit $R = (T_0, T_1, T_2)$ un réaliseur de G. Chaque ensemble T_i est un arbre contenant tous les sommets internes de G ainsi que le sommet v_i.

Soit $<_i$ la relation d'ordre induite par T_0 sur les sommets de $G = (V, E)$: $x <_i y$ si et seulement si x est un descendant de y dans T_i. A partir de ces trois relations d'ordres partiels sur l'ensemble V, Schnyder déduit trois ordres totaux sur $V \cup E$.

Théorème 1.2.2. *(**Schnyder 1989**) [86]*
Soit G un graphe. G est planaire si et seulement si $<_G$ est de dimension 3.

Une preuve de ce théorème est également disponible dans [59].

Théorème 1.2.3. *(**Schnyder 1990**) [87]*
Tout graphe plan maximal admet au moins un réaliseur qui peut être calculé en temps linéaire.

La figure 12 présente un exemple de graphe ainsi que l'un de ses réaliseurs.

Par la suite, on dira que les arêtes de l'arbre T_i seront coloriées avec la couleur i, où $i \in \{0, 1, 2\}$. Le bleu désignera la couleur "0", le rouge la couleur "1" et le vert la couleur "2". Lorsque nous désignerons les couleurs, nous écrirons $i + 1$ à la place de $(i + 1) \mod 3$ et $i - 1$ à la place de $(i - 1) \mod 3$. Pour chaque arbre T_i, on note $\bar{T}_i$ l'arbre composé des arêtes de T_i augmenté des arêtes (v_{i-1}, v_i) et (v_{i+1}, v_i).

Soient u_1 et u_2 deux sommets d'un réaliseur de G. On écrit $u_1 >^i_{ccw} u_2$ (resp. $u_1 >^i_{cw} u_2$) si u_1 est après u_2 dans l'ordre préfixe trigonométrique (resp. anti-trigonométrique) de l'arbre T_i. Le parent d'un sommet u de T_i est noté $P_i(u)$. On notera également $Ch_i(u)$ la liste des enfants de u dans l'ordre anti-trigonométrique. De même, $Ch_i(u, k)$ désigne

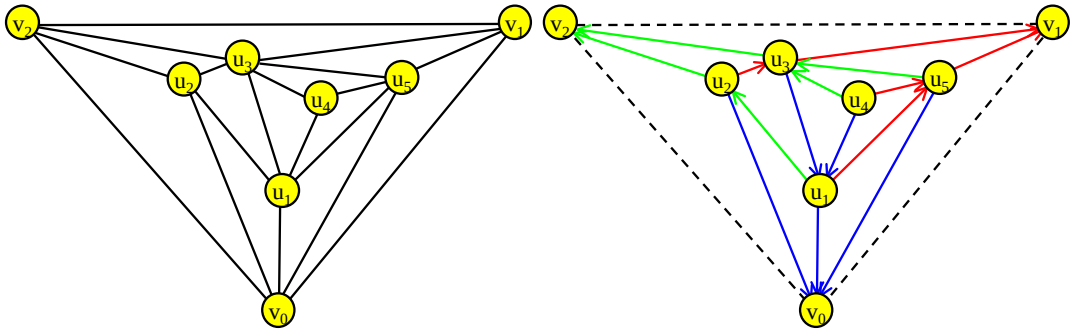

Figure 12 – Un exemple de réaliseur. A gauche, le graphe sous-jacent et à droite, l'un de ses réaliseurs.

le k-ième enfant de u dans l'arbre T_i. Soient u_1 et u_2 deux sommets d'un réaliseur tels que u_1 est un descendant de u_2 dans l'arbre T_i. On note $u_1 \xrightarrow{i} u_2$ le chemin de u_1 à u_2 dans l'arbre T_i.

Dans l'exemple de la figure 12, on peut observer que $Ch_0(u_1) = (u_3, u_4)$ et que $P_2(u_5) = u_3$.

Soit $F = (e_0, e_1, e_2)$ une face d'un graphe plan maximal G avec $e_j = \{u_j, u_{j+1}\}$. Soit $R = (T_0, T_1, T_2)$ un réaliseur de G.

Propriété 1.2.1. *Si u_1 est le parent de u_2 dans l'arbre T_i alors $u_1 >_{cw}^{i+1} u_2$ et $u_2 >_{cw}^{i-1} u_1$.*

La preuve de cette propriété découle des deux faits qui suivent.

Fait 1.2.1. *Supposons que e_0 soit coloriée i.*
Si e_0 et e_1 sont orientées vers u_1 alors e_1 est coloriée i.

Démonstration. Si e_1 est coloriée i, alors le parent du sommet u_1 dans T_{i+1} se trouve dans la face F. Ceci est impossible. □

Fait 1.2.2. *Supposons que l'arête e_0 soit coloriée i.*

Si e_0 et e_1 sont respectivement orientées vers u_1 et u_2 alors l'arête e_1 est coloriée $i+1$.

De manière similaire, si e_0 et e_2 sont respectivement orientées vers u_0 et u_1 alors l'arête e_2 est coloriée 1.

Démonstration. Supposons que les arêtes e_0 et e_1 soient respectivement orientées vers u_1 et u_2. Si e_1 n'est pas coloriée $i+1$, le parent du sommet u_1 dans T_{i+1} serait dans la face F. Ceci est impossible. En appliquant un argument similaire, on montre la deuxième partie du fait. □

Une conséquence des deux faits précédents, est que les seules colorations possibles pour une face strictement intérieure sont celles représentées par la figure 13. Remarquons, que parmi les colorations possibles, seules les 2 premières configurations utilisent

les trois couleurs. Par la suite, nous dirons qu'une face coloriée de cette manière est une *face tricolore*. Toutes les autres colorations utilisent 2 couleurs. Par la suite, on appellera *cw-face* (resp. *ccw-face*) une face tricolore dont les arêtes tournent dans le sens anti-trigonométrique (resp. trigonométrique). De même on appellera *cw-triangle* (resp. *ccw-triangle*) un 3-cycle tricolore dont les arêtes tournent dans le sens anti-trigonométrique (resp. trigonométrique).

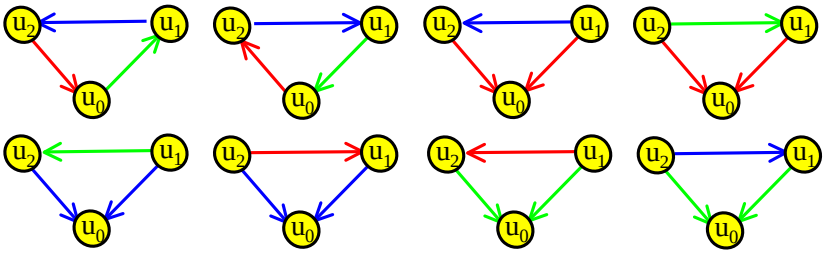

Figure 13 – 8 colorations possibles des arêtes d'une face.

Propriété 1.2.2. *Soit $R = (T_0, T_1, T_2)$ un réaliseur. Si u est un descendant de v dans l'arbre T_i, alors u ne peut être un descendant ou un ancêtre de v dans T_j pour $i \neq j$.*

Démonstration. Supposons que u est un descendant de v dans T_0 et u est un ancêtre de v dans T_1.

Comme le sommet u satisfait la condition locale, $P_1(u)$ est dans la région délimitée par le cycle $C = (u \xrightarrow{0} v, v \xleftarrow{1} u)$ (voir figure 14). Soit t le sommet commun à C et au chemin $u \xrightarrow{1} v_1$. Le sommet t ne peut être sur le chemin $u \xrightarrow{0} v$, à cause de la condition locale appliquée sur le sommet t. Donc t se trouve sur le chemin $u \xrightarrow{1} v$. Dans ce cas, nous avons un cycle colorié 1 : $t \xleftarrow{1} u, u \xleftarrow{1} t$ (voir figure 14). Ceci est impossible car l'ensemble des arêtes forme un arbre (T_1). Donc, si u est un descendant de v dans T_0 alors u ne peut pas être un ancêtre de v dans T_1.

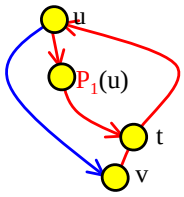

Figure 14 – Configuration impossible où u est un descendant de v dans T_0 et u est un ancêtre de v dans T_1.

Un raisonnement similaire pourrait être effectué dans le cas où u serait un descendant de v dans l'arbre T_0, alors u ne pourrait pas être un descendant v dans T_1. Par symétrie les autres cas se ramènent à ces deux cas. □

Propriété 1.2.3. *Soit $R = (T_0, T_1, T_2)$ un réaliseur. Soit (u, v) une arête de R. Si $u = P_1(v)$ où $u = P_2(v)$ ou $u \in Ch_0(v)$ alors u est après v dans l'ordre postfixe trigonométrique de T_2.*

Démonstration. Considérons les trois cas suivants :

- $u = P_2(v)$: évident.
- $u = P_1(v)$: Supposons que u soit avant v dans l'ordre postfixe trigonométrique de T_2. Soit w le plus petit ancêtre commun de u et de v dans l'arbre T_2. Le cycle $C = ((v,u), u \underline{2}_{\rightarrow} w, w \underline{2}_{\leftarrow} v)$ détermine une région du plan (voir figure 15 a.). Afin de respecter la condition locale sur le sommet u, $P_0(u)$ doit être dans cette région. Considérons le sommet t qui appartient à $u \underline{0}_{\rightarrow} v_0$ et au cycle C. La condition locale sur le sommet t implique que t se situe sur le chemin $u \underline{2}_{\rightarrow} w$. Donc t est un ancêtre de u dans l'arbre T_1. Ceci est en contradiction avec la propriété 1.2.2.
- $u \in Ch_0(v)$: Soit w le plus petit ancêtre commun à $P_1(v)$ et v dans l'arbre T_2. Le sommet u doit se trouver dans la région délimitée par le cycle $C = ((v, P_1(v)), P_1(v) \underline{2}_{\rightarrow} w, w \underline{2}_{\leftarrow} v)$ (voir figure 15 b.). Dans cette région, tous les sommets sont après le sommet v dans l'ordre postfixe trigonométrique de T_2. Donc u est après v dans cet ordre.

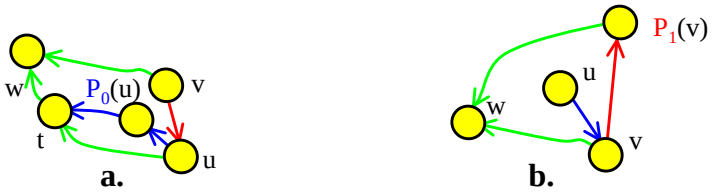

Figure 15 – **a.** Configuration impossible où $P_1(v)$ est avant v dans l'ordre postfixe trigonométrique de T_2. **b.** Région délimitée par $C = ((v, P_1(v)), P_1(v) \rightarrow^2 w, w \rightarrow^2 v)$.

□

Définition 1.2.2. *Une 3-orientation d'un graphe plan maximal G est une orientation des arêtes de G où tous les sommets internes de G sont de degré sortant 3.*

Théorème 1.2.4. *(***Fraysseix et Ossona de Mendez, 1994***) [48]*
Soit G un graphe plan maximal. Les réaliseurs de G sont en bijection avec les 3-orientations de G.

Les ordres canoniques ont été introduits par Fraysseix, Pach et Pollack [51] pour les graphes plans maximaux. Plus tard Kant [64] a proposé une généralisation de cette définition aux graphes plans triconnexes.

Définition 1.2.3. *(***Fraysseix, Pach et Pollack, 1990***) [51]*
Soit G un graphe plan maximal. Un Ordre Canonique est un ordre total sur les sommets de G $u_0 = v_0, u_1 = v_1, u_2, u_3, \ldots, u_n = v_2$ tel que pour tout $4 \leq k \leq n$:

1. *le sous-graphe induit $G_{k-1} \subset G$ induit par $u_0, u_1, \ldots, u_{k-1}$ soit biconnexe et que le cycle C_{k-1} constitué des arêtes de la face extérieure contient l'arête (v_0, v_1).*
2. *v_k appartient au cycle C_k et ses voisins dans G_{k-1} forment un chemin dans $C_{k-1} \subset \{v_0, v_1\}$.*

Propriété 1.2.4. *[49] Soit $R = (T_0, T_1, T_2)$ un réaliseur d'un graphe plan G. Le parcours préfixe de l'arbre T_0 est un ordre canonique de G.*

Si on considère le réaliseur de la figure 12, on peut observer que l'ordre $v_0, v_1, u_5, u_1, u_4, u_3, u_2, v_2$ est bien un ordre canonique.

1.3 Treillis des réaliseurs d'un graphe plan maximal

Définition 1.3.1. *Soit R un réaliseur d'un graphe plan maximal et C un ccw-triangle (resp. cw-triangle) de R. L'opération de recoloration S_+ sur C (resp. S_-) est définie de la manière suivante :*

1. *Inverser l'orientation des arêtes de C ;*
2. *Incrémenter (resp. Décrémenter) la couleur des arêtes du cycle C ;*
3. *Décrémenter (resp. Incrémenter) la couleur des arêtes à l'intérieur de C ;*
4. *Laisser les autres arêtes du réaliseur inchangées.*

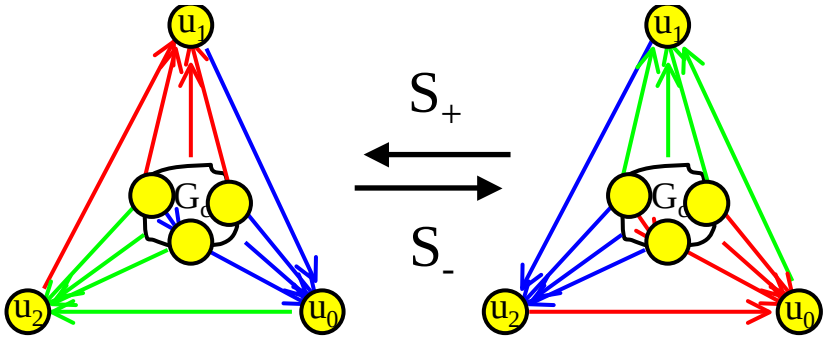

Figure 16 – Recoloration d'un triangle d'un réaliseur.

Fait 1.3.1. *Soit G un graphe plan maximal et R un réaliseur de G. Si l'on applique une opération S_+ (resp. S_-) sur un ccw-triangle (resp. cw-triangle) C de R, on obtient un nouveau réaliseur de G.*

Propriété 1.3.1. *Si un réaliseur R contient un k-cycle tricolore anti-trigonométrique (resp. trigonométrique) alors il contient un cw-triangle (resp. ccw-triangle).*

Démonstration. Soit C un cycle tricolore anti-trigonométrique composé du chemin $u_0 \xrightarrow{1} u_1$, $u_1 \xrightarrow{2} u_2$, $u_2 \xrightarrow{0} u_0$ (le cas d'un cycle tricolore trigonométrique est complètement symétrique).

Supposons, sans perte de généralité, que le chemin $u_0 \underset{\rightarrow}{1} u_1$ soit de longueur au moins 2. Soit t_1 le successeur de u_0 dans ce chemin.

Le chemin $t_1 \underset{\rightarrow}{2} v_2$ intersecte soit le chemin $u_2 \underset{\rightarrow}{0} u_0$ soit le chemin $u_1 \underset{\rightarrow}{2} u_2$. De cette manière on obtient un nouveau cycle tricolore anti-trigonométrique (voir figure 17) strictement inclus dans la région délimitée par le cycle C. En itérant cette construction on obtient finalement un cw-triangle.

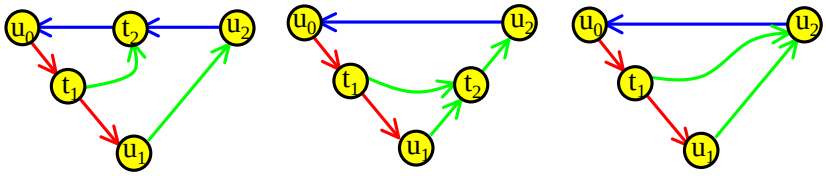

Figure 17 – Chaque k-cycle tricolore contient un triangle tricolore.

□

Lemme 1.3.1. *[79]*
Soit R un réaliseur d'un graphe plan maximal G. Soit C un ccw-triangle (resp. cw-triangle). Alors si l'on applique S_+ (resp. S_-) sur C on obtient un nouveau réaliseur R' de G.

Cette opération induit une relation sur les réaliseurs : on dit que $R \preceq R'$ si R' peut être obtenu à partir de R en appliquant des opérations S_+.

Propriété 1.3.2. *$\preceq$ est une relation d'ordre.*

Démonstration. Soit $R = (T_0, T_1, T_2)$ un réaliseur et un cw-triangle $C = (u_0, u_1, u_2)$ comme représenté sur la figure 16. L'arête (u_0, u_2) est coloriée 2 et est orientée vers u_2. D'après la propriété 1.2.1, $u_2 <^0_{cw} u_0$. Soit $R' = (T'_0, T'_1, T'_2)$ le réaliseur obtenu à partir de R en appliquant l'opération S_+ sur C. On peut observer que $deg_0(u_2) < deg'_0(u_2)$. Ceci implique que $T_0 <_{cw} T'_0$, et donc, que l'opération S_+ fait strictement croître l'arbre T_0. Ceci montre que la relation $\preceq$ est une relation anti-symétrique. De plus, cette relation est de manière évidente réflexive et transitive, donc il s'agit bien d'une relation d'ordre. □

Théorème 1.3.1. *(**[79, 83]**)*
Soit G un graphe plan maximal. L'EPO $(\mathcal{R}(G), \preceq)$ est un treillis distributif.

Remarquons que les 3-orientations constituent un cas particulier des c-orientations considérées par Propp [83]. Une démonstration indépendante de la structure de treillis distributif des c-orientations est donnée dans [83].

Le *réaliseur minimal*, noté $R_-(G)$, est le réaliseur de G qui ne possède pas de cw-triangle. Le *réaliseur maximal*, noté $R_+(G)$ est le réaliseur de G qui ne possède pas de ccw-triangle.

Soit un réaliseur $R = (T_0, T_1, T_2)$. Soit $u_1 = v_0, u_2 = v_2, u_3, \ldots, u_{n-1}, u_n = v_1$ les sommets de $\bar{T_0}$ dans l'ordre préfixe anti-trigonométrique. On dit que $\bar{T_0}$ vérifie

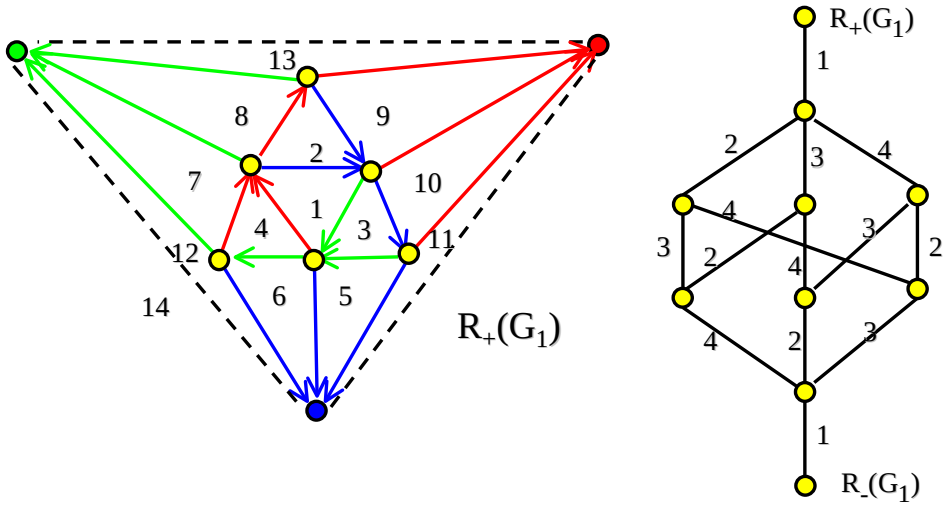

Figure 18 – A gauche, le réaliseur maximal du graphe plan G_1. A droite, la structure du treillis des réaliseurs du graphe G_1 est représentée. Le numéro sur une arête correspond au numéro de la face qu'il faut recolorier pour passer d'un réaliseur à un autre.

la *propriété branche* si et seulement si pour chaque sommet u_j et $u_i = P_0(u_j)$, soit $P_2(u_j) = P_2(u_i)$ soit $u_k = P_2(u_j)$ avec $i < k < j$ (i.e. $P_2(u_j)$ est un descendant de u_i dans $\bar{T}_0$).

Dans le réaliseur de la figure 12, les sommets u_1, u_3, u_4 et u_5 vérifient la propriété branche. En revanche, le sommet u_2 ne la vérifie pas.

Propriété 1.3.3. *Soit $R = (T_0, T_1, T_2)$ un réaliseur. Le réaliseur R est un réaliseur minimal si et seulement si $\bar{T}_i$ vérifie la propriété branche dans R, pour chaque $i \in \{0, 1, 2\}$.*

Démonstration. Montrer que R est minimal si et seulement si $\bar{T}_i$ vérifie la propriété branche dans R, pour chaque $i \in \{0, 1, 2\}$, est équivalent à montrer que R est minimal si et seulement si $\bar{T}_0$ vérifie la propriété branche dans R. En effet, la propriété que R soit minimal ou pas, est stable par permutation circulaire des arbres T_i.

Supposons que R ne soit pas minimal et que $\bar{T}_0$ vérifie la propriété branche. Soit (u, v, w) un cw-triangle de R. Supposons, sans perte de généralité que $u = P_0(w)$. Dans ce cas, $P_2(u)$ est égale à $P_2(w)$ ou $P_2(w)$ est un descendant de u contenu dans la région dont la frontière est le cw-triangle (u, v, w). Clairement il est impossible que $P_2(u) = P_2(w)$. D'après la condition locale, $P_2(w)$ se trouve à l'extérieur de la région dont la frontière est le cw-triangle (u, v, w), d'où la contradiction.

Supposons maintenant que R soit minimal. Soit u_j un sommet interne de R. Soient $u_i = P_0(u_j), u_k = P_2(u_j)$ et $u_l = P_2(u_i)$ (voir figure 19). Supposons que $k < l$ (ce qui revient à dire que $\bar{T}_0$ ne vérifie pas la propriété branche), et montrons la contradiction.

Soit $u_h = P_0(u_l)$, et soit P le chemin u_h vers u_k dans $\bar{T}_0$. Notons que

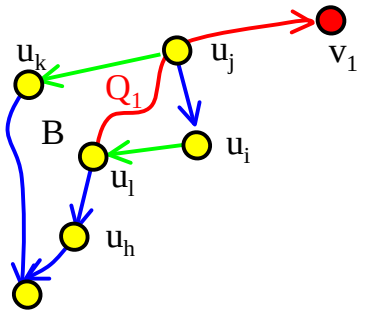

Figure 19 – Illustration des notations de la preuve de la propriété 1.3.3.

$u_l \notin P$ car u_l n'est pas un ancêtre de u_k $(k < l)$. Considérons le cycle $C = (u_i, u_l), (u_l, u_h), P, (u_k, u_j), (u_j, u_i)$ et soit B la région qui admet pour frontière C. Soit Q_1 le chemin dans $\bar{T}_1$de u_l à v_1, la racine de $\bar{T}_1$. D'après la condition locale, la première arête de Q_1 doit appartenir à $B \cup C$. Comme $v_1 \notin B$, Q_1 doit intersecter C. D'après la propriété 1.2.2, l'intersection doit se faire en un sommet u_t tel que $t > l$. Cette intersection ne peut être u_i, toujours d'après la condition locale sur le sommet u_i. Puisque chaque sommet u_r de P, $r \leq \max\{h, k\} < l$, on a également $P \cap Q_1 = \emptyset$, Q_1 insersecte donc C en u_j. Le cycle composé de la partie de Q_1 allant de u_l à u_j et des arêtes (u_j, u_i) et (u_i, u_l) est un cycle tricolore anti-trigonométrique. D'après la propriété 1.3.1 le réaliseur R contient donc un cw-triangle. Ceci est en contradiction avec le fait que R soit minimal. □

Définition 1.3.2. *Un graphe G est dit k-dégénéré si la suppression récursive de tous les sommets de degré inférieur ou égal à k conduit au graphe à zéro sommet.*

Si l'on considère le graphe de la figure 20, on observe que les sommets 1 et 2 sont de degré 3. Lorsque l'on supprime ces deux sommets, le sommet 3 devient de degré 3. Si maintenant on supprime ce sommet, on obtient le graphe K_4, dont tous les sommets sont de degré 3.

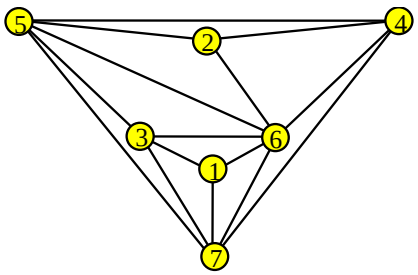

Figure 20 – Exemple de graphe 3 dégénéré.

Propriété 1.3.4. *Un graphe plan maximal admet un unique réaliseur si et seulement s'il est 3-dégénéré.*

Démonstration. Soit G un graphe plan maximal 3-dégénéré. Considérons une séquence qui supprime récursivement tous les sommets de degré ≤ 3. Le graphe étant triconnexe, chaque sommet supprimé est de degré exactement égal à 3. A chaque fois que l'on supprime un sommet, on obtient un nouveau graphe plan maximal. Donc un graphe 3-dégénéré peut être obtenu à partir de K_4 en insérant successivement un sommet au milieu d'une face et en reliant ce nouveau sommet aux 3 sommets de la face.

Montrons par induction qu'un graphe plan maximal 3-dégénéré admet un unique réaliseur. Clairement K_4 admet un unique réaliseur. Supposons que tous les graphes plans maximaux 3-dégénérés de taille n admettent un unique réaliseur.

Soit G un graphe plan maximal 3-dégénéré de taille $n+1$. Soit v un sommet de degré 3 de G. Soit $\{u_1, u_2, u_3\}$ les voisins de v dans G.

Soit $G' = G \setminus \{v\}$. Par hypothèse de récurrence, G' admet un unique réaliseur R'. Ce réaliseur R' ne contient pas de triangle tricolore. Quelle que soit la coloration de la face (u_1, u_2, u_3) du réaliseur R', il existe une unique manière de colorier et d'orienter les arêtes $(v, u_1), (v, u_2)$ et (v, u_3), pour obtenir un réaliseur R de G à partir de R'. Le sommet v ne possède pas d'arête entrante donc il ne peut pas appartenir à un triangle tricolore et donc R ne possède pas de triangle tricolore.

Le treillis des réaliseurs de G est donc réduit à un unique réaliseur R.

Soit G un graphe plan maximal qui n'est pas 3-dégénéré. Soit G' le graphe obtenu après suppression récursive de tous les sommets de degré 3. Le graphe plan G' est aussi plan maximal.

Montrons dans un premier temps que G' contient un sous-graphe plan maximal G_1 de plus de 5 sommets ne contenant pas K_4.

On peut remarquer que G' contient au moins 5 sommets. Donc si G' ne contient pas K_4, on prend $G_1 = G'$. Sinon, soient u_1, u_2, u_3, u_4 quatre sommets de G' qui forment une clique.

Cette clique partitionne le plan en quatre composantes connexes, dont trois d'entre elles sont bornées. Soit (u_1, u_2, u_3) la frontière de la composante du plan non bornée. Si toutes les composantes connexes bornées sont vides (ne contiennent pas de sommets) u_4 serait de degré 3, ce qui est en contradiction avec la définition de G'. Supposons sans perte de généralité que la région bornée par le triangle (u_1, u_2, u_4) ne soit pas vide. Soit G_2 le graphe plan maximal contenu dans la région (u_1, u_2, u_4) ayant pour face extérieure (u_1, u_2, u_4). Ce graphe est au moins de taille 5 (dans le cas contraire le sommet interne serait de degré 3). En réitérant cette construction sur G_2 on obtient un graphe G_1 qui ne contient pas K_4.

Soit (t_0, t_1, t_2) les sommets de la face extérieure de G'. Si ces trois sommets forment un triangle alors il est tricolore. Dans le cas contraire supposons, sans perte de généralité, que les arêtes (t_1, t_0) et (t_2, t_0) soient coloriées 0 et que (t_2, t_1) soit coloriée 1. Soit u le voisin commun à t_1 et t_2 dans G_1. D'après la condition locale appliquée respectivement sur t_2 et t_1, clairement $P_2(u) = t_2$ et $P_1(u) = t_1$ (voir figure 21).

Comme G' ne contient pas K_4, $u' = P_0(u) \neq t_0$. De même, soit $P_1(u')$ est différent de t_1 soit $P_2(u')$ est différent de t_2. Supposons donc que $P_1(u') \neq t_1$. En appliquant

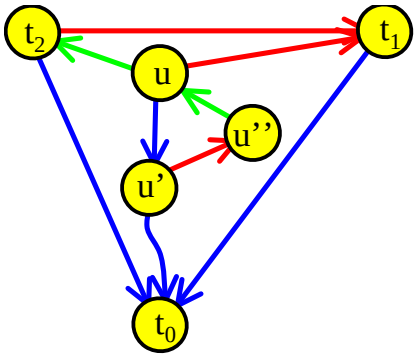

Figure 21 – Illustration des notations de la preuve de la propriété 1.3.4.

encore une fois la condition locale, on montre que $P_2(P_1(u')) = u$, et donc que G' contient un triangle tricolore.

Comme nous l'avons vu dans la première partie de la preuve, il est possible de construire un réaliseur R de G à partir de R'. Le triangle tricolore de R' est également un triangle tricolore de R. Donc G admet plusieurs réaliseurs. □

Propriété 1.3.5. *Soit G un graphe plan maximal. Soit G' un autre dessin de G. Il existe une bijection entre $R(G)$ et $R(G')$.*

Démonstration. Pour montrer cette propriété, nous allons considérer dans un premier temps que G possède comme face extérieure (v_0, v_1, v_2) et que G' possède comme face extérieure (v'_0, v_1, v_2). En d'autres termes, la face extérieure de G' est une face adjacente à la face extérieure de G. Soit G_R (resp. G_L) le sous-graphe de G situé dans la région entourée par $v'_0 \underrightarrow{v}_0, (v_0, v_1), (v_1, v'_0)$ (resp. $(v'_0, v_2), (v_2, v_0) v_0 \underleftarrow{v}_0$)

L'opération de retournement est définie comme suit :

1. Retourner les arêtes du chemin $v'_0 \underrightarrow{0} v_0$.
2. Recolorier les arêtes de G_L coloriées 0 en 1 et celles coloriées 1 en 0.
3. Recolorier les arêtes de G_R coloriées 0 en 2 et celles coloriées 2 en 0.

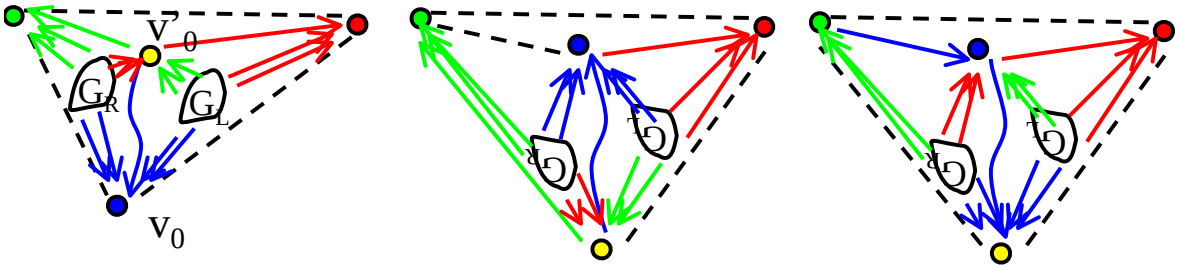

Figure 22 – Opération de changement de face extérieure sur un réaliseur.

Cette opération définit de manière évidente une bijection entre les réaliseurs de G et ceux de G'. Maintenant si la face extérieure de G' n'est pas voisine de la face extérieure

de G, il existe une suite de faces deux à deux adjacentes allant de la face extérieure de G à celle de G'. On peut construire de proche en proche une bijection permettant de passer des réaliseurs de G à ceux de G'. □

La figure 23 représente le réaliseur minimal d'un graphe plan G_2. Ce graphe plan diffère de celui de la figure 18 uniquement par le choix de la face extérieure. Comme on peut le constater les deux treillis de réaliseurs ont bien le même nombre d'éléments. En revanche, ils ne sont pas isomorphes.

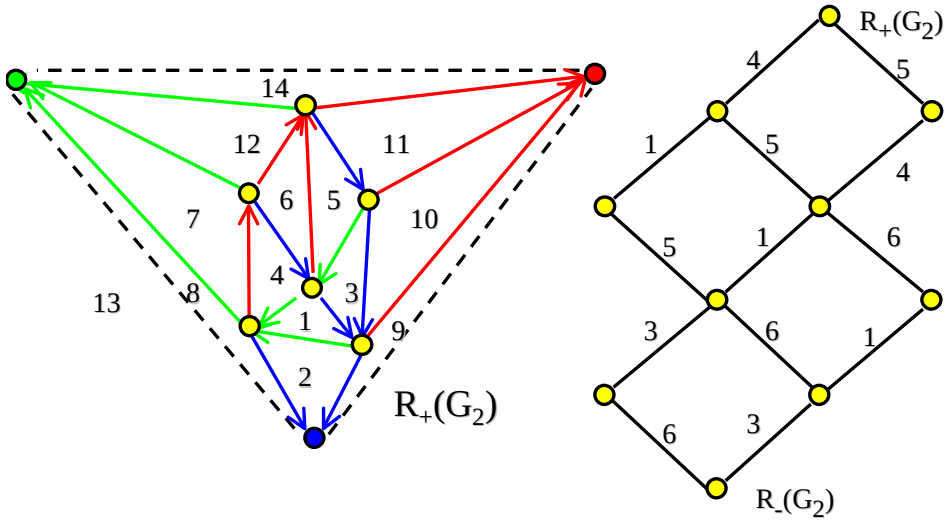

Figure 23 – Graphe de la figure 18 avec une face extérieure différente. Le treillis qui lui est associé possède le même nombre d'éléments que celui du graphe d'origine.

1.4 Généralisations des réaliseurs

1.4.1 Réaliseurs d'un graphe plan triconnexe

Définition 1.4.1. *(***Di Battista, Tamassia et Vismara***) [7]*
Soit G un graphe plan triconnexe. Un réaliseur R d'un graphe plan triconnexe G est un triplet $(\bar{T}_0, \bar{T}_1, \bar{T}_2)$ d'arbres recouvrants de G vérifiant les propriétés suivantes :

1. *Les racines v_0, v_1 et v_2 des 3 arbres recouvrants sont situées sur la face extérieure.*
2. *Chaque arête de G appartient à un ou deux des arbres recouvrants.*
3. *Si une arête (u, v) appartient à deux arbres recouvrants, alors si u est l'enfant de v dans le premier alors v est l'enfant de u dans le deuxième.*
4. *Considérons les arêtes de G avec les orientations qu'elles ont dans les trois arbres recouvrants (i.e. vers la racine), et lorsqu'une arête appartient à deux arbres elle est considérée deux fois.*

(a) *Chaque sommet de u, différent des trois racines, possède exactement trois arêtes sortantes. L'ordre circulaire des arêtes sortantes autour de u induit un ordre sur les arbres autour de u. Tous les sommets différents des trois racines, ont le même ordre circulaire pour les arbres recouvrants.*

(b) *Pour chaque sommet de G les arêtes entrantes qui appartiennent au même arbre recouvrant apparaissent de manière consécutive entre les arêtes sortantes des deux autres arbres recouvrants (la première et la dernière arête entrante pouvant coïncider avec les arêtes sortantes).*

5. *Toutes les arêtes adjacentes à la racine v_i de l'arbre T_i appartiennent à l'arbre T_i.*

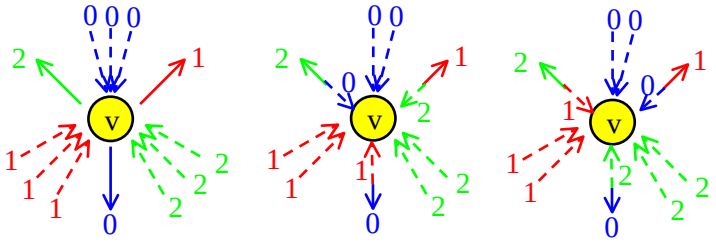

Figure 24 – Condition locale généralisée

La figure 25 représente un exemple de réaliseur triconnexe.

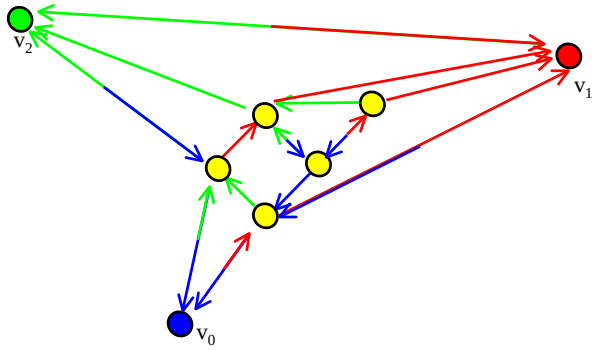

Figure 25 – Exemple de réaliseur triconnexe

Remarquons que si $R = (T_0, T_1, T_2)$ est un réaliseur d'un graphe plan maximal G, alors $R' = (\overline{T}_0, \overline{T}_1, \overline{T}_2)$ est un réaliseur de G au sens de la définition 1.4.1.

Cet objet permet d'obtenir des dessins convexes (i.e. les faces sont représentées par des polygones convexes) [7, 44]. Il permet également d'établir un schéma de routage tolérant aux pannes [7, 93]. Ezra Miller [75] et Felsner [45] quant à eux ont étudié les relations entre les réaliseurs des graphes triconnexes et certaines partitions planes. Plus récemment, Felsner a montré que l'ensemble des réaliseurs d'un graphe triconnexe avait également une structure de treillis distributif [46]. Ce dernier résultat généralise

aux réaliseurs des graphes plans triconnexes le résultat établi par de Mendez [79] pour les réaliseurs de graphes plans maximaux (voir théorème 1.3.1).

1.4.2 Arbres recouvrants ordonnés

Une autre généralisation des réaliseurs aux graphes planaires connexe a été proposé par Chiang, Lin et Lu [23].

Soit T un arbre recouvrant d'un graphe plan G. Deux sommets sont *non-apparentés* s'il ne sont pas ancêtres l'un de l'autre. Une arête de G est non-apparentée si elle relie deux sommets non-apparentés.

Définition 1.4.2. *(**Chiang, Lin et Lu, 2001**) [23]*
Soit $u_1, u_2, \ldots, u_n$ les sommets de G dans l'ordre préfixe anti-trigonométrique de T. Le sommet v_i est ordonné dans G considérant T si les arêtes adjacentes de v_i dans G se répartissent en 4 blocs (potentiellement vides) autour de v_i dans l'ordre anti-trigonométrique (voir figure 26) :

- *$B_P(v_i)$: arête vers le parent de v_i dans T ;*
- *$B_<(v_i)$: arêtes non-apparentées vers des sommets v_j avec $j < i$;*
- *$B_C(v_i)$: arêtes adjacentes aux enfants de v_i ; et*
- *$B_>(v_i)$: arêtes non-apparentées vers des sommets v_j avec $j > i$.*

La première arête de $B_>$ (resp. la dernière de $B_<$) est appelée arête-frontale (resp. arête-dorsale) Un arbre recouvrant T de G est ordonné si tous les sommets de H sont ordonnés. La paire (T, G) est appelé une paire ordonnée.

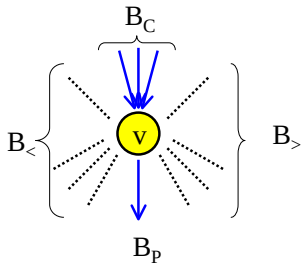

Figure 26 – Répartition des arêtes d'une paire ordonnée autour d'un sommet.

Théorème 1.4.1. *(**Chiang, Lin et Lu, 2001**) [23]*
Soit G_1 un graphe planaire. On peut calculer en temps linéaire un graphe plan G de G_1 et un arbre recouvrant T de G_1 tel que (T, G) soit une paire ordonnée.

Pour finir, remarquons que si $R = (T_0, T_1, T_2)$ est un réaliseur d'un graphe plan maximal G, alors $(\bar{T}_0, G)$, $(\bar{T}_1, G)$ et $(\bar{T}_2, G)$ sont des paires ordonnées.

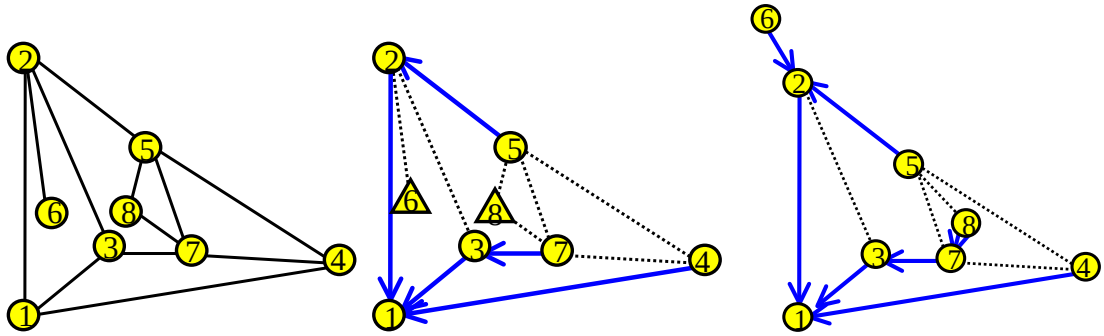

Figure 27 – Exemple de graphe. Le premier dessin n'admet pas d'arbre ordonné. Le second admet un arbre ordonné.

Première partie

Etude des réaliseurs

Chapitre 2

Extension du théorème de Wagner aux réaliseurs

Introduction

Jusqu'à présent nous avons considéré les réaliseurs d'un graphe donné. Dans ce chapitre, nous nous intéressons à l'ensemble des réaliseurs de l'ensemble des graphes plans maximaux de taille n. De ce fait, un réaliseur n'est plus défini comme un triplet d'arbres recouvrants, mais tout simplement comme un triplet d'arbres vérifiant certaines propriétés. Par la suite nous noterons l'ensemble des réaliseurs de taille n : $\mathcal{R}_n$.

Dans ce chapitre une généralisation du théorème de Wagner aux réaliseurs est présentée. Wagner [94] a montré que l'on peut transformer un graphe planaire maximal de taille n en n'importe quel autre de taille n uniquement en effectuant des transformations de diagonales, appelées *flips diagonaux* ou plus simplement *flips*. Un flip est une opération qui consiste à supprimer la diagonale (u_1, u_2) d'un quadrilatère (u_1, u_2, u_3, u_4) et la remplacer par la diagonale opposée (u_2, u_4) (voir figure 28). Pour que cette opération soit possible il est nécessaire que u_2 et u_3 ne soient pas adjacents. Dans le cas contraire, l'opération de flip créerait une arête double entre u_2 et u_3. Ainsi, on peut obtenir tous les graphes planaires maximaux de taille n par des flips diagonaux. Ce théorème a été étendu aux triangulations du plan projectif, du tore ainsi que de la bouteille de Klein [35, 78]. Plus récemment Gao, Urrutia et Wang [53] ont étudié les flips sur les graphes planaires maximaux étiquetés. Ils ont montré que la distance (en nombre de flips) entre deux graphes planaires maximaux étiquetés était $O(n \log(n))$.

Les flips sont également liés au théorème des 4 couleurs. Dans [38], des flips diagonaux signés (voir figure 29) ont été utilisés pour définir des transformations entre des triangulations signées d'un polygone. Il a été montré que l'existence d'une suite de flips signés entre deux triangulations d'un polygone est équivalente à l'existence d'une 4-coloration pour tous graphes planaires [38, 39, 61].

Nous proposons une extension du théorème de Wagner aux réaliseurs de taille n.

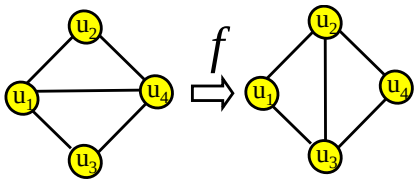

Figure 28 – Flip diagonal sur les graphes plans maximaux.

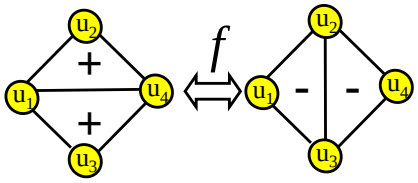

Figure 29 – Flip signé.

Pour ce faire, nous avons introduit deux nouvelles opérations : deux *flips diagonaux coloriés*. Nous avons montré que l'ensemble des réaliseurs de taille n est un *EPO*.

En utilisant ce résultat, nous avons aussi caractérisé le nombre de nœuds internes d'un réaliseur. Plus précisément, nous avons prouvé que $\xi_0+\xi_1+\xi_2-\Delta = n-1$ où ξ_j est le nombre de nœuds internes de l'arbre T_i et Δ est le nombre de faces tricolores. Comme application de cet invariant, nous trouvons qu'un arbre recouvrant ordonné [23], d'un graphe plan maximal, avec au plus $\lfloor \frac{2n+1-\Delta}{3} \rfloor$ feuilles peut être calculé en temps linéaire. Grâce à la relation entre le nombre de nœuds internes et le nombre de faces tricolores nous montrons que le nombre d'arêtes flippables d'un réaliseur est d'au moins $n-4-\Delta$.

La section 2.1 présente les flips diagonaux ainsi que le théorème de Wagner. Dans la section 2.2 nous présentons une version colorié des flips diagonaux sur les réaliseurs. Enfin, dans la dernière section 2.3 nous établissons des relations entre le nombre d'arêtes flippables, le nombre de nœuds internes ainsi que le nombre de faces tricolores d'un réaliseur.

2.1 Flips diagonaux et théorème de Wagner

Définition 2.1.1. *Soit G un graphe plan maximal. Soient u_2, u_1, u_4 et u_3, u_4, u_1 deux faces adjacentes, où u_2 n'est pas un voisin de u_3. Un flip diagonal est l'opération qui consiste à supprimer l'arête (u_1, u_4) et à ajouter l'arête (u_2, u_3) (voir figure 28).*

On peut remarquer que la condition u_2 n'est pas un voisin de u_3 est nécessaire pour éviter d'avoir des graphes avec des arêtes multiples.

Théorème 2.1.1. *(**Wagner 1936**) [94]*
Soient G_1 et G_2 deux graphes planaires maximaux, possédant n sommets. Il existe une séquence de flips diagonaux qui transforme G_1 en G_2.

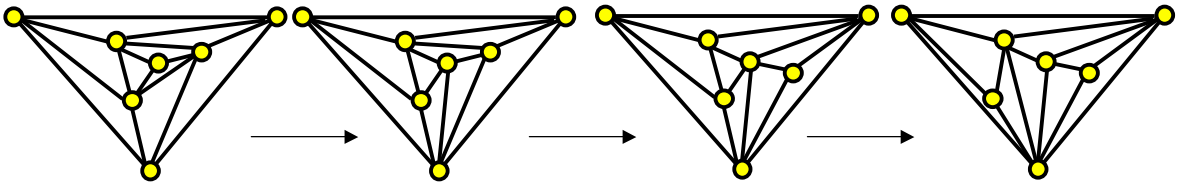

Figure 30 – Exemple de séquence de flips permettant de passer d'un graphe planaire maximal à un autre de même taille.

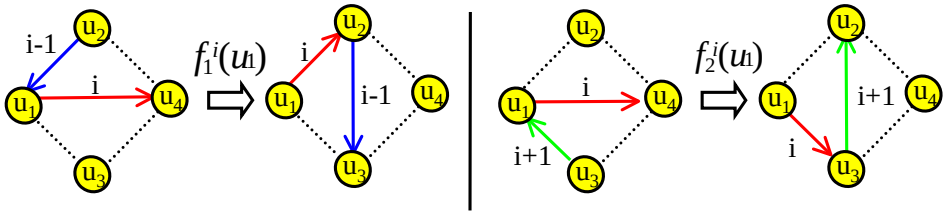

Figure 31 – Flips diagonaux sur les réaliseurs.

La figure 30 montre comment on peut passer d'un graphe planaire maximal à un autre.

2.2 Flips diagonaux sur les réaliseurs

Comme le montre la figure 31, nous proposons une version colorée des flips diagonaux pour les réaliseurs : f_1^i et f_2^i. On voit aisément que si l'on applique un flip diagonal de type f_1^i ou de type f_2^i, on obtient un nouveau réaliseur.

Le choix du type de flip que l'on peut appliquer sur une arête dépend de la configuration du quadrilatère. Remarquons que si l'arête (u_2, u_1) est coloriée $i-1$ et orientée vers u_1 et que l'arête (u_3, u_1) est coloriée $i+1$ et orientée vers u_1, alors on peut appliquer indifféremment le flip $f_1^i(u_1)$ ou le flip $f_2^i(u_1)$ (voir figure 32).

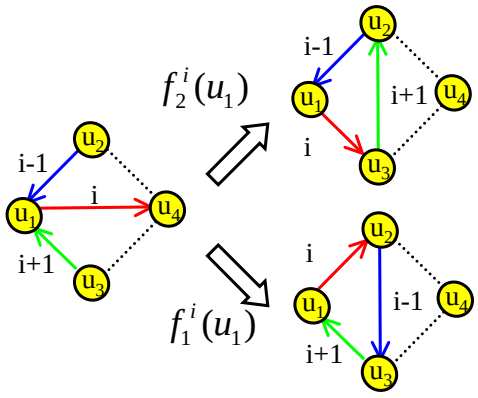

Figure 32 – Configuration où il est possible d'appliquer un flip $f_1^i(u_1)$ ou un flip $f_2^i(u_1)$.

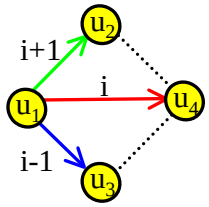

Figure 33 – Configuration où une arête ne peut pas être flippée.

En revanche, si l'arête (u_2, u_1) est coloriée $i+1$ et orientée vers u_2 et que l'arête (u_3, u_1) est coloriée $i-1$ et orientée vers u_3, l'arête (u_1, u_4) ne peut pas être flippée (voir figure 33). C'est ce qui rend le théorème de Wagner non trivial sur les réaliseurs.

Propriété 2.2.1. *Soit $R = (T_0, T_1, T_2)$ un réaliseur. Soit $R' = (T'_0, T'_1, T'_2)$ un réaliseur obtenu à partir de R en appliquant un flip de type f_1^i (resp. f_2^i). Les propriétés suivantes sont vérifiées : $T'_i <_{cw} T_i$, $T'_{i-1} >_{ccw} T_{i-1}$ (resp. $T'_i <_{ccw} T_i$, $T'_{i+1} >_{cw} T_{i+1}$)*

Démonstration. Considérons le flip $f_1^i(u_1)$ de la figure 31. L'arête (u_1, u_3) peut être coloriée $i-1$ et orientée vers u_3 ou coloriée $i+1$ et orientée vers u_1. Dans les deux cas, $u_1 <_{ccw}^{i-1} u_3$. Comme le nombre d'enfants de u_1 dans l'arbre T'_{i-1} est plus grand que dans l'arbre T_{i-1}, $T'_{i-1} >_{ccw} T_{i-1}$.

L'arête (u_2, u_4) peut être coloriée i et orientée vers u_4 ou coloriée $i+1$ et orientée vers u_2. Dans les deux cas, $u_4 <_{cw}^{i} u_2$. Comme le nombre d'enfants de u_1 dans l'arbre T'_i est plus petit que dans l'arbre T_i, $T'_i <_{cw} T_i$. □

2.2.1 Structure de $\mathcal{R}_n$ et théorème de Wagner

Soit $\mathcal{R}_n$ l'ensemble des réaliseurs des graphes de taille n. L'ensemble $\mathcal{R}_n$ peut être représenté par un graphe orienté dont chaque sommet représente un réaliseur et chaque arête (R, R') coloriée i signifie que R peut être transformé à l'aide d'un flip de type f_1^i. La figure 34 montre l'ensemble des réaliseurs de taille 6. Sur la partie droite de cette figure, on peut voir que l'on peut transformer le réaliseur 6 en le réaliseur 5 à l'aide un flip de type f_1^0.

Par la suite, on définit la relation $(f_1^i | f_1^{i+1})^*$ ainsi : on écrit que $R(f_1^i | f_1^{i+1})^* R'$ si R peut être transformé en R' par séquence de flips de type f_1^i et f_1^{i+1}. Soit $(\mathcal{R}_n, f_1^i | f_1^{i+1})$ l'ensemble des réaliseurs de taille n, équipé de la relation $(f_1^i | f_1^{i+1})^*$.

Lemme 2.2.1. *$(\mathcal{R}_n, f_1^i | f_1^{i+1})$ est un EPO.*

Démonstration. Pour montrer que $(\mathcal{R}_n, f_1^i | f_1^{i+1})$ est un EPO il faut montrer que la relation $(f_1^i | f_1^{i+1})^*$ est une relation d'ordre.

réflexivité : la séquence vide transforme R en R donc la relation est réflexive.

transitivité : soient R_1, R_2 et R_3 trois réaliseurs tels que $R_1(f_1^i | f_1^{i+1})^* R_2 (f_1^i | f_1^{i+1})^* R_3$. En concaténant les séquences qui transforment R_1 en R_2 et R_2 en R_3, on obtient une séquence de flips qui transforme R_1 en R_3. Donc $R_1(f_1^i | f_1^{i+1})^* R_3$.

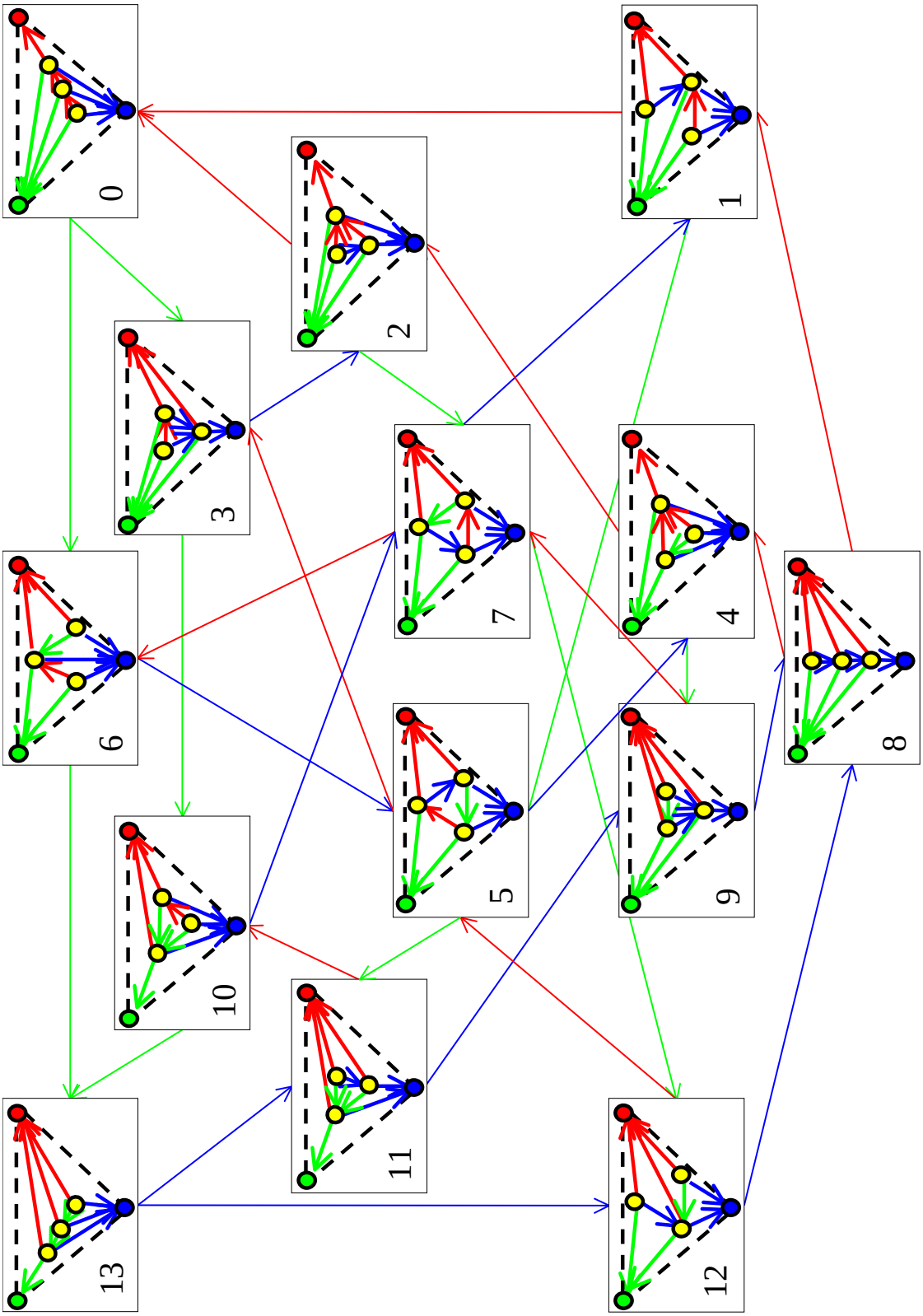

Figure 34 – L'ensemble $\mathcal{R}_6$ muni des opérations f_1.

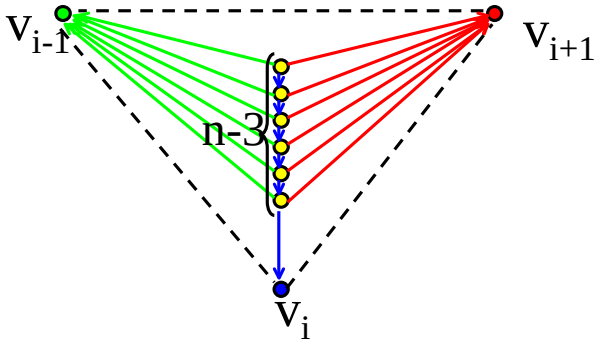

Figure 35 – Le réaliseur D_n^i.

anti-symétrie : soient $R = (T_0, T_1, T_2)$ et $R' = (T'_0, T'_1, T'_2)$ tels que $R(f_1^i|f_1^{i+1})^* R'(f_1^i|f_1^{i+1})^* R$. D'après la propriété 2.2.1, $T_i \geq_{cw} T'_i \geq_{cw} T_i$. Donc $T_i = T'_i$. Comme un réaliseur est entièrement défini par deux de ses arbres, $R = R'$. □

Soit D_n^i le réaliseur de taille n où tous les arbres T_{i-1} et T_{i+1} sont des arbres de profondeur 1 enracinés respectivement en v_{i-1} et v_{i+1} (voir figure 35).

Lemme 2.2.2. *D_n^{i-1} est la borne supérieure de $(\mathcal{R}_n, f_1^i|f_1^{i+1})$ et D_n^{i+1} la borne inférieure. Le réaliseur D_n^{i-1} est la borne inférieure de $(\mathcal{R}_n, f_2^{i-1}|f_2^i)$ et D_n^{i+1} la borne supérieure.*

Démonstration. Soit $R = (T_0, T_1, T_2)$ un réaliseur de taille n. Si $R \neq D_n^{i+1}$ alors soit T_i possède un nœud interne soit T_{i-1} possède un nœud interne. Si T_i possède un nœud interne u_1, alors on peut appliquer le flip $f_1^{i+1}(u_1)$. De manière similaire, si T_{i-1} possède un nœud interne u_2 alors, on peut appliquer le flip $f_1^i(u_2)$. Dans les deux cas, R n'est pas un élément maximal de $(\mathcal{R}_n, f_1^i|f_1^{i+1})$. Donc l'unique élément maximal de $(\mathcal{R}_n, f_1^i|f_1^{i+1})$ est D_n^{i-1}. Un raisonnement semblable montre que D_n^{i+1} est la borne inférieure de $(\mathcal{R}_n, f_1^i|f_1^{i+1})$ et que $(\mathcal{R}_n, f_2^{i-1}|f_2^i)$ est borné par D_n^{i-1} et par D_n^{i+1}. □

Lemme 2.2.3. *Il existe une séquence de flips qui transforme n'importe quel réaliseur en n'importe quel autre réaliseur de même taille.*

Démonstration. Soient R et R' deux réaliseurs de taille n. Comme D_n^{i-1} est l'unique élément maximum de $(\mathcal{R}_n, f_1^i|f_1^{i+1})$, il existe une séquence S_1 de flips de type f_1^i et f_1^{i+1} qui transforme R en D_n^{i-1}. Il existe aussi une séquence de flips de type f_1^i et f_1^{i+1} qui transforme R' en D_n^{i-1}. Comme l'inverse d'un flip de type f_1^i est un flip de type f_2^{i-1} et que l'inverse d'un flip de type f_1^{i+1} est un flip de type f_2^i, il existe une séquence S_2 composée de flips de type f_2^{i-1} et f_2^{i-1} qui transforme D_n^{i-1} en R'. Donc la concaténation des séquences de flips S_1 et S_2 transforme R en R'. □

Dans le précédent lemme, il était nécessaire d'utiliser des flips de type f_1 et de type f_2. Le théorème suivant utilise uniquement des flips de type f_1.

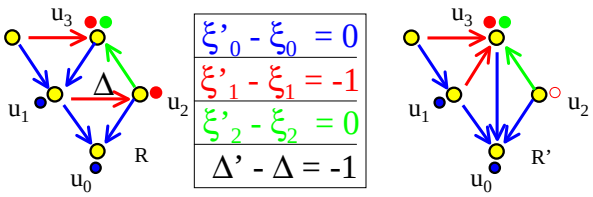

Figure 36 – Exemple de configuration de flip.

Théorème 2.2.1. *Soient R_1 et R_2 deux réaliseurs de taille n. Il existe une séquence de flips composée uniquement de flips de type f_1 qui transforme R_1 en R_2.*

Démonstration. Comme D_n^0 est la borne supérieure de $(\mathcal{R}_n, f_1^1|f_1^2)$, il existe une séquence S_1 de flips de type f_1^1 et f_1^2 qui transforme R en D_n^0. Comme D_n^0 est la borne supérieure de $(\mathcal{R}_n, f_1^1|f_1^2)$, il existe une séquence de flips de type f_2^1 et f_2^2 qui transforme R' en D_n^0. Donc il existe une séquence S_1 composée de flips de type f_1^0 et f_1^1 qui transforme D_n^0 en R'. La concaténation de S_1 et S_2 est une séquence composée de flips de type f_1^0, f_1^1 et f_1^2 qui transforme R en R'. □

2.3 Relations entre le nombre de faces tricolores, le nombre de nœuds internes et le nombre d'arêtes flippables

2.3.1 Nombre de nœuds internes

On note Δ le nombre de faces tricolores d'un réaliseur. On note également ξ_i le nombre de nœuds internes d'un arbre T_i.

Lemme 2.3.1. *Soit R un réaliseur. Soit R' obtenu en appliquant un flip de type f_1 sur le réaliseur R. La somme $\xi_0 + \xi_1 + \xi_2 - \Delta$ est identique pour les deux réaliseurs.*

Démonstration. Pour vérifier ce lemme, nous devons vérifier les différentes configurations de flips de type f_1^i sur le quadrilatère et la face adjacente à l'arête rentrante (u_1, u_3).

Si l'on considère la configuration de flip de la figure 36, nous pouvons voir que u_2 est un nœud interne de T_1 mais qu'il n'est pas un nœud interne de T_1'. De plus, R possède une face tricolore tandis que R' n'en possède pas. Donc le lemme est vérifié si l'on applique un flip de type f_1^i sur une telle configuration. Le tableau au centre de la figure 36 montre la différence du nombre de nœuds internes dans l'arbre T_0, T_1 et T_2 ainsi que la différence du nombre de faces tricolores.

Les 32 configurations possibles de flips de type f_1^i ont été également considérées (voir figure 37). Pour les 32 configurations le lemme est vérifié.

□

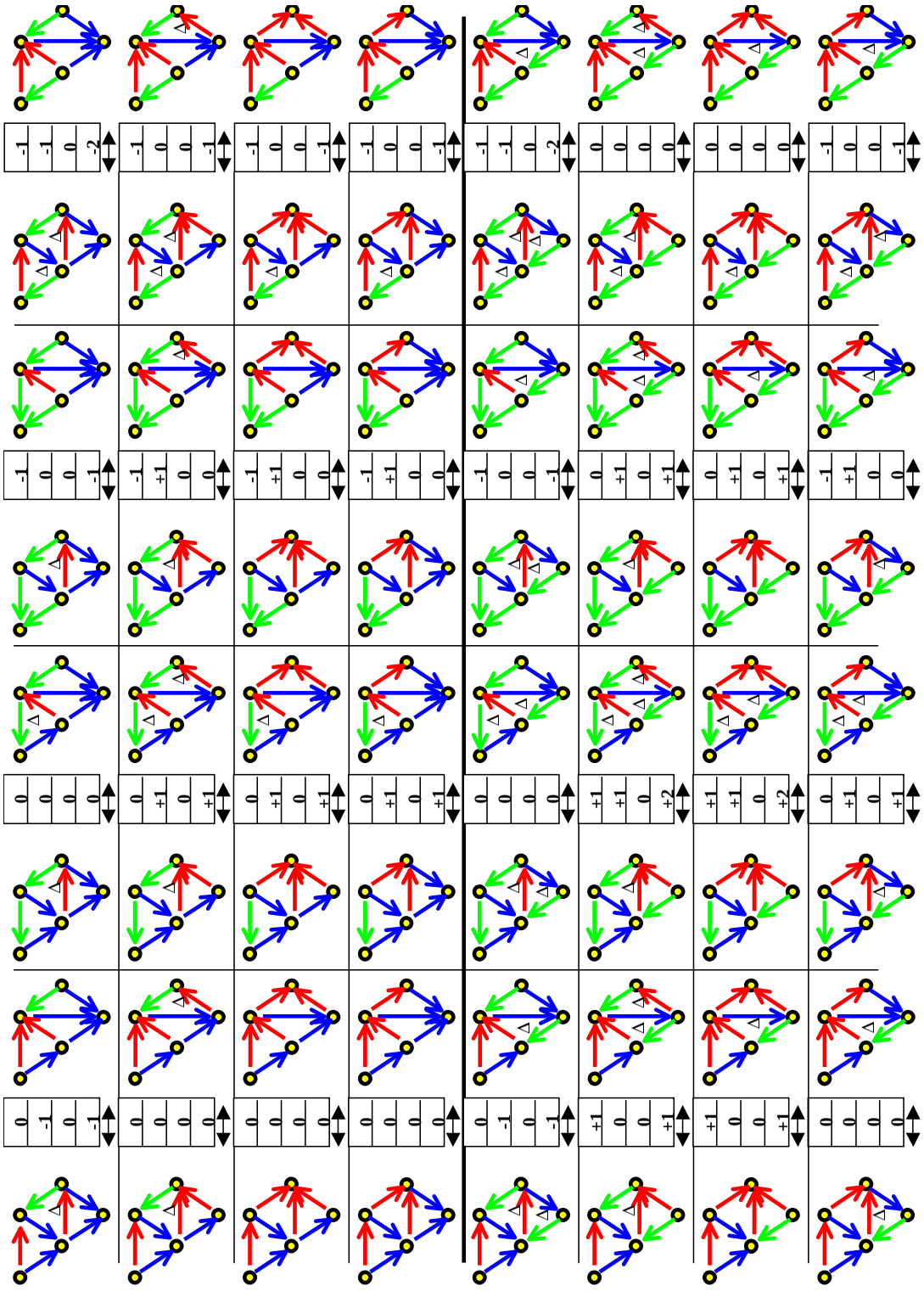

Figure 37 – 32 configurations de flips de type f_1.

Théorème 2.3.1. *Pour tout réaliseur $R(T_0, T_1, T_2)$ de taille n :*

$$\xi_0 + \xi_1 + \xi_2 - \Delta = n - 1.$$

Démonstration. Puisque tous les réaliseurs de taille n peuvent être obtenus à partir de D_n^0, il suffit d'évaluer cette somme sur ce dernier réaliseur. Le réaliseur D_n^0 possède $n-3$ nœuds internes dans l'arbre T_0, un nœud interne dans l'arbre T_1 et un nœud interne dans l'arbre T_2. De plus, ce réaliseur ne possède aucune face tricolore ($\Delta = 0$). Donc $\xi_0 + \xi_1 + \xi_2 - \Delta = n - 1$. □

Récemment, Fraysseix et Ossona de Mendez [50] m'ont communiqué une preuve plus simple de ce théorème :

Démonstration. On dit qu'un sommet est un *pôle d'une face* s'il possède deux arêtes sortantes sur cette face. Clairement chaque face bicolore possède un unique pôle et les faces tricolores ne possèdent pas de pôle. De plus les faces internes portant une arête externe possèdent chacune un unique pôle. Si u est une feuille de l'arbre T_i, alors u est le pôle de la face $(u, P_{i+1}(u), P_{i-1}(u))$. Ainsi le nombre de faces bicolores est égale au nombre de feuilles dans les arbres T_i. Le nombre de feuilles de l'arbre T_i vaut $n-2-\xi_i$. Comme les faces strictement-intérieures sont soit tricolores soit bicolores :

$$2n - 5 = \Delta + (n - 2 - \xi_0) + (n - 2 - \xi_1) + (n - 2 - \xi_2).$$

En simplifiant l'équation précédente on retrouve la formule du théorème 2.3.1. □

Un *arbre recouvrant ordonné* ("*orderly spanning tree*" en anglais) [23] d'un graphe plan maximal peut être obtenu à partir d'un arbre de Schnyder T_i en insérant l'arête (v_{i+1}, v_i) à T_i.

Corollaire 2.3.1. *Soit R un réaliseur d'un graphe plan maximal G. Un arbre recouvrant ordonné de G ayant au plus $\lfloor \frac{2n+1-\Delta}{3} \rfloor$ feuilles peut être obtenu en temps linéaire de R.*

Démonstration. Ce corollaire vient du fait que le nombre de feuilles d'un arbre de Schnyder T_i est $n-1-\xi_i$ et qu'un arbre recouvrant ordonné obtenu à partir de T_i possède deux feuilles de plus, le sommet v_{i+1} et le sommet v_{i-1}. Parmi les trois arbres de Schnyder, on prend l'arbre qui possède le moins de feuilles. Le théorème 2.3.1 nous assure que cet arbre possède au plus $\lfloor \frac{2n-5-\Delta}{3} \rfloor$. □

2.3.2 Nombre d'arêtes flippables d'un réaliseur

On dit qu'une arête est *flippable* si l'on peut appliquer un flip sur cette arête. Plus formellement, une arête (u_1, u_4) bordant deux faces (u_1, u_4, u_2) et (u_1, u_3, u_4) est flippable si et seulement si u_2 et u_3 ne sont pas adjacents.

Théorème 2.3.2. *(**Gao, Urrutia et Wang 2001**) [53]*
Tout graphe planaire maximal à n sommets possède au moins $n-2$ arêtes flippables.

Dans le cas des réaliseurs la définition de flippable peut être adaptée. Soit $e = (u_1, u_4)$ une arête coloriée i.

On dit que e est *f_1-flippable* si l'on peut appliquer l'opération $f_1^i(u_1)$. On dit que e est *flippable* si l'on peut appliquer l'opération $f_1^i(u_1)$ ou $f_2^i(u_1)$.

Théorème 2.3.3. *Soit R un réaliseur possédant Δ faces tricolores. Le nombre d'arêtes f_1-flippables de R est $n - 4 + \Delta$.*

Démonstration. Une arête (u_1, u_4) coloriée i est f_1-flippable si et seulement si u_1 est un nœud interne (autre que v_i) dans l'arbre T_{i-1}. Donc le nombre d'arêtes f_1-flippables coloriées 0 est $\xi_2 - 1$, le nombre d'arêtes f_1-flippables coloriées 1 est $\xi_0 - 1$ et le nombre d'arêtes f_1-flippables coloriées 2 est $\xi_1 - 1$. Le théorème 2.3.1 nous permet de conclure. □

2.4 Conclusion

Dans ce chapitre nous avons proposé une extension du théorème de Wagner aux réaliseurs. A l'aide de ce théorème nous avons établi une relation entre le nombre de nœuds internes des arbres d'un réaliseur et le nombre de faces tricolores. Cette dernière relation précise l'inégalité proposée dans [23].

De plus, nous avons montré que $(\mathcal{R}_n, f_1^i | f_1^{i+1})$ avait une structure d'EPO borné. Naturellement, nous nous sommes demandé si $(\mathcal{R}_n, f_1^i | f_1^{i+1})$ avait une structure de treillis distributif. Malheureusement, nous avons constaté que ni $(\mathcal{R}_6, f_1^i | f_1^{i+1})$, ni $(\mathcal{R}_n, f_1^i | f_1^{i+1} | f_2^{i+1})$, ni $(\mathcal{R}_n, f_1^i | f_2^{i+1})$ n'avaient une telle structure de treillis. Toutefois, une question reste en suspens : $(\mathcal{R}_n, f_1^i | f_1^{i+1} | f_2^{i+1})$ est-il également un EPO borné. Remarquons que cela est vrai pour $n \leq 6$.

Des questions restent encore à étudier : quelle est la distance (en termes de nombres de flips) qui sépare deux réaliseurs de taille n ? Gao, Urrutia et Wang [53] ont étudié les flips diagonaux sur les graphes plans étiquetés. Peut-on étendre leurs résultats aux réaliseurs étiquetés ? Si oui, quelle est la distance qui sépare deux réaliseurs étiquetés ? Quelle est la distance qui sépare deux réaliseurs qui ne différent l'un de l'autre que par leur étiquetage ? etc.

La notion de flip diagonal s'applique naturellement aux cartes dont les faces sont des triangles. L'opération de flip diagonal colorié peut s'exprimer sous la forme de deux *demi-flips* (voir figure 38) : $f_1^i(u_1) = fus_1^i(u_1) \circ sep_1^{i-1}(u_2)$ et $f_2^i(u_1) = fus_2^i(u_1) \circ sep_2^{i+1}(u_3)$. Nous pensons qu'une généralisation du théorème de Wagner aux réaliseurs de graphes triconnexes peut être proposée à l'aide des demi-flips. Cette généralisation serait la première qui s'appliquerait à des cartes dont les faces ne sont pas des triangles.

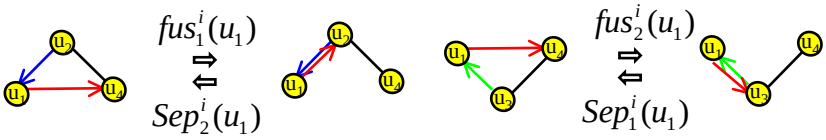

Figure 38 – Opération de demi-flip sur les réaliseurs des graphes triconnexes.

Chapitre 3

Bijection entre les réaliseurs et les paires de chemins de Dyck ne se coupant pas

Introduction

Un *chemin de Dyck* de longueur $2n$ est un chemin, composé de pas Nord-Est et Sud-Est de longueur $\sqrt{2}$, partant du point $(0,0)$ et se terminant au point $(2n,0)$ et restant dans le quart de plan positif. La paire (g,h) est une *paire de chemins de Dyck ne se coupant pas* si h reste dans la partie du quart de plan situé au-dessus du chemin g (voir figure 39). De tels chemins ont été étudiés entre autre par D. Gouyou-Beauchamps [57, 58]. Ils peuvent être vus comme un cas particulier des configurations étoilées [69] que nous considérerons plus en détail dans le chapitre suivant.

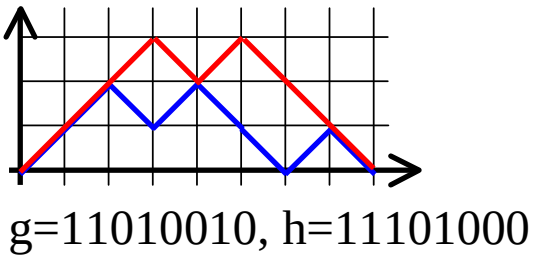

Figure 39 – Exemple de chemins de Dyck ne se coupant pas.

Dans ce chapitre nous établissons une bijection entre ces paires chemins et les réaliseurs. Cette bijection permet l'énumération des réaliseurs de taille n, ainsi que leur génération exhaustive. De plus, cette bijection nous permet de déduire que le nombre moyen de faces tricolores d'un réaliseur est asymptotiquement de $n/2+o(n)$. La génération aléatoire uniforme d'un réaliseur est aussi une application de cette bijection, et sera développée dans le chapitre suivant.

Le principe de la bijection est le suivant. A chaque réaliseur R on associe un réaliseur particulier R_c, appelé *réaliseur étoile*. Un réaliseur étoile est un réaliseur dont l'arbre T_2 est une étoile, i.e. dont tous les sommets sont des voisins du sommet v_2 (voir figure 40). Nous montrons qu'un réaliseur R est entièrement défini par son réaliseur étoile R_c muni d'une séquence particulière de flips, appelée *séquence préfixe de flips*, transformant R_c en R. Le réaliseur étoile et la séquence de flips peuvent être codés à l'aide d'une paire de chemins de Dyck ne se coupant pas.

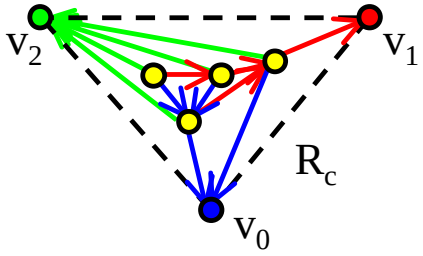

Figure 40 – Exemple de réaliseur étoile.

Le réaliseur étoile est entièrement défini par son arbre T_0. L'arbre T_0 est codé par le premier chemin de Dyck, le second codant la séquence de flips.

Ce chapitre est organisé de la manière suivante : la section 3.1 est une présentation des chemins de Dyck ne se coupant pas ainsi que de leur énumération, la section 3.2 introduit les réaliseurs étoiles ainsi que les séquences préfixes de flips, la section 3.3 présente une bijection entre les réaliseurs et les chemins de Dyck ne se coupant pas, et la section 3.4 propose quelques applications de cette bijection.

3.1 Chemins de Dyck qui ne se coupent pas

Soit un ensemble fini appelé *alphabet* où les éléments sont appelés *lettres*. Ici, nous utiliserons l'alphabet $A = \{1, 0\}$. Un *mot* f est une séquence finie de lettres $f_1 f_2 \dots f_n$. L'ensemble A^* de tous les mots sur l'alphabet est équipé de l'opération concaténation qui met bout à bout deux mots. La *longueur d'un mot*, notée $|f|$, est le nombre de lettres de f. Pour une lettre x, on note $|f|_x$ le nombre d'occurrences de x dans le mot f. Un mot f' est un *facteur gauche* de f s'il existe un mot f'' tel que $f = f'f''$. Un morphisme δ de A^* vers $\mathbb{N}$ est défini par $\delta(1) = 1$, $\delta(0) = -1$ et $\delta(f'f'') = \delta(f') + \delta(f'')$. Le *langage de Dyck* D est défini de la manière suivante : $D = \{f \in A^* | \delta(f) = 0 \text{ et } \forall f'$ facteur gauche de $f, \delta(f') \geq 0\}$. Nous notons $D_n = D \cap A^{2n}$. Nous notons également $open(k, f)$ la position de la k-ième "1" de f.

Un *chemin de Dyck* est un chemin codé par un mot de Dyck de la manière suivante : un pas Nord-Est est codé par un "1" et un pas Sud-est est codé par un "0". Ces chemins partent du point $(0, 0)$, ne traversent jamais l'axe des abscisses et terminent sur l'axe des abscisses. De manière classique, les mots de Dyck de longueur $2n - 2$ sont utilisés pour coder les arbres enracinés ordonnés de taille n. La figure 41 montre un arbre enraciné

ordonné ainsi que le mot de Dyck le codant. On appelle *pic* d'un chemin de Dyck, un pas Nord-Est suivit d'un pas Sud-Est. Dans un mot de Dyck, un pic correspond au motif 10. Si un arbre T est codé par un mot de Dyck f alors le nombre de feuilles de T correspond au nombre de pics de f. Dans la figure 41, on peut observer que l'arbre dessiné comporte trois feuilles et que le mot qui le code comporte trois pics.

Une paire (g, h) de $D_n \times D_n$ est une *paire de chemins de Dyck ne se coupant pas* si pour tout g' et h' respectivement facteurs gauches de g et h tels que $|g'| = |h'|$, alors $\delta(h') \geq \delta(g')$.

On note V_n l'ensemble des paires de chemins de Dyck de longueur $2n$ ne se coupant pas . De manière naturelle, une *paire de mots de Dyck ne se coupant pas* est une paire de mots de Dyck codant deux chemins de Dyck ne se coupant pas. La figure 39 montre un exemple de paire de Chemins de Dyck ne se coupant pas.

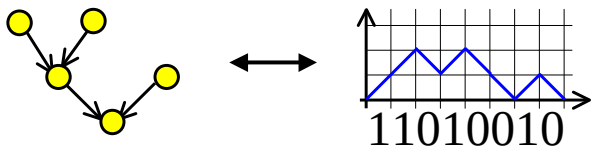

Figure 41 – Codage d'un arbre ordonné enraciné à l'aide d'un mot de Dyck.

Pour en terminer avec la présentation des paires de chemins de Dyck qui ne se coupent pas, rappelons que ces paires de chemins sont énumérées par une différence de produits de Catalan :

Théorème 3.1.1. *(**Gouyou-Beauchamps 1986**) [57]*
Le nombre de paires de chemins de Dyck qui ne se coupent pas de longueur $2n$ *est :*

$$|V_n| = C_{n+2}C_n - C_{n+1}^2,$$

où C_n *désigne le nombre de Catalan* $\frac{(2n)!}{n!(n+1)!}$.

Les premières valeurs de $|V_n|$ sont $1, 1, 3, 14, 84, 594, 4719, \ldots$.

3.2 Réaliseurs étoiles et séquences préfixes de flips

Dans cette section, nous présentons une classe particulière de réaliseurs, les *réaliseurs étoiles* ainsi qu'une manière canonique de transformer n'importe quel réaliseur en un réaliseur étoile. Cette transformation canonique est appelée *séquence préfixe de flips*.

3.2.1 Réaliseurs étoiles

Définition 3.2.1. *Un réaliseur étoile* $R_c = (T_0, T_1', E_{n-2})$ *est un réaliseur où* E_{n-2} *est une étoile de taille* $n-2$ *dont toutes les arêtes sont orientées vers le centre de l'étoile* v_2*, i.e.* E_{n-2} *est un arbre enraciné de profondeur* 1.

Dans le premier réaliseur de la figure 43, le sommet v_2 est voisin de tous les sommets internes du graphe. Donc ce réaliseur est un réaliseur étoile.

Propriété 3.2.1. *Soit T_0 un arbre ordonné enraciné de taille $n-2$. Il existe un unique arbre T_1' tel que $R_c = (T_0, T_1', E_{n-2})$ soit un réaliseur étoile.*

Démonstration. Tout d'abord, on peut remarquer qu'il y a une unique manière de connecter T_0 et E_{n-2} : l'ordre préfixe anti-trigonométrique de T_0 correspond à l'ordre trigonométrique autour de v_2. Une fois que T_0 et E_{n-2} sont connectés, nous obtenons un graphe plan. Soit $F_k = (v_2, u_i, u_{i_1}, u_{i_2}, \ldots, u_{i_t}, u_{i+1})$ une face de ce graphe plan (voir figure 42). Les parents dans T_1' des sommets $u_i, u_{i_1}, u_{i_2}, \ldots, u_{i_{t-1}}$ doivent être des sommets de cette face. C'est la seule manière de satisfaire la condition locale (voir définition 1.2.1). Pour la même raison, le seul sommet qui peut être le parent de $u_i, u_{i_1}, u_{i_2}, \ldots, u_{i_{t-1}}$ est le sommet u_{i+1}. Pour chaque sommet u_i, seul un sommet peut être le parent de u_k dans T_1' sans violer la condition locale de la définition d'un réaliseur. Donc il existe un seul arbre T_1' tel que $R_c = (T_0, T_1', E_{n-2})$ soit un réaliseur. □

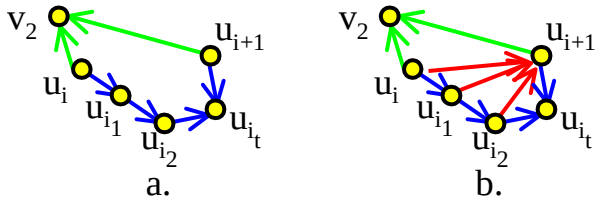

Figure 42 – **a.** Une face du graphe plan obtenue à partir de la connexion de T_0 et E_{n-2}. **b.** La même face, avec les arêtes de T_1' dedans.

Par la suite, on dira que $R_c = (T_0', T_1', E_{n-2})$ est le réaliseur étoile associé au réaliseur $R = (T_0, T_1, T_2)$ si $T_0 = T_0'$. Bien évidemment, le graphe sous-jacent à R_c n'est pas forcément le même que le graphe sous-jacent à R.

A partir de la construction précédente de l'arbre T_1', on peut déduire la propriété suivante.

Propriété 3.2.2. *Soit $R_c = (T_0, T_1', T_2)$ un réaliseur étoile. Soit G_c le graphe plan maximal associé à R_c. Soient $u_1, u_2 \ldots, u_{n-3}$ les sommets internes de G_c dans l'ordre préfixe anti-trigonométrique de T_0. Le nombre d'enfants de u_k dans T_1' est égal au nombre de sommets de la branche droite de son frère gauche dans T_0.*

Par la suite, nous utiliserons cette propriété pour calculer le nombre de flips diagonaux que l'on peut appliquer sur un sommet d'un réaliseur.

3.2.2 Séquence préfixe de flips

Définition 3.2.2. *Soit $R_c = (T_0, T_1', E_{n-2})$ un réaliseur étoile. Une séquence préfixe de flips, ou SPF, est une séquence de flips $(f_1^2(u_1), f_1^2(u_2), \ldots, f_1^2(u_p))$ qui peut être*

appliquée à R_c tel que pour tout i, j : $i < j \Rightarrow u_i \leq_{cw} u_j$.

Par la suite, une SPF sera représentée par une liste de $n-2$ nombres, spécifiant le nombre de flips à appliquer sur chaque sommet interne du réaliseur. Par exemple, la SPF $(f_1^2(u_3), f_1^2(u_4), f_1^2(u_4))$ est représentée par la suite $(0, 0, 1, 2)$. On notera $\#f(u_k)$ le nombre de flips de la SPF sur le sommet u_k. Dans la séquence précédente on observe que $\#f(u_3) = 1$ et que $\#f(u_4) = 2$. La figure 43 montre la SPF précédente appliquée sur un réaliseur étoile.

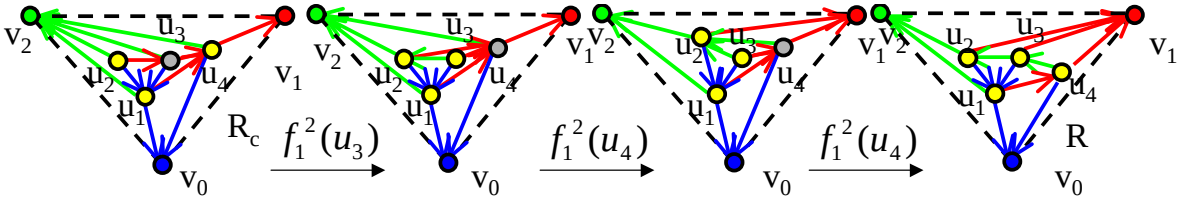

Figure 43 – Exemple de séquence préfixe de flips : $(0, 0, 1, 2)$.

Remarquons qu'une SPF ne modifie pas l'arbre T_0 du réaliseur étoile. Dans la suite de cette section nous noterons $Ch'_i(u)$ les enfants de u dans l'arbre T'_i.

Propriété 3.2.3. *Soit $R_c = (T_0, T'_1, E_{n-2})$ un réaliseur étoile. Soient S une SPF et $R = (T_0, T_1, T_2)$ le réaliseur obtenu en appliquant S à R_c. Pour chaque sommet de T_0, nous avons : $|Ch_1(u_k)| = |Ch'_1(u_k)| + \#f(u_{k-1}) - \#f(u_k)$*

Démonstration. La propriété peut être reformulée de la manière suivante : quand on applique un flip sur un sommet u_k dans la séquence S, $|Ch_1(u_k)|$ est décrémenté et $|Ch_1(u_{k+1})|$ est incrémenté.

De manière évidente, quand un flip $f_1^2(u_k)$ est appliqué, $|Ch_1(u_k)|$ est décrémenté. Montrons donc que lorsqu'on applique un flip $f_1^2(u_k)$, $|Ch_1(u_{k+1})|$ est incrémenté. Pour cela montrons par induction sur k que lorsqu'un flip peut être appliqué sur u_k, u_{k+1} se trouve après (dans l'ordre anti-trigonométrique) $P_2(u_k)$ dans la liste d'adjacence de u_k.

Tout d'abord, remarquons que dans un réaliseur étoile, u_{k+1} est juste après $P_2(u_k) = v_2$ dans la liste d'adjacence de u_k.

Supposons qu'après avoir appliqué les flips de S sur les $k-1$ premiers sommets de T_0, u_{i+1} est juste après $P_2(u_i)$ dans la liste d'adjacence de u_i pour tout $i \geq k$. Après l'application de $f_1^2(u_k)$, u_{k+1} est toujours juste après $P_2(u_k)$ dans la liste d'adjacence u_k (voir figure 31). Donc lorsqu'on applique les $\#f(u_k)$ flips sur u_k, u_{k+1} est juste après $P_2(u_k)$ dans la liste d'adjacence de u_k.

De plus, les modifications faites par les flips $f_1^2(u_k)$ sont restreintes à la région délimitée par $(v_2, u_{k+1}, u_{k+1} \underset{\rightarrow}{0} v_0)$. Donc u_{k+2} est juste après $P_2(u_{k+1})$ dans la liste d'adjacence de u_{k+1} et pour chaque $i > k+1$, la liste d'adjacence de u_i n'est pas changée.

Donc dans une séquence préfixe, à chaque fois qu'un flip $f_1^2(u_k)$ est effectué, le nombre d'enfants de u_k dans T_1 est décrémenté et le nombre d'enfants de u_{k+1} dans T_1 est incrémenté. □

La propriété 3.2.3 peut aussi s'exprimer de la manière suivante :

$$\#f(u_k) = |Ch'_1(u_k)| + \#f(u_{k-1}) - |Ch_1(u_k)|.$$

Lemme 3.2.1. *Soient $R = (T_0, T_1, T_2)$ un réaliseur et $R_c = (T_0, T'_1, E_{n-2})$ son réaliseur étoile associé. Il existe une unique SPF $S_c w$ qui transforme R_c en R.*

Démonstration. Existence : Soit R un réaliseur. Considérons l'algorithme suivant :

pour chaque sommet u_k dans l'ordre préfixe anti-trigonométrique de T_0 **faire**
  tant que u_k n'est pas un voisin de v_2 **faire**
    Effectuer le flip $f_2^1(P_2(u_k))$
  fin tant que
fin pour

On ne peut pas opérer un nombre infini de fois le flip $f_2^1(P_2(u_k))$. Donc l'algorithme termine. Quand l'algorithme termine, un réaliseur étoile est obtenu, puisque tous les sommets internes de G sont voisins de v_2. L'inverse d'un flip $f_2^1(P_2(u_k))$ est un flip $f_1^2(u_k)$ (voir figure 31). La séquence inverse de la séquence de flips construite par le précédent algorithme est une séquence préfixe. Donc, pour chaque réaliseur R, il existe une séquence préfixe de flips qui transforme R_c en R.

Unicité : Deux réaliseurs ayant des réaliseurs étoiles différents ne peuvent être identiques puisqu'il n'ont pas le même arbre T_0. Soient S_{cw1} et S_{cw2} deux SPF. Soient R_1 (resp. R_2) le réaliseur obtenu en appliquant S_{cw1} (resp. S_{cw1}) au réaliseur R_c. Soit k le plus petit indice pour lequel les deux séquences n'appliquent pas le même nombre de flips sur le sommet u_k. La propriété 3.2.3 nous dit que $|Ch_1(u_k)|$ dans R_1 est différent de $|Ch_1(u_k)|$ dans R_2. Donc si l'on applique deux SPF différentes sur un réaliseur étoile R_c, on obtient deux réaliseurs différents. □

Considérons la séquence de flips $(f_1^2(u_4), f_1^2(u_3))$ (voir figure 44). Cette séquence (qui n'est pas une SPF) appliquée sur le réaliseur étoile R_c de la figure 43, donne le même réaliseur R, que celui obtenu par la séquence préfixe de flips $(0, 0, 1, 2)$. Bien

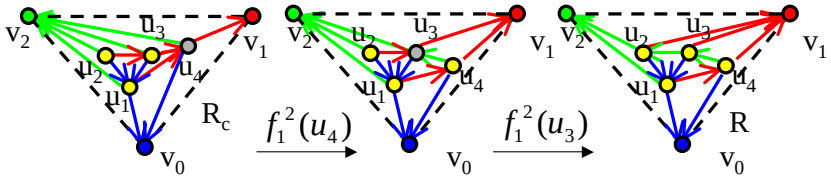

Figure 44 – Séquence non-préfixe de flips $(f_1^2(u_4), f_1^2(u_3))$.

qu'il existe plusieurs séquences de flips transformant R_c en R, il existe une seule SPF transformant R_c en R.

Propriété 3.2.4. *Soient $R = (T_0, T_1, T_2)$ et $R' = (T_0, T_1, T'_2)$ deux réaliseurs. Alors $T_2 = T'_2$.*

Démonstration. Nous avons déjà vu que la propriété était vraie pour les réaliseurs étoiles (voir propriété 3.2.1). Montrons par l'absurde qu'elle est vraie pour tous les réaliseurs. Supposons donc que $T_2 \neq T_2'$. Les réaliseurs R et R' partagent le même réaliseur étoile R_c, mais ont deux SPF différentes S et S'. Soit u_k, le premier sommet dans l'ordre anti-trigonométrique de T_0 tel que $\#f(u_k) \neq \#f'(u_k)$. D'après la propriété 3.2.2, ceci implique $|Ch_1(u_k)|$ dans R est différent de $|Ch_1(u_k)|$ dans R'. Ceci est en contradiction avec le fait que R et R' partagent le même arbre T_1. □

3.3 Algorithmes de codage et décodage d'un réaliseur à l'aide de mots de Dyck

Dans cette section nous présentons une bijection entre les réaliseurs de taille n et les paires de chemins de Dyck qui ne se coupent pas de longueur $2n-6$. Pour démontrer ce résultat nous présentons un algorithme qui code un réaliseur de taille n en une paire de chemins de Dyck de longueur $2n-6$, puis un algorithme de décodage. Ces deux algorithmes ont une complexité linéaire.

3.3.1 Algorithme de codage

Algorithme 1 algorithme de codage.

Construire le réaliseur étoile R_c associé à R
Coder l'arbre T_0 avec un mot de Dyck g.
$h \leftarrow g$
pour chaque sommet u_k dans l'ordre préfixe anti-trigonométrique de T_0 **faire**
  $\#f(u_k) \leftarrow |Ch_1'(u_k)| - |Ch_1(u_k)| + \#f(u_{k-1})$
  Déplacer $open(k,h)$ de $\#f(u_k)$ rang vers la gauche h.
fin pour

Propriété 3.3.1. *Dans l'algorithme 1, le nombre de flips appliqués sur le sommet u_k est inférieur ou égal au nombre de "0" consécutifs qui précèdent $open(k,h)$ dans h.*

Démonstration. Lorsqu'aucun flip n'a été effectué, $h = g$. Le nombre de "0" consécutifs juste avant $open(k,h)$ dans h est exactement le nombre de sommets de la branche droite de son frère gauche. $|Ch_1'(u_k)|$ est égal au nombre de sommets de la branche droite de son frère gauche (voir propriété 3.2.2). Comme $\#f(u_k) \leq |Ch_1'(u_k)|$, la propriété est satisfaite.

Supposons maintenant que la propriété soit vérifiée pour $i \leq k-1$. Le nombre de "0" consécutifs juste avant $open(k,h)$ est $|Ch_1'(u_k)| + \#f(u_{k-1})$. Comme $\#f(u_k) \leq |Ch_1'(u_k)| + \#f(u_{k-1})$ (voir propriété 3.2.3), la propriété est aussi vérifiée pour $i = k$. □

Lemme 3.3.1. *L'algorithme 1 code un réaliseur R de taille n avec une paire de chemins de Dyck ne se coupant pas de longueur $2n-6$. De plus, cet algorithme est linéaire.*

Démonstration. On peut remarquer dans un premier temps que les mots g et h produits par l'algorithme sont des mots de Dyck qui ne se coupent pas.

Montrons donc que l'algorithme définit une fonction injective. C'est à dire que deux réaliseurs différents sont codés par deux paires de chemins différentes. Soit $R = (T_0, T_1, T_2)$ et $R' = (T'_0, T'_1, T'_2)$ deux réaliseurs différents. Soit (g, h) (resp. (g', h')) la paire de chemins de Dyck ne se coupant pas obtenue à partir de l'algorithme précédent depuis le réaliseur R (resp. R'). Si $T_0 \neq T'_0$ alors g est différent de g'. Soit S_f (resp. S'_f) la séquence préfixe de flips associée au réaliseur R (resp. R'). Soit k le premier indice tel que $\#f(u_k) \neq \#f'(u_k)$. Après le k-ième passage dans la boucle, $open(k, h) \neq open(k, h')$. Durant le reste de l'exécution de l'algorithme, $open(k, h)$ et $open(k, h')$ ne seront pas changés, donc $h \neq h'$. En conséquence, pour deux réaliseurs différents on construit deux paires de chemins de Dyck ne se coupant pas différentes. L'algorithme 1 est un algorithme de codage des réaliseurs.

Complexité : pour la construction du réaliseur étoile, chaque sommet de T_0 est connecté avec une arête sortante dans T_2 vers le sommet v_2 et avec des arêtes entrantes dans T_1 depuis tous les sommets de la branche droite de son frère gauche. Une telle construction peut être réalisée en temps linéaire. Le codage de l'arbre T_0 s'effectue de manière classique en temps linéaire. Le traitement de chaque sommet u_k consiste à calculer le nombre de flips sur le sommet u_k et à intervertir deux lettres dans le mot h. Ceci s'effectue en temps constant. Au final, l'algorithme est linéaire. □

Pour coder le réaliseur R de la figure 43, il faut coder son réaliseur étoile R_c associé ainsi que la séquence préfixe de flips : $(0, 0, 1, 2)$. L'arbre T_0 de R est celui représenté par la figure 41. Le mot de Dyck qui lui est associé est $g = 11010010$. Pour coder la SPF, nous devons déplacer le troisième "1" d'un pas vers la gauche et le quatrième "1" de deux pas vers la gauche pour obtenir le deuxième mot : $h = 11101000$. Le réaliseur R de la figure 43 est donc codé par la paire de chemins de Dyck ne se coupant pas (g, h).

3.3.2 Algorithme de décodage

Soit (g, h) une paire de chemins de Dyck ne se coupant pas.

Pour détailler l'algorithme de décodage d'un réaliseur, définissons au préalable quelques fonctions élémentaires. La fonction *concat(L_1, L_2)* ajoute à la fin de la liste L_1 tous les éléments de la liste L_2 et la liste résultante est renvoyée par cette fonction. La fonction *Split(L, i)* supprime les i derniers éléments de la liste L et renvoie une liste contenant ces i éléments. La procédure *AddFirst(L, e)* ajoute l'élément e au début de la liste L. Naturellement, *Del(L, i)* supprime le i-ième élément de la liste L.

Lemme 3.3.2. *L'algorithme 2 calcule en temps linéaire un réaliseur R de taille n à partir d'une paire de chemins de Dyck ne se coupant pas de longueur $2n-6$.*

Algorithme 2 Algorithme de décodage.

Construire l'arbre T_0 associé à g
Construire le réaliseur étoile $R_c = (T_0, T_1', E_{n-2})$
$R = (T_0, T_1, T_2) \leftarrow R_c$
pour chaque sommet u_k dans l'ordre préfixe anti-trigonométrique de T_0 **faire**
  $\#f(u_k) \leftarrow open(g,k) - open(h,k)$
  $L \leftarrow Split(Ch_1(u_k), \#f(u_k))$
  $Ch_1(u_{k+1}) \leftarrow Concat(Ch_1(u_{k+1}), L)$
  $Del(Ch_2(P_2(u_k)), u_k)$
  $AddFirst(Ch_2(Ch_1(u_{k+1}, 0)), u_k)$
fin pour

Démonstration. Validité : Comme $h \geq g$ alors $0 \leq \#f(u_k) \leq |Ch_1'(u_k)| + \#f(u_{k-1})$ code un réaliseur étoile ainsi qu'une SPF valide. De plus, l'algorithme 2 construit le réaliseur codé par l'algorithme 1.

Complexité : comme dans l'algorithme de codage, la construction du réaliseur étoile s'effectue en temps linéaire. L'algorithme utilise des listes chaînées pour stocker la liste des enfants de chaque sommet dans chacun des arbres du réaliseur. L'opération *Split* s'effectue en $O(|Ch_1(u_k)|)$ opérations élémentaires. Globalement, les opérations *Split* s'effectue en $O(m) = O(n)$ opérations élémentaires. Les autres instructions de la boucle prennent $O(1)$ opérations. Donc, globalement l'algorithme est linéaire. □

Le théorème suivant découle directement des lemmes 3.3.1 et 3.3.2 :

Théorème 3.3.1. *L'ensemble des réaliseurs de taille n est en bijection avec l'ensemble des paires de chemins de Dyck qui ne se coupent pas de longueur* $2n-6$.

Corollaire 3.3.1. *Le nombre de réaliseurs de taille n est :*

$$|\mathcal{R}_n| = |V_{n-3}| = C_{n-3}C_{n-1} - C_{n-2}^2.$$

3.4 Comportement asymptotique des réaliseurs

Théorème 3.4.1. *Asymptotiquement, le nombre de réaliseurs de taille n est :*

$$|R_n| \sim \left(\frac{3}{512\pi n^5}\right) 2^{4n}.$$

Démonstration. Le développement asymptotique des nombres de Catalan est le suivant :

$$C_n = \frac{4^n}{\sqrt{\pi n^3}}\left(1 - \frac{9}{8n} + \frac{145}{128n^2} + O(\frac{1}{n^3})\right)$$

En utilisant cette expression de C_n dans la formule de $|R_n|$, on obtient le résultat. □

Le précédent théorème nous permet d'affirmer qu'il faut au moins $4n + O(\log(n))$ bits pour coder un réaliseur de taille n. Donc l'algorithme de codage présenté ici ainsi que celui présenté dans [26] sont optimaux en nombre de bits.

Le nombre de graphes plans maximaux de taille n étant le suivant [91] :

$$T_n = \frac{2(4n-11)!}{(n-2)!(3n-7)!},$$

nous pouvons évaluer le nombre moyen de réaliseurs d'un graphe plan maximal :

$$\frac{|R_n|}{|T_n|} = \frac{3(n-1)(3n-7)!(2n-4)!^2}{2n(2n-5)(4n-11)!(n-1)!^4} = 2^{(4-3\log_2(3))n+o(n)} \approx 2^{0,759n}.$$

Théorème 3.4.2. *Le nombre moyen de nœuds internes $\overline{\xi_i}$ d'un arbre T_i d'un réaliseur est :*

$$\overline{\xi_i} = n/2 + o(n).$$

Démonstration. Comme nous l'avons vu, le chemin du bas code l'arbre T_0. De plus, le nombre de nœuds internes de T_0 est égal à $n-k$ où k correspond au nombre de pics du chemin du bas. Nous allons donc montrer que le nombre moyen de pics du chemin du bas d'une paire de chemins de longueur $2n$ est $n/2 + o(n)$.

Soit $V_n(k)$ l'ensemble des paires de chemins de Dyck de longueur $2n$ dont le premier chemin possède k pics. Soit $\bar{k}$ le nombre moyen de pics du chemin du bas dans une paire de chemins de Dyck :

$$\bar{k} = \frac{\sum_{k=1}^{n-1} k\, V_n(k)}{V_n}$$

Soit V'_n l'ensemble des paires de chemins de Dyck dont le chemin du bas possède entre $n/2 - n^{3/4}$ et $n/2 + n^{3/4}$ pics.

En considérant séparément les éléments de V'_n et ceux de $V_n \setminus V'_n$, nous pouvons borner $\bar{k}$:

$$(n/2 - n^{3/4})\frac{|V'_n|}{|V_n|} \leq \bar{k} \leq (n/2 + n^{3/4})\frac{|V'_n|}{|V_n|} + n\frac{|V_n \setminus V'_n|}{|V_n|}$$

Le nombre de chemins de Dyck de longueur $2n$ avec k pics est donné par les nombres de Narayana [77] $u(n,k)$:

$$u(n,k) = \frac{1}{n+1}\binom{n+1}{k}\binom{n-1}{n-k}.$$

Il est clair que le nombre de paires de chemins qui ne se coupent pas dont le premier possède k pics est plus petit que le nombre de paires de chemins (se coupant ou pas) dont le premier possède k pics :

$$V_n(k) \leq u(n,k).C_n$$

Comme

$$\frac{|V'_n|}{|V_n|} \geq \frac{u(n, n/2 + n^{3/4}).C_n}{V_n} = 1 - \frac{O(n^{3/2})}{e^{4\sqrt{n}}}$$

nous pouvons borner $\bar{k}$ de la manière suivante :

$$(n/2 - n^{3/4})(1 - o(1)) \leq \bar{k} \leq (n/2 + n^{3/4}) + no(1)$$

d'où $\bar{k} = n/2 + o(n)$. □

Théorème 3.4.3. *Le nombre moyen de faces tricolores $\overline{\Delta}$ d'un réaliseur de taille n est :*

$$\overline{\Delta} = n/2 + o(n).$$

Démonstration. Utilisons le même raisonnement que pour la preuve du théorème 3.4.2, non plus sur T_0, mais sur l'arbre T_i qui possède le moins de nœuds internes.

En effet, on peut coder un réaliseur avec 2 arbres. Ensuite, on indique quel est l'arbre du réaliseur utilisé pour le codage (3 possibilités).

La même argumentation nous montre que le nombre moyen de nœuds internes, noté $\overline{\xi_{min}}$, de l'arbre T_i possédant le moins de nœuds internes parmi T_0, T_1 et T_2 tend vers $n/2 + O(\sqrt{n})$. De la même manière, le nombre moyen de nœuds internes de l'arbre T'_i possédant le moins de nœuds internes parmi T_0, T_1, T_2, noté $\overline{\xi_{max}}$, tend aussi vers $n/2 + O(\sqrt{n})$.

D'après la formule du théorème de Wagner sur les réaliseurs (théorème 2.3.1), pour n assez grand :

$$3\overline{\xi_{min}} - n \geq \overline{\Delta} \geq 3\overline{\xi_{max}} - n$$

Donc

$$\overline{\Delta} = \frac{n}{2} + o(n)$$

□

Corollaire 3.4.1. *Le nombre moyen de cw-faces $\overline{\Delta}_+$ et de ccw-faces $\overline{\Delta}_-$ d'un réaliseur de taille n est :*

$$\overline{\Delta}_+ = \overline{\Delta}_- = n/4 + o(n).$$

Une face strictement intérieure possède 8 colorations possibles (Voir figure 13). Le nombre de faces intérieures d'un graphe plan maximal est $2n - 8$. Pour n assez grand, un quart de ces faces est tricolores et trois quarts est bicolore. Par symétrie, les deux configurations tricolores sont équiprobables et les trois configurations bicolores sont équiprobables. Donc, quand n tend vers l'infini, chaque probabilité d'une coloration de face tend vers 1/8.

3.5 Conclusion

Dans ce chapitre nous avons proposé un algorithme de codage des réaliseurs en $4n$ bits s'appuyant sur une bijection. Cette bijection nous assure que le nombre de bits utilisés pour le codage est optimal. Comme le nombre de graphes plans maximaux est $2^{3,24n+o(n)}$, nous en déduisons que le nombre moyen de réaliseurs est de $2^{0,759n+o(n)}$. Cette dernière observation exclut tout espoir d'obtenir un algorithme à rejet de génération aléatoire de graphes plans maximaux utilisant les réaliseurs. En effet, avec un tel algorithme il faudrait un nombre exponentiel de tirages pour générer un graphe plan maximal.

Les symétries des réaliseurs peuvent être utilisées pour déduire des propriétés sur les paires de chemins de Dyck qui ne se coupent pas. Par exemple, dans le codage proposé, le nombre de contacts du premier chemin avec l'axe des abscisses correspond au degré de la racine de l'arbre T_0. De plus, le nombre de pas montants (resp. descendants) où les deux chemins sont en contact, correspond au degré de la racine de l'arbre T_2 (resp. T_1). Par permutation circulaire des trois arbres, on peut obtenir un autre codage du réaliseur. Dans ce nouveau codage, le degré de la racine de T_2 est égale au nombre de contacts avec l'axe des abscisses et le nombre de contacts entre deux pas descendants correspond au degré de la racine T_0. Donc ces trois statistiques (le nombre de contacts avec l'axe des abscisses, le nombre de superposition de pas montants et le nombre de superposition des pas descendants) sur les paires de chemins de Dyck qui ne se coupent pas suivent la même loi. Nous pensons que d'autres statistiques pourraient être mises en relation en utilisant les transformations appropriées sur les réaliseurs.

Chapitre 4

Génération aléatoire de pastèques

Introduction

Le modèle des *promeneurs méchants* décrit la situation où p promeneurs avancent simultanément d'un pas (Nord-Est ou Sud-Est) et ne partagent jamais la même position. Ce problème a été introduit par Fisher [47].

La trajectoire du i-ième promeneur est appelée i-ième *branche* de la pastèque. Dans ce chapitre nous allons considérer uniquement le cas où les p promeneurs partent respectivement des points $(0,0),(0,2),\ldots,(0,2p-2)$ et effectuent chacun l pas. L'ensemble des p branches de p promeneurs donnés est appelé *étoile*. Lorsque les p promeneurs arrivent respectivement en $(l,d),(l,d+2),\ldots,(l,d+2p-2)$ on parle alors de *pastèque* ("*watermelon*" en anglais) de *longueur* l et de *déviation* d. Une étoile (ou pastèque) est dite *avec mur* si aucune de ses branches ne traverse l'axe des abscisses. La figure 45 illustre ces définitions.

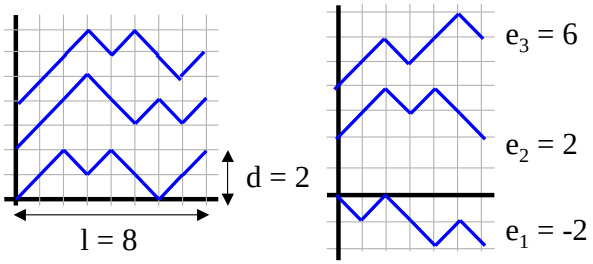

Figure 45 – (a) Une pastèque avec mur, 3 promeneurs ayant des trajectoires de longueur 8 et de déviation 2. (b) Une configuration étoile sans mur, avec 3 promeneurs ayant des trajectoires de longueur 6 et terminant respectivement aux ordonnées $(-2,2,6)$.

Les pastèques de déviation nulle sont en bijection directe avec les chemins de Dyck (ou chemins de Grand Dyck) qui ne se coupent pas. En effet, pour passer de p chemins de Dyck (ou Grand Dyck) qui ne se coupent pas à une pastèque de déviation nulle, il suffit de décaler le i-ième chemin de $2i$ pas vers le Nord.

De nombreux objets combinatoires sont en bijection avec certaines classes de pastèques. Parmi ceux-ci, citons les permutations de Baxter [92, 36, 37], les chemins sous-diagonaux [57], certaines classes de tableaux de Young [58, 33], les pavages d'un hexagone avec des losanges [97], les couplages parfaits de graphes en nids d'abeilles [32], les réaliseurs de graphes plans maximaux (voir chapitre 3), etc. Remarquons également que de nombreux problèmes de la physique statistique peuvent être formalisés en termes de promeneurs méchants [47, 42].

La génération aléatoire uniforme est un outil puissant permettant d'étudier certaines propriétés d'objets combinatoires. De manière classique elle permet d'évaluer par exemple la complexité moyenne d'un algorithme utilisant tel ou tel objet combinatoire. Cela permet également d'émettre et de tester des conjectures. En physique, cela permet également de valider certains modèles théoriques.

Pour qu'un algorithme de génération aléatoire soit utilisable, il est important qu'il possède une bonne complexité afin de pouvoir générer des objets de très grande taille.

Pour la génération aléatoire de pastèques, plusieurs algorithmes existent déjà.

Remarquons qu'une pastèque à une branche avec mur (resp. sans mur) n'est autre qu'un chemin de Dyck (resp. de Grand Dyck) et peut donc être généré en temps linéaire grâce à l'algorithme présenté dans [4] (resp. [96]). Les pastèques avec mur à 2 branches de longueur $2l$ sans contrainte de déviation sont en bijection avec les polyominos parallélogrammes de périmètre $2l+4$. De plus, il existe une bijection directe entre de tels polyominos et des chemins de Dyck de longueur $2l+2$. Les pastèques à 2 branches sans mur peuvent donc être générées en temps linéaire. Pour des pastèques ayant un nombre arbitraire de branches, il existe des algorithmes de génération aléatoire ayant une complexité quadratique, si l'on considère la longueur des branches [98, 68].

En utilisant les formules d'énumération des étoiles (avec mur [42] ou sans mur [69, 21]), nous proposons un autre algorithme de génération aléatoire de pastèques à p branches pour une déviation fixée. Notre algorithme est exponentiel par rapport au nombre de branches, mais linéaire par rapport à la longueur des branches. Remarquons que si le nombre de branches n'est pas suffisamment petit ($p > \log(l)$), les algorithmes proposés par Wilson [98] et Krattenthaler [68] sont plus efficaces, que celui présenté dans ce chapitre.

Remarquons qu'il peut être utile de générer des pastèques avec un petit nombre de branches très longues. En particulier, cela permet de générer en temps linéaire des réaliseurs et des permutations de Baxter ayant un nombre arbitraire de montées (i.e. pour une permutation π, le nombre d'indice i tel que $\pi(i) < \pi(i+1)$).

L'algorithme de génération que nous présentons ici, fait avancer simultanément les p promeneurs construisant petit à petit les branches. A chaque étape, chaque promeneur effectue un pas Nord-Est ou Sud-Est. Les formules d'énumération, nous permettent de calculer efficacement la probabilité de chacune des 2^p transitions possibles, afin que toutes les pastèques ainsi générées aient la même probabilité d'apparition. A chaque étape, une transition est choisie en considérant les probabilités calculées. Cet algorithme de génération de pastèques utilise $O(p^2 2^p l)$ opérations arithmétiques.

En utilisant cet algorithme ainsi que la bijection présentée dans le chapitre 3 nous avons pu générer de manière aléatoire des réaliseurs de taille n. Ces expérimentations nous permettent de conjecturer que la hauteur moyenne des arbres des réaliseurs tend vers $0,97\sqrt{(}n)+o(\sqrt{n})$ et que la variance du nombre de faces tricolores (resp. cw-faces) d'un réaliseur tend vers $n/8$ (resp. $3n/8$). Enfin, nous observons expérimentalement que 92% des triangles tricolores d'un réaliseur sont des faces.

La suite du chapitre est organisée de la manière suivante. La section 4.1 présente les formules d'énumération des étoiles ainsi que les algorithmes de génération aléatoire uniforme. La section 4.2 rappelle quelques bijections entre les pastèques et d'autres objets combinatoires et quelques exemples d'objets aléatoires y sont présentés. Enfin, la section 4.3 présente quelques expérimentations et conjectures sur la hauteur des branches des pastèques ainsi que sur les réaliseurs.

4.1 Algorithme de Génération aléatoire de pastèques

4.1.1 Génération aléatoire de mots

Rappelons l'algorithme général présenté dans [29].

Soit L un langage sur un alphabet A. SoitL_n l'ensemble des mots de L de longueur n. Pour un mot w dans L, une lettre a de A et un entier n, on définit $proba_n(w,a)$ de la manière suivante :

$$proba_n(w,a) = \frac{\text{Nombre de mots de } L_n \text{ commençant par wa}}{\text{Nombre de mots de } L_n \text{ commençant par w}}.$$

En utilisant cette probabilité, il est possible de générer uniformément des mots de L_n grâce à l'algorithme suivant :

Algorithme 3 Génération pas à pas de mots.

$w \leftarrow \epsilon$
tant que $|w| < n$ **faire**
  a $\leftarrow$ une lettre a avec la probabilité $proba_n(w,a)$
  $w \leftarrow wa$
fin tant que

Soit $w = (w_1 w_2 \dots w_n)$ un mot de L_n. La probabilité $proba_n(w)$ pour un mot w

d'être généré par l'algorithme 3 est :

$$\begin{aligned} proba_n(w) &= \frac{|\text{mots commençant par } w_1|}{|\text{mots commençant par } \epsilon|}\frac{|\text{mots commençant par } w_1, w_2|}{|\text{mots commençant par } w_1|}\cdots \\ &\quad \cdots\frac{|\text{mots commençant par } w_1 w_2 \ldots w_n|}{|\text{mots commençant par } w_1 w_2 \ldots w_{n_1}|} \\ &= \frac{|\text{mots commençant par } w_1 w_2 \ldots w_n|}{|\text{mots commençant par } \epsilon|} \\ &= \frac{1}{|L_n|} \end{aligned}$$

Donc cet algorithme génère uniformément des mots de L_n. Ce principe a déjà été utilisé pour générer des mots de Dyck [4], des mots de Fibonacci [82] ainsi que des mots de n'importe quel *fg-langage* [30]. Un langage L est un fg-langage si l'ensemble des facteurs gauches de L est inclus dans L et que pour tout mot $u \in L$, il existe un mot $v \in L$ tel que u est un préfixe (ou facteur-gauche) propre de v (i.e. $u \neq v$ et u préfixe de v).

4.1.2 Génération de pastèques

Le principe de l'algorithme est de choisir simultanément, le pas (Nord-Est ou Sud-Est) de chaque promeneur avec la probabilité appropriée. Soit W_l^p l'ensemble des pastèques à p chemins de longueur l. Une *transition* tr est un vecteur de taille p composé de valeurs de $\{+1, -1\}$ indiquant pour chaque promeneur s'il doit effectuer un pas Nord-Est ou un pas Sud-Est. Plus précisément, $tr_i = +1$ indique que le i-ième promeneur effectue un pas Nord-Est. Soit w un facteur gauche de longueur k d'une pastèque (i.e. une étoile à p branches). On note $w + tr$ l'étoile de longueur $k + 1$ où la i-ième branche commence par la i-ième branche de w et se termine par un pas montant si $tr_i = +1$ et un pas descendant sinon.

Comme dans le cas de la génération de mots pas à pas, nous pouvons définir la probabilité d'une transition tr par

$$proba_l(w, tr) = \frac{\text{Nombre de pastèques dans } W_l^p \text{ commençant par } (w + tr)}{\text{Nombre de pastèques dans } W_l^p \text{commençant par } w}.$$

Soit $e = (e_1, e_2, \ldots, e_p)$ les ordonnées finales de chacun des promeneurs de w. Le nombre de pastèques de longueur l commençant par w est exactement le nombre d'étoiles de longueur $l - k$ se terminant en $(e_1, e_2, \ldots, e_p)$. Les deux théorèmes suivants donnent l'énumération de ces étoiles avec ou sans mur.

Théorème 4.1.1. *(***Essam et Guttmann** *[42])*
Soient $e_1 < e_2 < \cdots < e_p$ avec $e_i \equiv l(\mod 2), i = 1, 2, \ldots, p$. Le nombre d'étoiles de longueur l sans mur se terminant en $e = (e_1, e_2, \ldots, e_p)$ est :

$$2^{-\binom{p}{2}} \prod_{1\le i\le p} \frac{(l-i+p)!}{(\frac{l+e_i}{2})!(\frac{l-e_i}{2}+p-1)!} \prod_{1\le i<j\le p} (e_j-e_i)$$

Théorème 4.1.2. *(***Krattenthaler, Guttmann et Viennot** *[69])*
Soient $e_1 < e_2 < \cdots < e_p$ avec $e_i \equiv l(\mod 2), i = 1, 2, \ldots, p$. Le nombre d'étoiles de longueur l avec mur se terminant en $e = (e_1, e_2, \ldots, e_p)$ est :

$$2^{-p^2+p} \prod_{1\le i\le p} \frac{(e_i+1)(l+2i-2)!}{(\frac{l+e_i}{2}+p)!(\frac{l-e_i}{2}+p-1)!} \prod_{1\le i<j\le p} (e_j-e_i)(e_i+e_j+2)$$

Remarquons que dans le cas des pastèques à deux branches avec mur et sans déviation, on retrouve la formule donnée par Gouyou-Beauchamps (théorème 3.1.1).

On peut remarquer également que la probabilité d'une transition dépend uniquement de la hauteur de chaque promeneur ainsi que du nombre de pas qu'il reste à faire. Donc $proba_l(w, tr) = proba_l(e, k, tr)$ où e désigne le vecteur des ordonnées finales de w et k la longueur de w.

De plus, la hauteur respective de chaque promeneur de w et $w + tr$ ne diffère que de 1. Ceci implique de nombreuses simplifications dans le calcul de $proba_l(e, k, tr)$. Les algorithmes 4 et 5 permettent de calculer $proba_l(e, k, tr)$ pour les pastèques avec ou sans mur.

Algorithme 4 Calcul de $proba_l(e, k, tr)$ pour les pastèques à p branches avec mur.

$prob \leftarrow 1$
pour $i = 1$ à $p - 1$ **faire**
  pour $j = i + 1$ à p **faire**
    $prob \leftarrow prob \times \frac{e_j+tr_j-(e_i+tr_i)}{e_j-e_i} \times \frac{e_i+tr_i+e_j+tr_j+2}{e_i+e_j+2}$
  fin pour
fin pour
pour $i = 1$ à p **faire**
  $prob \leftarrow prob \times \frac{e_i+tr_i+1}{(e_i+1)(l-k+2i+1)}$
  si $tr_i > 0$ **alors**
    $prob \leftarrow prob \times (\frac{l-k-1-e_i}{2}+p)$
  sinon
    $prob \leftarrow prob \times (\frac{l-k+1+e_i}{2}+p)$
  fin si
fin pour
retourner $prob$

Les probabilités calculées sont des nombres rationnels. La valeur exacte de chaque probabilité $proba_l(e, k, tr)$ peut donc être calculée.

Pour choisir une transition tr avec la probabilité $proba_l(e, k, tr)$, la méthode d'inversion classique peut être utilisée [34] :

Algorithme 5 Calcul de $proba_l(e, k, tr)$ pour les pastèques à p branches sans mur.

$prob \leftarrow 1$
pour $i = 1$ à $p - 1$ **faire**
  pour $j = i + 1$ à p **faire**
    $prob \leftarrow prob \times \frac{e_j + tr_j - (e_i + tr_i)}{e_j - e_i}$
  fin pour
fin pour
pour $i = 1$ à p **faire**
  si $tr_i > 0$ **alors**
    $prob \leftarrow prob \times \frac{l-k-1-e_i+p/2}{2(l-k-i+p)}$
  sinon
    $prob \leftarrow prob \times \frac{l-k+1+e_i}{2(l-k-i+p)}$
  fin si
fin pour
retourner $prob$

Algorithme 6 Choix d'une transition.

$h \leftarrow$ un nombre aléatoire entre 0 et 1.
$tr \leftarrow 1$
$p \leftarrow proba_l(e, k, tr)$
tant que $p < h$ **faire**
  $tr \leftarrow Next(tr)$
  $p \leftarrow p + proba_l(e, k, tr)$
fin tant que
return tr

tr peut être vu comme un vecteur binaire codant un entier i en base 2, à ceci près que tr est un vecteur à valeurs dans $\{-1, +1\}$ au lieu d'être un vecteur à valeurs dans $\{0, 1\}$. La fonction $Next(tr)$ désigne alors le vecteur tel que tr' code le nombre $i + 1$.

La figure 46 est une pastèque sans mur générée de manière aléatoire et la figure 47 est une pastèque avec mur générée de manière aléatoire.

Théorème 4.1.3. *Il existe un algorithme de génération aléatoire uniforme de pastèques à p branches de longueur l (avec ou sans mur) s'exécutant en $O(p^2 2^p l)$ opérations arithmétiques.*

Démonstration. Dans les algorithmes 5 et 4, le calcul d'une transition s'effectue en $O(p^2)$ opérations arithmétiques (chaque algorithme contient 2 boucles imbriquées). A chaque étape, au plus 2^p transitions doivent être calculées. Donc dans le pire des cas, il faut $O(p^2 2^p)$ opérations pour que tous les promeneurs avancent d'un pas. Donc il faut au total $O(lp^2 2^p)$ pour générer une pastèque. □

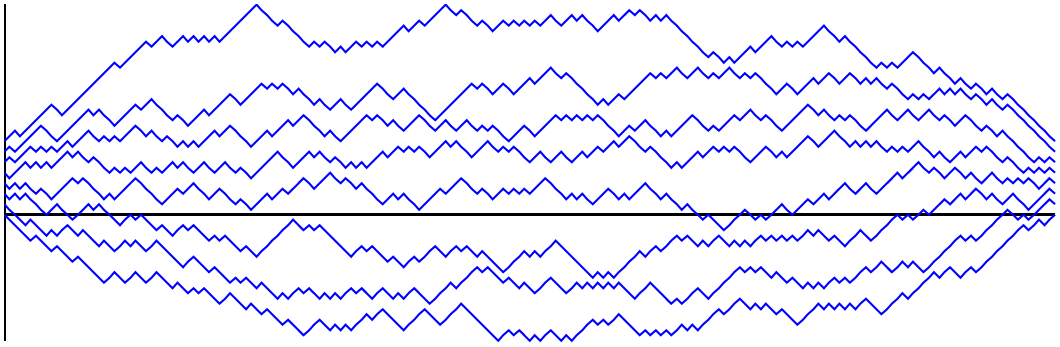

Figure 46 – Une pastèque aléatoire à 8 branches de longueur 200 sans mur.

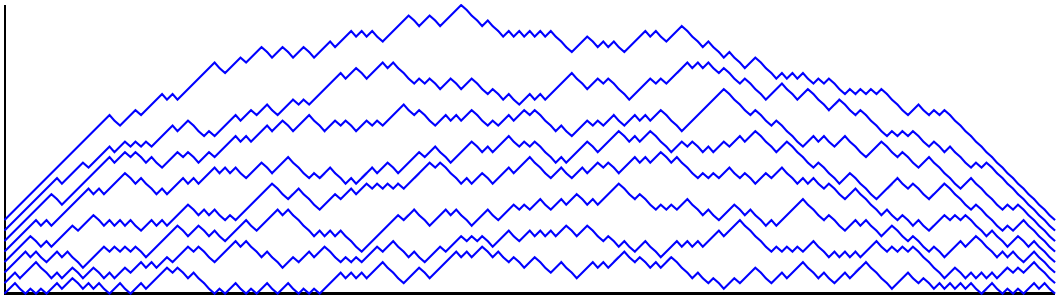

Figure 47 – Une pastèque aléatoire à 8 branches de longueur 200 avec mur.

Les algorithmes présentés ici ont été implémentés[1] en C++ en utilisant la bibliothèque Gnu-Multiple-Precision [60]. La figure 48 représente le temps de calcul utilisé pour générer des pastèques sans mur. Nous pouvons vérifier que le modèle de complexité considéré (nombre d'opérations arithmétiques) pour évaluer la complexité effective de l'algorithme est satisfaisant.

4.2 Applications à d'autres objets combinatoires

4.2.1 Polyominos parallélogrammes

Un *polyomino* est un ensemble fini et connexe de carrés unitaires sans point séparateur et défini à translation prêt. Un polyomino est *horizontalement convexe* (resp. *verticalement convexe*) si toutes les colonnes (resp. lignes) sont connexes. Un polyomino est *convexe* s'il est à la fois verticalement convexe et horizontalement convexe. Un *polyomino parallélogramme* est un polyomino défini par 2 chemins composés uniquement de pas Nord et de pas Est. Ces chemins sont disjoints à l'exception de leurs extrémités. Ces chemins peuvent être obtenus à partir d'une pastèque à deux branches en ajoutant

[1] Une implémentation est disponible à l'adresse suivante :
http : *//www.labri.fr/* ∼ *bonichon/waterlon*

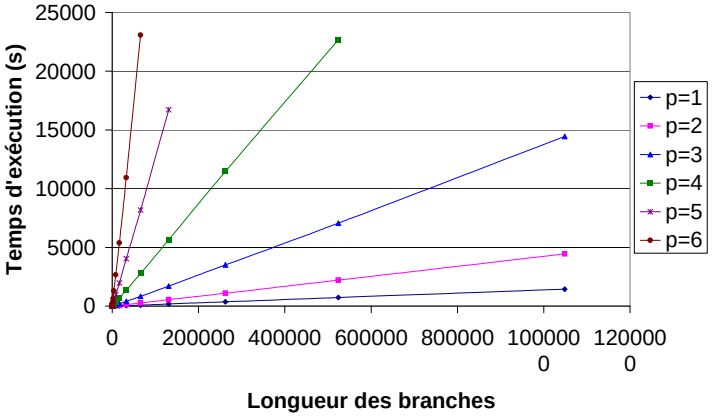

Figure 48 – Complexité des algorithmes de génération. (Test réalisé sur un Pentium 166)

un pas au début et à la fin de chaque chemin et en effectuant une rotation de $\pi/4$ dans le sens trigonométrique (voir figure 49).

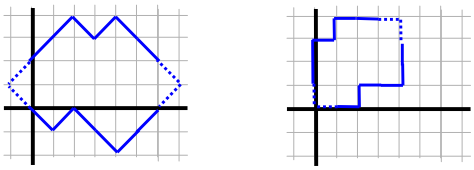

Figure 49 – Passage de la pastèque à 2 branches à un polyomino parallélogramme.

Les polyominos parallélogrammes de périmètre $2n$ peuvent être générés uniformément en temps linéaire en utilisant une bijection avec les mots de Dyck de longueur $2n-2$ [28].

La *largeur* d'un polyomino est le nombre de colonnes du polyomino. En utilisant l'algorithme de génération de pastèques à 2 branches sans mur, on obtient le corollaire suivant :

Corollaire 4.2.1. *Un polyomino parallélogramme de périmètre et largeur fixés peut être généré uniformément et en temps linéaire.*

Deux polyominos parallélogrammes sont *jumeaux* si le bord supérieur du premier correspond au bord inférieur du second. k polyominos parallélogrammes $p_1, p_2, \ldots, p_k$ sont dit *polyominos jumeaux* si pour tout $i < k$, le bord supérieur de p_i correspond au bord inférieur du polyomino p_{i+1}.

Il existe une bijection évidente entre l'ensemble des k polyominos jumeaux et les pastèques à $k+1$ branches sans mur.

Corollaire 4.2.2. *k polyominos jumeaux de périmètre et de largeur fixés peuvent être générés uniformément en temps linéaire.*

Un exemple de 7 polyominos jumeaux de périmètre 404 et de largeur 101 est donné dans la figure 50. Ces polyominos jumeaux sont ceux correspondant aux pastèques de la figure 46.

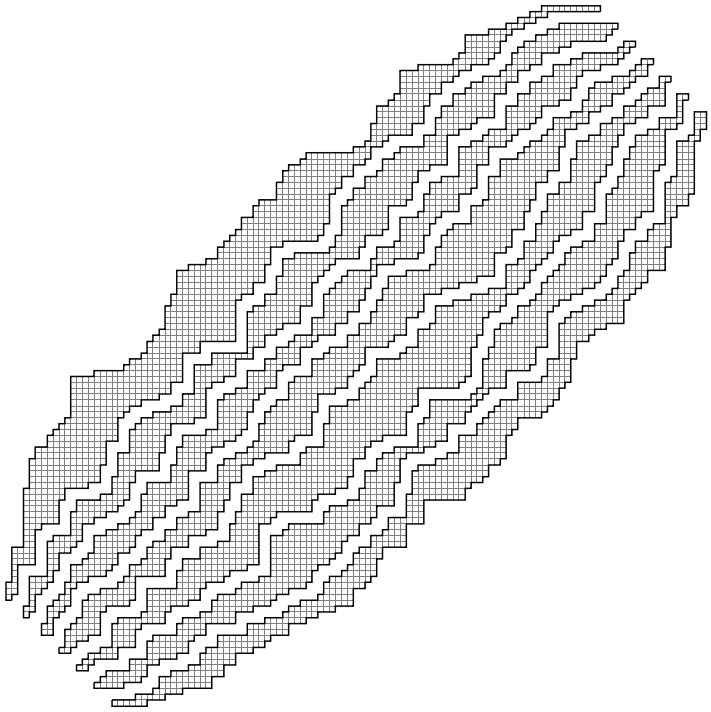

Figure 50 – 7 polyominos jumeaux.

4.2.2 Chemins planaires sous-diagonaux

Les chemins considérés dans cette sous-section sont les chemins qui partent de l'origine et qui sont composés uniquement de pas Nord, Sud, Est et Ouest. On appelle *boucle* un chemin qui retourne à l'origine. Un tel chemin est dit *sous-diagonale* s'il reste dans le huitième de plan situé au-dessus de l'axe des abscisses et de la première diagonale.

Gouyou-Beauchamps [57] a montré que les chemins sous-diagonaux de longueur n arrivant au point (x, y) étaient en bijection avec les pastèques à 2 branches avec mur arrivant respectivement aux points (n, y) et $(n, y + x)$.

Etablir cette bijection est relativement simple : lorsque les deux promeneurs montent, le chemin effectue un pas Est, lorsque les deux promeneurs descendent, le chemin effectue un pas Ouest, lorsque les deux promeneurs s'éloignent l'un de l'autre, le chemin effectue un pas Nord et enfin lorsque les deux promeneurs se rapprochent alors le chemin effectue un pas Sud.

Corollaire 4.2.3. *On peut générer uniformément des chemins sous-diagonaux selon leurs longueurs et les coordonnées de leur point d'arrivée en temps linéaire.*

La figure 51 montre un exemple de boucle sous-diagonale de longueur 10^5 générée aléatoirement.

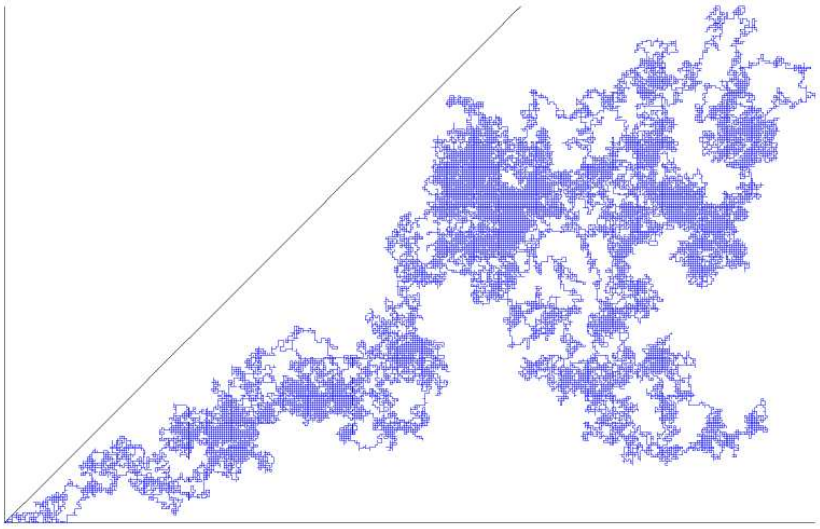

Figure 51 – Une boucle sous-diagonale de longueur 10^5 générée aléatoirement.

4.2.3 Réaliseurs aléatoires

Comme nous l'avons vu dans le chapitre 3, il existe une bijection entre les réaliseurs de taille n et les pastèques à deux branches de longueur $2n-6$ avec mur. De plus, le réaliseur correspondant à une telle pastèque peut être calculé en temps linéaire. En conséquence, on peut générer de manière aléatoire et uniforme un réaliseur en temps linéaire.

La figure 52 montre un réaliseur aléatoire de taille 100. L'algorithme utilisé pour dessiner le réaliseur est celui proposé par Schnyder [87].

4.2.4 Arbres binaires jumeaux et permutations de Baxter

Soit $Tree_n$ l'ensemble des arbres binaires à n sommets. Classiquement, un arbre binaire t est encodé par un mot de Dyck. De manière classique, un arbre binaire t peut être encodé par un mot de Dyck, défini récursivement de la manière suivante :

$$code(t) = 1\ code(\text{sous} - \text{arbre gauche}(t))\ 0\ code(\text{sous} - \text{arbredroit}(t))$$

$$code(\emptyset) = \epsilon \text{ o } \emptyset \text{ dsigne l}'\text{arbre binaire de Tree}_0.$$

Définition 4.2.1. *L'ensemble des arbres binaires jumeaux* $Twin_n \subseteq Tree_n \times Tree_n$ *est défini de la manière suivante :*

$Twin_n = \{(a_1, a_2) : a_1, a_2 \in Tree_n et\ \Theta(code(a_1)) = \Theta^c(code(a_2)\},$

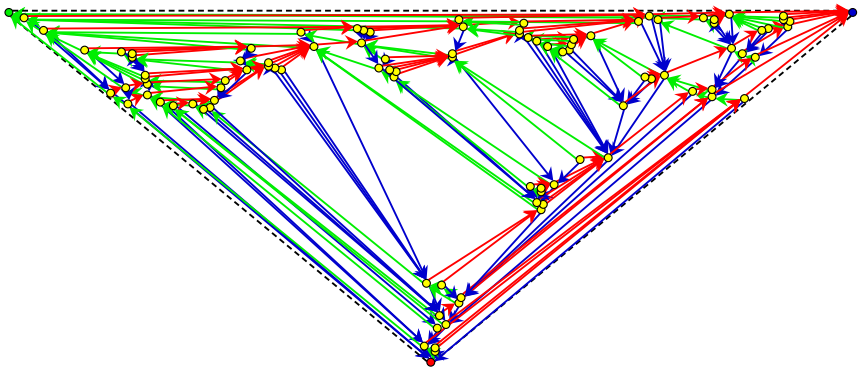

Figure 52 – Réaliseur aléatoire de taille 100.

où Θ consiste à étiqueter les feuilles gauches (resp. droite) d'un arbre binaire complété par la lettre 0 (resp. la lettre 1) à l'exception des deux feuilles extrêmes et Θ^c est identique à Θ à la différence près que les lettres 0 et 1 sont inversées.

Théorème 4.2.1. *(***Dulucq et Guibert [37]***) Il existe une bijection entre les arbres binaires jumeaux de taille n et les pastèques sans mur à 3 branches de longueur n.*

Corollaire 4.2.4. *Les arbres binaires jumeaux à n sommets et r arêtes droites dans le premier arbre peuvent être générés de manière aléatoire et uniforme en temps linéaire.*

Soit S_n l'ensemble des nombres entiers compris entre 1 et n.

Définition 4.2.2. *Une permutation π de S_n est une permutation de Baxter si et seulement si, pour tout entier $i \in [n-1]$, π peut être factorisée de manière unique par $\pi = \pi'.i.\overset{<}{\pi}.\overset{>}{\pi}.i+1.\pi''$ ou $\pi = \pi'.i.\overset{>}{\pi}.\overset{<}{\pi}.i+1.\pi''$,*

où toutes les lettres de $\overset{<}{\pi}$ (resp. $\overset{>}{\pi}$) sont plus petites que i (resp. plus grand que $i+1$).

Le *nombre de montées* d'une permutation π est égal au nombre d'indices i tel que $\pi(i) < \pi(i+1)$.

Théorème 4.2.2. *(***Dulucq et Guibert [36]***) Il existe une bijection entre les permutations de Baxter sur S_n avec r montées et les arbres binaires jumeaux de $Twin_n$ avec r arêtes gauches dans le premier arbre.*

Cet algorithme qui calcule la permutation de Baxter à partir d'arbres binaires jumeaux possède une complexité quadratique dans le pire des cas. On peut donc générer uniformément des permutations de Baxter de S_n à r montées en $O(n^2)$ dans le pire des cas.

4.3 Expérimentations

4.3.1 Hauteur des branches dans les pastèques avec mur

La *hauteur* d'un promeneur désigne, la hauteur maximale qu'il atteint lors de sa marche. La hauteur d'une pastèque est définie naturellement comme la hauteur de son dernier promeneur.

Si l'on prend l'exemple de la pastèque de la figure 45 (a), son deuxième promeneur est de hauteur 5 et la pastèque est de hauteur 7.

On note $\bar{H}_w(l,i,p)$ (resp. $\bar{H}_{nw}(l,i,p)$), avec $i \leq p$, la hauteur moyenne du i-ième promeneur parmi l'ensemble des pastèques à p branches de longueur l avec mur (resp. sans mur). La hauteur moyenne des pastèques à p branches de longueur l avec mur (resp. sans mur) sera notée $\bar{H}_w(l,p)$ (resp. $\bar{H}_{nw}(l,p)$) à la place $\bar{H}_w(l,p,p)$ (resp. $\bar{H}_{nw}(l,p,p)$).

Théorème 4.3.1. *(De Bruijn, Knuth et Rice 1972) [27]*
La hauteur moyenne d'un chemin de Dyck de longueur l est :

$$\bar{H}_w(l,1) = \sqrt{\frac{\pi l}{2}} + O(1).$$

Notre but est de généraliser ce résultat aux pastèques. Pour cela, nous avons effectué des expérimentations sur la hauteur des pastèques à l'aide de l'algorithme de génération présenté précédemment.

Le résultat de ces expérimentations est présenté dans les Figures 53 et 54.

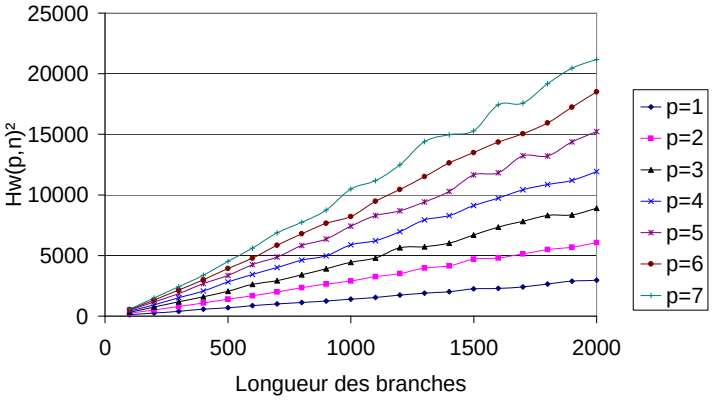

Figure 53 – Hauteur moyenne des pastèques avec mur en fonction du nombre de branches et de la longueur des branches.

Les valeurs présentées dans les Figures 53, 54 et 55 ont été obtenues à partir de la génération de pastèques de longueur $i * 100$ avec $i = 1, 2, \ldots, 20$. Pour chacune de ces tailles, 100 pastèques ont été générées.

Résultat Expérimental 4.3.1. *La hauteur moyenne d'une pastèque à p branches avec mur est expérimentalement :*

$$\bar{H}_w(p,p,l) \approx \sqrt{(1,67p - 0,06)l} + o(\sqrt{l}).$$

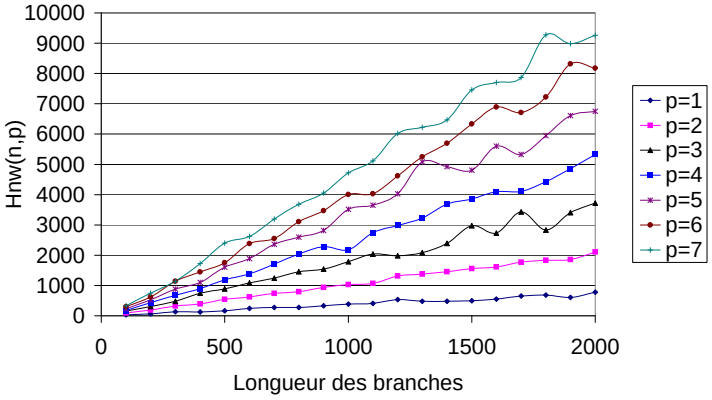

Figure 54 – Hauteur moyenne des pastèques sans mur en fonction du nombre de branches et de la longueur des branches.

Résultat Expérimental 4.3.2. *La hauteur moyenne d'une pastèque à p branches sans mur est expérimentalement :*

$$\bar{H}_{nw}(p,p,l) \approx \sqrt{(0,82p-0,46)l} + o(\sqrt{l}).$$

Résultat Expérimental 4.3.3. *La hauteur moyenne de la première branche d'une pastèque à p branches avec mur est :*

$$\bar{H}_{nw}(1,p,l) \approx a(p)\sqrt{l} + o(\sqrt{l}),$$

avec

$$a(p) \approx p^{-0,36}\sqrt{\frac{\pi}{2}}.$$

La figure 55 représente les valeurs expérimentales de $a(p)$.

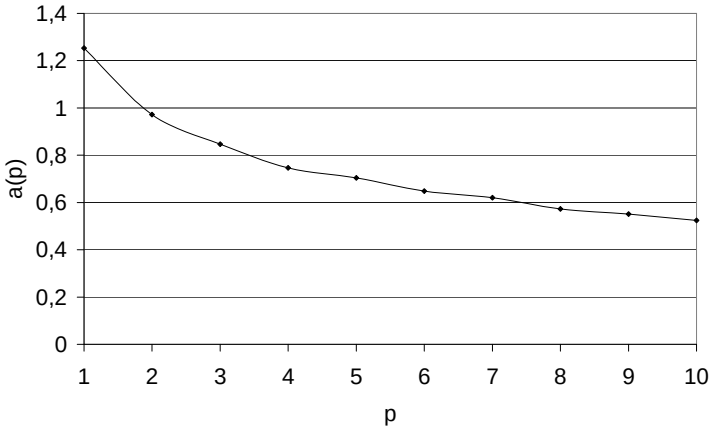

Figure 55 – $a(p)$.

4.3.2 Réaliseurs aléatoires

L'efficacité de certains algorithmes utilisant les réaliseurs [23, 16, 70] dépend du nombre de leurs faces tricolores et donc du nombre de feuilles des arbres du réaliseur. Comme nous l'avons vu, le nombre moyen de faces tricolores d'un réaliseur tend vers $\frac{n}{2}+o(n)$. La figure 56 confirme bien ce résultat. On peut également observer que presque tous les triangles tricolores sont des faces :

$$\frac{\Delta}{\text{triangles tricolores}} \approx 92\%.$$

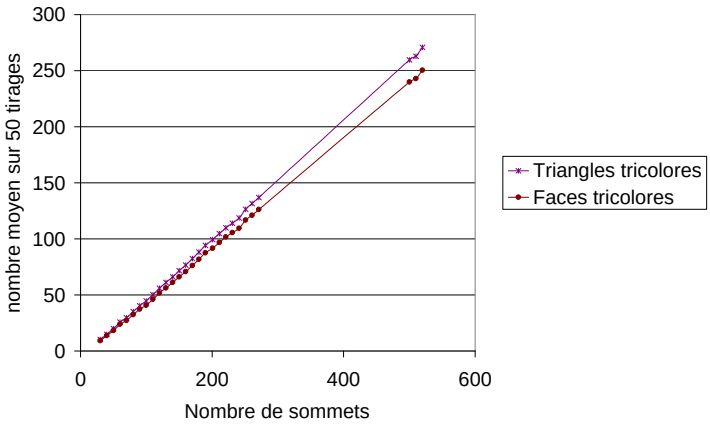

Figure 56 – Nombre de faces tricolores et nombre de triangles tricolores.

Nous avons également déjà vu que $\overline{\Delta}_+ = \overline{\Delta}_- = \frac{n}{4} + o(n)$ (voir corollaire 3.4.1). Les expérimentations sur les réaliseurs aléatoires nous permettent de proposer la conjecture suivante :

Conjecture 4.3.1.

$$Var(\Delta_+) = Var(\Delta_-) = \frac{n}{8} + o(n) \quad (1)$$

$$CoVar(\Delta_+, \Delta_-) = \frac{n}{16} + o(n) \quad (2)$$

$$Var(\Delta) = \frac{3n}{8} + o(n). \quad (3)$$

Remarque : la troisième formule de la conjecture se déduit des 2 premières car $Var(X+Y) = Var(X) + Var(Y) + 2CoVar(X,Y)$.

Intéressons nous maintenant à la profondeur d'un arbre d'un réaliseur aléatoire. Comme nous l'avons remarqué, il y a une bijection directe entre les paires de chemins de Dyck ne se coupant pas et les pastèques à 2 branches avec mur. De plus la hauteur de la premier branche de la pastèque correspond exactement à la hauteur du premier chemin de Dyck.

Comme nous l'avons vu dans le chapitre 3, la hauteur de la première branche d'une pastèque à 2 branches est exactement la profondeur de l'arbre T_0 du réaliseur. Pour des raisons de symétrie, les trois arbres ont la même profondeur. La figure 55 nous donne donc la profondeur moyenne d'un arbre d'un réaliseur.

Résultat Expérimental 4.3.4. *La profondeur moyenne d'un arbre d'un réaliseur de taille n est :*

$$0,97\sqrt{n} + o(\sqrt{(n)}).$$

4.4 Conclusion

Nous venons de voir un algorithme qui construit aléatoirement et uniformément des pastèques. Cet algorithme est efficace pour p petit (complexité en $O((p^2 \times 2^p)l)$), et particulièrement pour les exemples considérés ($p = 2$ et $p = 3$). Nous pensons que cet algorithme pourrait être amélioré pour atteindre une complexité en $O(p \times 2^p)l)$, en utilisant d'un parcours de type "Gray code" [56] des transitions. Dans un tel parcours, on passe d'une transition à une autre, en changeant une seule case du vecteur de transition. Ce parcours permettrait de calculer rapidement la probabilité d'une transition en fonction de la probabilité de la précédente. Toutefois comme l'algorithme présenté n'est efficace que pour p petit, il nous parait préférable de présenter un algorithme théoriquement moins efficace, mais plus facile à implémenter et à comprendre.

Les algorithmes présentés ici génèrent des pastèques. Avec des modifications mineures, ils peuvent être également utilisés pour générer des étoiles dont les altitudes d'arrivées sont fixées.

Si au contraire, on souhaite générer une pastèque de déviation quelconque (par exemple pour générer une permutation de Baxter sans contrainte sur le nombre de montées) à l'aide des algorithmes présentés ici, il "suffit" de choisir avec la probabilité appropriée la déviation de la pastèque. Ceci pourrait être fait en utilisant des techniques similaires à celles proposées par Alonso [3].

Deuxième partie

Utilisations des réaliseurs pour le dessin et le codage

Chapitre 5

Un algorithme de dessin lignes brisées basé sur les réaliseurs

Introduction

De part le nombre et la variété de ses applications, le sujet du dessin de graphe a reçu une intense attention. Ces applications peuvent être regroupées en deux catégories : la représentation d'information (Modèle Conceptuel des Données, interactions entre gènes, Hiérarchie des classes, structure d'un site web, carte mentale, etc.) et le routage de circuit électronique. Dans le premier cas, le dessin doit faire ressortir de la manière la plus claire l'information contenue dans le graphe. Dans le deuxième cas, le dessin doit respecter les contraintes technologiques et la surface du dessin doit être la plus faible possible afin d'obtenir des circuits les moins coûteux possibles.

Ici nous nous intéressons aux dessins planaires de graphes planaires. Dans de tels dessins, les arêtes ne se coupent pas. Rappelons que si le graphe n'est pas planaire on peut se ramener au cas du dessin de graphe planaire, soit en partitionnant les arêtes en groupe de graphes planaires, soit en remplaçant chaque intersection d'arête par un sommet de degré 4 qui ne sera pas affiché.

Bien qu'il existe de nombreux algorithmes de dessin de graphes dans l'espace, nous nous intéressons ici uniquement aux dessins de graphes dans le plan. Plus précisément aux *dessins plans* de graphes planaires. On entend par dessin plan, un dessin sans croisement.

Suivant le type d'applications et le type de graphes considérés, plusieurs modèles de dessins sont utilisés. Les sommets sont représentés soit par des points du plan (généralement des points de coordonnées entières), soit par des rectangles. Les arêtes quant à elles sont représentées par des lignes droites ou par des lignes brisées. Dans le cas où les sommets sont représentés par des points et les arêtes par des lignes droites, on parle de *dessin lignes droites* ("straight-line drawing" en anglais). Dans le cas où les sommets sont représentés par des points et les arêtes par des lignes brisées, on parle de *dessin lignes brisées* ("polyline drawing" en anglais). La figure 57 montre quelques

exemples de dessins lignes brisées.

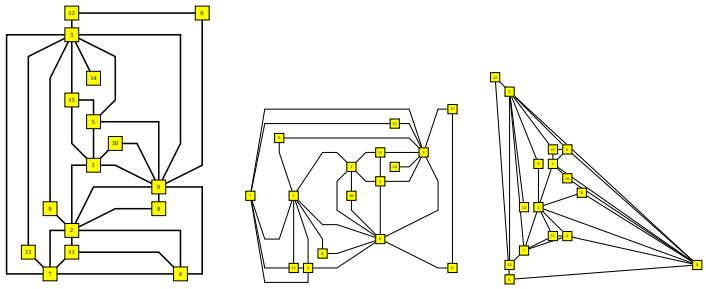

Figure 57 – Différents dessins lignes brisées d'un même graphe. De gauche à droite : "Mixed-Model" [62], "quasi-orthogonal", lignes droites [87].

On parlera de *dessins orthogonaux* lorsque les arêtes sont représentées par des séquences de lignes horizontales et verticales (voir figure 58)

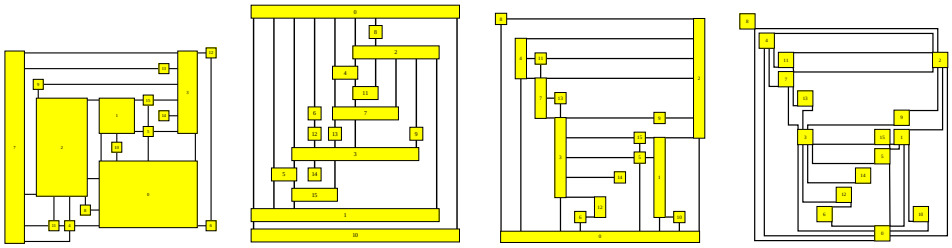

Figure 58 – Différents dessins orthogonaux d'un même graphe. De gauche à droite : Giotto, visibilité, dessin de 2-visibilité, Kandinsky.

Dans ce chapitre nous nous intéressons aux dessins lignes brisées.
Les critères de qualité classiquement considérés pour ce type de dessins sont :

- Taille de la grille (en effet, une grille petite assure, pour une surface d'affichage fixée une distance minimale entre les sommets).
- Résolution angulaire, qui désigne l'angle minimal entre deux arêtes adjacentes. Si cette valeur est trop petite, il devient difficile de distinguer deux arêtes au voisinage d'un sommet.
- Nombre de brisures. Plus il y a de brisures, plus le dessin parait compliqué. De plus, si une arête possède de nombreuses brisures, il devient difficile d'identifier ses extrémités.
- Complexité de l'algorithme de dessin. La plupart des algorithmes de dessin ont une complexité linéaire.

De manière générale, il est difficile d'optimiser simultanément tous ces critères. Dans [62] un bon compromis est obtenu entre la taille de la grille, le nombre de brisures et la résolution angulaire. Plus précisément les dessins utilisent une grille de

$(2n-5) \times (\frac{3}{2}n - \frac{7}{2})$, au plus $5n-15$ brisures (chaque arête est brisée au plus 3 fois) et une résolution angulaire supérieure à $\frac{2}{\deg_{max}}$. Dans [22] les dessins obtenus utilisent une grille de $30n \times 15n$, au plus une brisure par arête et une résolution angulaire de $\Theta(1/\deg_{max})$. Ici notre but est d'obtenir des dessins sur une grille de taille optimale avec un faible nombre de brisures. Schnyder [87] a montré qu'il est possible de dessiner un graphe plan à l'aide de lignes droites sur une grille de $(n-2) \times (n-2)$. De plus, De Fraysseix, Pach et Pollack [51] ont montré que certains graphes de taille n nécessitent une grille $(\lfloor \frac{2(n-1)}{3} \rfloor) \times (\lfloor \frac{2(n-1)}{3} \rfloor)$ pour être dessinés. Par ailleurs, Chrobak et Nakano [25] ont proposé un algorithme donnant des dessins lignes droites de largeur optimale et de hauteur au plus $4\lfloor \frac{2n-1}{3} \rfloor - 1$. Nous présentons ici un algorithme s'appuyant sur les réaliseurs produisant des dessins lignes brisées sur une grille de surface optimale $(\frac{4(n-1)^2}{9})$ et de largeur optimale $(\lfloor \frac{2(n-1)}{3} \rfloor)$ où chaque arête est brisée au plus une fois et le nombre total de brisures est au plus $n-2$. Ces dessins étant obtenus en temps linéaire, l'algorithme possède un intérêt théorique et pratique. La suite du chapitre est organisée de la manière suivante. La section 5.1 donne de manière formelle les définitions de dessins lignes brisées et donne une borne inférieure sur la taille de la grille. La section 5.2 présente le principe de dessin lignes brisées d'un réaliseur. La section 5.3 introduit la notion de *stratification-faible*[1] d'un réaliseur qui est un ensemble de contraintes sur les ordonnées des sommets d'un réaliseur afin de pouvoir le dessiner. La section 5.4 donne l'algorithme de dessin d'un réaliseur stratifié. Enfin, la section 5.5 présente un algorithme linéaire permettant de calculer une stratification-faible d'un réaliseur.

5.1 Dessin lignes brisées d'un graphe planaire

Un *dessin lignes brisées* d'un graphe G est un dessin de G où les sommets sont représentés par des points ayant des coordonnées entières et les arêtes par des lignes brisées, où les brisures ont aussi des coordonnées entières. Un dessin *planaire lignes brisées* est un dessin lignes brisées où les arêtes ne se croisent pas. La *largeur* d'un dessin lignes brisées est définie par la différence entre la plus petite abscisse d'un sommet ou d'une brisure et la plus grande abscisse d'un sommet ou d'une brisure. De manière similaire la *hauteur* d'un dessin lignes brisées est donnée par la différence entre la plus petite ordonnée d'un sommet ou d'une brisure et la plus grand ordonnée d'un sommet ou d'une brisure.

Par exemple la taille de la grille du premier dessin de la figure 57 est 9×12. Ce dessin possède également 21 brisures et les arêtes sont brisées au plus 2 fois.

La propriété suivante a été prouvé pour les dessins en lignes droites [51], utilisant des triangles emboîtés (cf. figure 59). Utilisant la même construction, le résultat s'étend directement aux dessins lignes brisées.

[1] Dans [19] une version plus restrictive de nivelage, appelé *stratification*, a été définie. Cette définition plus restrictive, fut introduite pour calculer des dessins 2-visibilité. La stratification-faible étant moins contraignante, elle autorise des nivelages moins hauts et donc des dessins plus compacts.

Propriété 5.1.1. *Pour chaque $n \geq 3$, il existe un graphe plan à n sommets H_n dont la largeur et la hauteur de la grille de chacun de ses dessins planaires lignes brisées est d'au moins $\lfloor \frac{2(n-1)}{3} \rfloor$ et la surface de la grille est d'au moins $\frac{4(n-1)^2}{9}$.*

Démonstration. Pour des raisons de symétrie, on peut considérer uniquement la largeur du dessin. H_n est construit récursivement. H_3 étant le triangle v_1, v_2, v_3 et pour $n \geq 4$, H_n est obtenu en ajoutant le sommet v_n dans la face extérieure de H_{n-1} et en le connectant aux sommets v_{n-1}, v_{n-2} et v_{n-3}. De cette manière la face extérieure de H_n est constituée des sommets v_{n-2}, v_{n-1} et v_n.

Remarquons tout d'abord que pour $n = 3, 4, 5$, un dessin lignes brisées de H_n utilise respectivement des grilles de largeur au moins $1, 2$ et 2, ce qui vaut bien $\lfloor \frac{2(n-1)}{3} \rfloor$. Par induction, on observe qu'il faut au moins deux colonnes de plus pour dessiner H_n que pour dessiner H_{n-3}. □

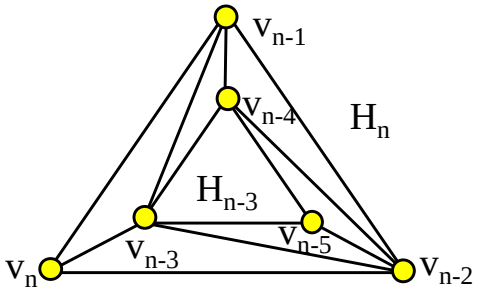

Figure 59 – Constructions des graphes H_n.

5.2 Principe de dessin lignes brisées d'un réaliseur

Etant donné G un graphe plan maximal et $R = (T_0, T_1, T_2)$ un de ses réaliseurs, on calcule un dessin lignes brisées de G.

Après avoir choisi un arbre de R, disons T_0, une colonne est allouée pour chaque feuille u de T_0 dans l'ordre d'apparition dans un parcours préfixe anti-trigonométrique. On note $x(u)$ le numéro de cette colonne. Chaque nœud interne de T_0 est placé sur la colonne d'une des feuilles de son sous-arbre.

Maintenant, il reste à calculer l'ordonnée $y(u)$ de chaque sommet u de G. Pour cela, nous définissons d'abord quelques règles sur la position des brisures des arêtes :

- Si une brisure est nécessaire pour l'arête $(u, P_0(u))$, elle aura les coordonnées suivantes : $(x(u), y(P_0(u)) + 1)$.
- Si une brisure est nécessaire pour une arête $(u, P_1(u))$, elle aura les coordonnées suivantes : $(x(first\_leaf(u)), y(u))$, où $first\_leaf(u)$ désigne la première feuille du sous-arbre issu de u.

– De manière similaire, si une brisure est nécessaire pour une arête $(u, P_2(u))$, elle aura les coordonnées suivantes : $(x(last\_leaf(u)), y(u))$, où $last\_leaf(u)$ désigne la dernière feuille du sous-arbre issu de u.

La figure 60 illustre la manière dont les arêtes de différents types sont dessinées.

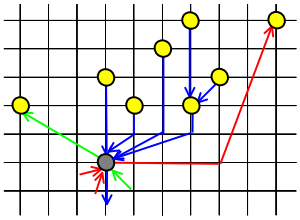

Figure 60 – Dessins d'une arête.

Pour un dessin planaire lignes brisées, les arêtes ne doivent pas se chevaucher, même partiellement. Pour éviter cela, d'autres règles sont nécessaires. La configuration gauche de la figure 61 illustre le cas où deux arêtes se chevauchent. Nous proposons une nouvelle règle concernant l'abscisse des sommets : si $v = P_2(u)$ et $\mathrm{y}(v) = y(u)$ alors $x(v) = x(last\_leaf(v))$ sinon $x(v) = x(first\_leaf(v))$. Comme on le voit sur la configuration droite de la figure 61, lorsqu'on applique cette règle le chevauchement disparaît.

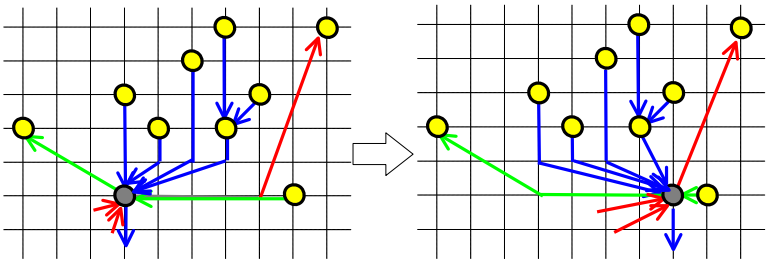

Figure 61 – Configuration de chevauchement d'arête et configuration corrigée.

Finalement, nous considérons les arêtes externes (v_1, v_0) et (v_2, v_0) comme si elles appartenaient à l'arbre T_0 et que l'arête externe (v_1, v_2) appartient à l'arbre T_2.

Ayant défini la méthode de dessin d'un réaliseur (et donc d'un graphe plan maximal) nous proposons un ensemble de contraintes sur les ordonnées des sommets qui assure qu'un tel type de dessin est possible. Par la suite, nous montrerons que des ordonnées qui vérifient ces contraintes peuvent être calculées en temps linéaire pour n'importe quel réaliseur. De plus, nous montrerons que les dessins obtenus ont une taille de grille optimale.

5.3 Stratification-faible d'un réaliseur

Un *nivelage* d'un graphe G est une application de l'ensemble des sommets de G vers l'ensemble des entiers positifs. Comme nous le verrons, un nivelage peut être utilisé pour définir l'ordonnée des sommets dans le plan. Nous définissons un nivelage particulier d'un réaliseur, qui permet d'obtenir des dessins lignes brisées de taille optimale.

Définition 5.3.1. *Soient G un graphe plan maximal, $R = (T_0, T_1, T_2)$ un réaliseur de G et L un nivelage de G. $L_{\max_1}(u)$ et $L_{\max_2}(u)$ sont respectivement définis de la manière suivante : $L_{\max_1}(u) = \max(L(u'), u' \in Ch_1(u))$ et $L_{\max_2}(u) = \max(L(u'), u' \in Ch_2(u))$. On dit que L est une stratification-faible de R si pour tout sommet interne u de G, les conditions suivantes sont vérifiées :*

1. $L(v_0) = 0$
2. $L(P_0(u)) < L(u)$
3. $L(u) \leq L(P_1(u))$
4. $L(u) \leq L(P_2(u))$
5. $L_{\max_2}(u) < L(P_1(u))$
6. $L_{\max_1}(u) < L(P_2(u))$
7. $\min(L_{\max_1}(u), L_{\max_2}(u)) < L(u)$
8. $L(v_1) \geq L_{\max_2}(v_2)$

$L(v)$ est appelé le *niveau* du sommet v. La *hauteur* d'un nivelage est égale à $max(L(v), v \in V(G))$. La condition 1 fixe le niveau de la racine.

Les conditions 2,3 et 4 définissent le niveau d'un sommet par rapport aux niveaux de ses parents dans les arbres T_0, T_1 et T_2.

La condition 4 assure que chaque enfant v d'un sommet u dans T_2 est placé sur un niveau plus bas que le niveau $L(P_1(u)) - 1$. De plus, comme v est un enfant de u dans T_2, v doit être situé sur un niveau plus petit que $L(u)$ (cf. condition 4).

Par transitivité, on peut remarquer que ce qui est vrai pour les enfants de T_2 est aussi vrai pour chaque descendant v de u dans T_2 : $L(v) < L(u)$ et $L(v) < L(P_1(u)) - 1)$. Donc l'arête $(u, P_1(u))$ peut être dessinée sans croisement. De manière symétrique, la condition 6 assure que l'arête $(u, P_2(u))$ peut être aussi dessinée sans croisement.

La condition 7 assure qu'un sommet u ne peut pas avoir simultanément sur son niveau un de ses enfants dans T_1 et un de ses enfants dans T_2. En d'autres termes si u possède un enfant dans T_1 sur un niveau k et un enfant dans T_2 aussi sur un même niveau k, alors $L(u) > k$.

Enfin, la dernière condition garantit que l'arête externe (v_1, v_2) peut être dessinée sans croisement.

5.4 Dessin lignes brisées d'une stratification-faible d'un réaliseur

Pour chaque sommet v, on note $x_L(v)$ (resp. $x_R(v)$) l'abscisse de la feuille la plus à gauche (resp. la plus à droite) du sous-arbre issu de v.

L'algorithme suivant calcule les abscisses des sommets internes ainsi que les coordonnées des brisures des arêtes à partir d'une stratification-faible d'un réaliseur. Les coordonnées ainsi calculées donnent un dessin planaire lignes brisées du graphe.

Algorithme 7 Dessin lignes brisées.

pour chaque sommet u de G, $y(u) \leftarrow L(u)$
Associer à chaque feuille de T_0 une colonne de la gauche vers la droite.
pour chaque nœud interne u de T_0 **faire**
  si $L(u) = L_{\max_2}(u)$ **alors**
    $x(u) \leftarrow x_R(u)$
  sinon
    $x(u) \leftarrow x_L(u)$
  fin si
  pour chaque enfant v de u dans T_0 **faire**
    ajouter la brisure $(x(v), y(u)+1)$ à l'arête (u, v) si nécessaire.
  fin pour
  ajouter la brisure $(x_L(u), y(u))$ à l'arête $(v, P_2(u))$ si nécessaire.
  ajouter la brisure $(x_R(u), y(u))$ à l'arête $(v, P_1(u))$ si nécessaire.
fin pour
ajouter la brisure $x_L(v_0), y(v_0)+1$ à l'arête (v_0, v_1)

Lemme 5.4.1. *Soit G un graphe plan maximal possédant n sommets. Soit L une stratification-faible d'un réaliseur $R = (T_0, T_1, T_2)$ de G. Soit p le nombre de feuilles de T_0 et k la hauteur de L. L'algorithme 7 calcule un dessin planaire lignes brisées de G en temps linéaire sur une grille $(p+1) \times (k)$. De plus, le dessin obtenu possède au plus $n-2$ brisures et chaque arête possède au plus une brisure.*

Démonstration. Les conditions 5 et 6 de la définition 5.3.1 assurent que pour chaque sommet u, on peut dessiner les arêtes vers $P_1(u)$ et $P_2(u)$ sans risquer de croisement.

La condition 7 assure que u ne peut pas avoir simultanément deux enfants, un dans T_1 et un dans T_2, sur son niveau. Donc si le sommet u possède un enfant v dans T_2 sur son niveau, l'abscisse de u est fixée à $x_R(u)$. Ceci permet d'éviter un chevauchement entre l'arête $(u, P_1(u))$ et l'arête (v, u) (voir figure 61). Dans le cas ou u ne possède pas d'enfant dans T_2 sur son niveau, l'abscisse de u est fixée à $x_L(u)$ (même si u ne possède pas d'enfant dans T_1). Ceci permet d'éviter un éventuel chevauchement entre l'arête (u, P_2) et une arête entre u et un de ses enfants dans T_1 (voir figure 60).

La condition 8 assure que l'arête (v_1, v_2) peut être dessinée en ligne droite. Puisque v_0 ne possède pas d'enfant dans T_2, $x(v_0) = x_L(v_0)$ et donc l'arête (v_0, v_2) peut aussi être dessinée en ligne droite.

Maintenant, regardons quelles sont les arêtes pouvant être dessinées en ligne droite.

- Si u est une feuille de T_0, alors les arêtes $(u, P_1(u))$ et $(u, P_2(u))$ sont dessinées en ligne droite.
- Si If $x(u) = x_L(u)$ (resp. $x(u) = x_R(u)$) alors l'arête $(u, P_2(u))$ (resp. $(u, P_1(u))$) et l'arête partant de u vers le premier enfant de u (resp. dernier enfant de u) dans T_0 sont dessinées en ligne droite.

Toutes les autres arêtes possèdent exactement une brisure. Comme nous l'avons vue dans la section précédente, le choix des coordonnées des brisures assure un dessin lignes brisées sans chevauchement ni croisement.

Le nombre d'arêtes brisées partant d'un sommet u et allant vers un de ses enfants dans T_0 ou vers $P_1(u)$ ou vers $P_2(u)$ est borné par le nombre de ses enfants dans T_0 (au moins deux des arêtes ainsi considérées sont des lignes droites). Donc le nombre de brisures sur les arêtes internes est borné par le nombre d'arêtes dans T_0 : $n - 3$. De plus, les arêtes (v_0, v_1) et (v_1, v_2) sont des lignes droites et l'arête (v_0, v_2) peut être brisée. Donc au final le nombre de brisures nécessaires est au plus de $n - 2$. □

5.5 Algorithme de calcul d'une stratification-faible d'un réaliseur

Dans cette section, nous présentons un algorithme qui construit une stratification-faible d'un réaliseur. Dans un premier temps, tous les sommets de G sont placés sur le niveau 0. Ensuite, l'algorithme traite chaque sommet interne de G dans l'ordre postfixe trigonométrique de T_2. Le traitement d'un sommet v est le suivant : on applique de manière séquentielle des *règles* de réévaluation des niveaux (voir algorithme 8).

On peut remarquer que ces règles ne font qu'augmenter le niveau des sommets v, $P_1(v)$ et $P_2(v)$. Quand les règles 1 et 2 ont été appliquées sur le sommet v, le niveau de v restera inchangé jusqu'à la fin de l'exécution de l'algorithme. On dit donc qu'une fois appliquées les règles 1 et 2 sur le sommet v, v devient *fixé*.

Lemme 5.5.1. *Soit G un graphe plan maximal. Soit $R = (T_0, T_1, T_2)$ un réaliseur de G. L'algorithme 8 calcule une stratification-faible de R en temps linéaire.*

Démonstration. Le sommet v_0 est fixé sur le niveau 0 donc la condition 1 de la définition 5.3.1 est vérifiée. Les sommets sont fixés dans l'ordre postfixe trigonométrique de T_2. Donc un sommet u est traité quand ses enfants dans T_1 et T_2 ainsi que son père dans T_0 ont été fixés (voir propriété 1.2.3). Comme le premier sommet traité dans la boucle principale est un enfant de v_0, une feuille de T_1 et une feuille de T_2, les conditions 2, 3, 4, 5, 6 et 6 de la définition 5.3.1 sont vérifiées pour le sommet u.

Algorithme 8 Construction d'une stratification-faible.

pour chaque sommet u de G **faire**
  $L(u) \leftarrow 0$
fin pour
pour chaque sommet interne u de G dans l'ordre trigonométrique postfixe de T_2 **appliquer les règles :**
  1. $L(u) \leftarrow \max(L(u), L(P_0(u)) + 1)$
  2. $L(u) \leftarrow \max(L(u), \min(L_{\max_1}(u), L_{\max_2}(u)) + 1)$
  3.a $L(P_2(u)) \leftarrow \max(L(P_2(u)), L(u))$
  3.b $L(P_1(u)) \leftarrow \max(L(P_1(u)), L(u))$
  4.a $L(P_2(u)) \leftarrow \max(L(P_2(u)), L_{\max_1}(u) + 1)$
  4.b $L(P_1(u)) \leftarrow \max(L(P_1(u)), L_{\max_2}(u) + 1)$
fin pour
5. $L(v_1) \leftarrow \max(L(v_1), L_{\max_2}(v_2) + 1)$

Supposons maintenant que les conditions de la définition 5.3.1 sont vérifiées pour les m premiers sommets fixés. Soit u le prochain sommet à être fixé. Montrons que ces conditions sont vérifiées aussi pour u lorsqu'il devient fixé.

La règle 1 assure que le sommet u est sur un niveau plus haut que celui de son parent dans T_0. Donc après l'application de la règle 1, la condition 2 est vérifiée pour le sommet u.

La règle 2 assure que si un sommet u possède 2 enfants, un dans T_1 et un dans T_2 situés sur le même niveau que lui, alors $L(u)$ est incrémenté. Donc après l'application de la règle 2, la condition 6 est vérifiée pour le sommet u.

La règle 3.a (resp. 3.b) assure que le sommet $P_2(u)$ (resp. $P_1(u)$) est sur un niveau plus haut que u. Ceci correspond à la condition 3 (resp. 4) de la définition 5.3.1.

De manière analogue, les règles 4.a et 4.b assure que les conditions 5 et 6 sont satisfaites pour le sommet u.

Pendant le traitement du sommet u, les niveaux de $u, P_1(u)$ et $P_2(u)$ ne sont pas diminués. Donc les conditions 2, 3, 4, 5, 6 et 7 restent vérifiées pour les m premiers sommets fixés.

Donc à la fin de la boucle principale, les conditions 2 à 7 sont vérifiées pour tous les sommets internes de G. De plus, la dernière étape de l'algorithme assure que la condition 8 est vérifiée. Donc à la fin de l'exécution de l'algorithme, L est une stratification-faible du réaliseur R.

L'algorithme traite chaque sommet une fois. Le traitement d'un sommet s'effectue en temps constant. Donc l'algorithme est linéaire. □

Fait 5.5.1. *A chaque étape de l'algorithme 8, pour tout niveau* $i < max\{L(u), u \in V(G)\}$*, il existe un sommet fixé* v*, tel que* $L(v) = i$*.*

En d'autres termes, la stratification-faible ainsi construite ne contient aucun niveau vide.

Lemme 5.5.2. *Soit R un réaliseur. Soit L une stratification-faible de R générée par l'algorithme 8. Si v est une feuille de T_0, alors il existe $u \neq v$ tel que $L(v) = L(u)$. De plus, cette propriété est aussi vraie pour le sommet v_2.*

Démonstration. Puisque v_2 n'appartient ni à T_0 ni à T_1, les seules règles qui peuvent changer son niveau sont les règles 3.a et 4.a. Lors de l'application de la règle 5, soit $L(v_2) = L_{\max_2}(v_2)$ et alors v_2 est sur le même niveau que l'un de ses enfants dans T_2, soit $L(v_2) = L_{\max_2}(v_2) + 1$ et alors v_1 est sur le même niveau que v_2. Dans tous les cas, v_2 n'est pas seul sur son niveau.

Soit un sommet v, feuille de T_0. Quand v devient fixé, on peut distinguer deux configurations :

- Cas 1 : il y a un sommet u tel que $L(u) > L(v)$. Tous les niveaux plus bas que $L(u)$ contiennent au moins un sommet fixé (cf. Fait 5.5.1). Comme v n'est pas encore fixé, $L(v)$ contient déjà un sommet fixé.
- Cas 2 : L ne contient pas de sommet plus haut que v au moment où v devient fixée. Soit u le premier sommet à apparaître sur le niveau $L(v) + 1$. Ce sommet apparaît donc sur $L(v) + 1$ après que v devienne fixé. Un tel sommet u existe toujours car à la fin de l'algorithme, v_1 est un sommet situé un niveau plus haut que ceux où se trouvent les sommets internes de G.

  Considérons la règle qui a placé le sommet u sur le niveau $L(v) + 1$.

  - Cas 2.1 : règle 1. Alors $L(P_0(u)) = L(v)$. Puisque v est une feuille de T_0, $P_0(u)$ est différent de v.
  - Cas 2.2 : règle 2. Il y a donc au moins 2 sommets fixés sur le niveau $L(v)$: un enfant de u dans T_1 et un enfant de u dans T_2.
  - Cas 2.3 : règle 3.a (resp. 3.b). Le sommet u possède alors un enfant fixé w dans T_2 (resp. dans T_1) tel que $L(w) = L(v) + 1$. Ceci est en contradiction avec le fait que u est le premier sommet fixé sur le niveau $L(v) + 1$.
  - Cas 2.4 : règle 4.a (resp. 4.b). Dans ce cas si $u = P_1(v)$ (resp. $u = P_2(v)$) alors $L(v) = L_{\max_2}(v)$ (resp. $L(v) = L_{\max_1}(v)$). Donc v possède un enfant w fixé dans T_2 (resp. T_1) tel que $L(v) = L(w)$.
  - Cas 2.5 : règle 5. Alors on a $u = v_1$ et $L(v_2) = L_{\max_2}(v_2)$. Donc $L(v) = L(v_2)$.

Comme nous venons de le voir, si le sommet v est une feuille de T_0 alors v n'est pas tout seul sur son niveau. □

Lemme 5.5.3. *L'algorithme 8 calcule en temps linéaire, une stratification-faible d'un réaliseur $R = (T_0, T_1, T_2)$, de hauteur au plus $n - \lfloor \frac{p}{2} \rfloor - 1$ où p désigne le nombre de feuilles de l'arbre T_0.*

Démonstration. Comme nous l'avons vu, il y au moins un sommet par niveau (voir Fait 5.5.1). De plus, une feuille de T_0 ne peut être seule sur son niveau (lemme 5.5.2). Donc dans le pire des cas, c'est à dire celui où la stratification-faible est la plus haute,

le dernier niveau contient uniquement v_1, et chaque autre niveau contient soit un seul nœud interne de T_0, soit deux feuilles de T_0, soit une feuille de T_0 et v_2. Ce qui nous donne $n-2-p$ niveau possédant un nœud interne de T_0, $\lfloor \frac{p+1}{2} \rfloor$ niveaux pour les feuilles de T_0 et v_2 et un dernier niveau pour v_1. Au final, la hauteur de la stratification-faible obtenue par le précédant algorithme est au plus $n - \lfloor \frac{p}{2} \rfloor - 1$. □

Théorème 5.5.1. *Soit G un graphe plan à n sommets. Le graphe G admet un dessin lignes brisées sur une grille de surface au plus $\frac{4(n-1)^2}{9}$ et de largeur au plus $\lfloor \frac{2(n-1)}{3} \rfloor$. De plus, chaque arête possède au plus une brisure et globalement le dessin possède au plus $n-2$ brisures.*

Démonstration. Soit G' une triangulation planaire de G. Elle peut être obtenue en temps linéaire. Soit $R = (T_0, T_1, T_2)$ un réaliseur de G'.

Soit T_i un des arbres de R qui possède au plus $\lfloor \frac{2n-5}{3} \rfloor$ feuilles. Le corollaire 2.3.1 nous assure qu'un tel arbre existe. Soit $R' = (T'_0 = T_i, T'_1 = T_{i+1}, T'_2 = T_{i+2})$ le réaliseur obtenu par permutation circulaire à partir de R. Le réaliseur R' est aussi un réaliseur de G'. Soit p le nombre de feuilles de T'_0.

L'algorithme 8 calcule en temps linéaire une stratification-faible de R' de hauteur $n - \lfloor \frac{p}{2} \rfloor - 1$. En utilisant cette stratification-faible, l'algorithme 7 calcule un dessin lignes brisées de G sur une grille $(p+1) \times (n - \lfloor \frac{p}{2} \rfloor - 1)$ avec au plus $n-2$ brisures. Puisque $p \leq \lfloor \frac{2n-5}{3} \rfloor$, la largeur du dessin est donc d'au plus $\lfloor \frac{2(n-1)}{3} \rfloor$. Et donc la surface de la grille est d'au plus $\frac{4(n-1)^2}{9}$. La propriété 5.1.1 assure qu'une telle largeur et une telle surface sont nécessaires pour dessiner certains graphes plans de taille n. Donc l'algorithme 7 produit des dessins de surfaces et de largeurs optimales pour la classe des graphes plans. □

La figure 62 représente un exemple de dessin lignes brisées obtenu à l'aide de l'algorithme présenté.

5.6 Conclusion

L'algorithme que nous avons présenté produit des dessins lignes brisées sur des grilles de surface et de largeur optimales pour la classe des graphes plans. Les dessins obtenus possèdent au plus $n-2$ brisures. Dans la pratique, le nombre de brisures effectivement utilisé est plus petit. On peut se demander s'il est possible d'obtenir des dessins similaires avec un nombre nettement plus faible de brisures (au plus $n/2$ brisures, par exemple).

Dans le cas de graphes plans non maximaux, il pourrait être intéressant d'évaluer la taille de la grille et le nombre de brisures nécessaires en fonction du nombre d'arêtes.

Enfin, nous avons vu que le dessin était de largeur optimale pour la classe des graphes plans (i.e. la face extérieure est fixée). Dans le cas de graphes planaires, on peut espérer obtenir des dessins sur des grilles de largeur plus petite. Par exemple, en

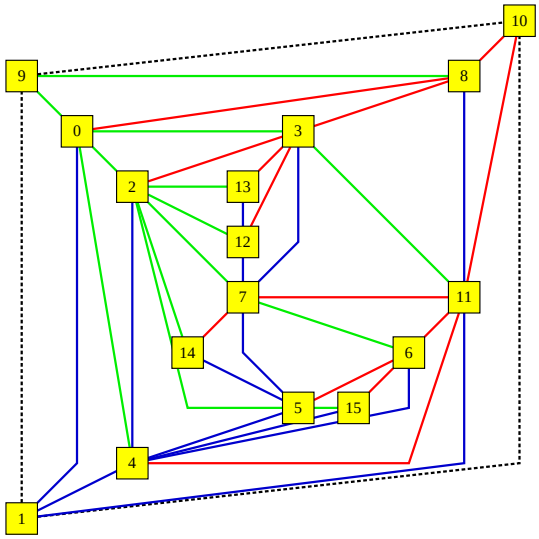

Figure 62 – Exemple de dessin lignes brisées obtenu par l'algorithme 7. Le graphe dessiné possède 16 sommets. Le dessin est effectué sur une grille de taille 9×9 et contient 7 brisures.

choisissant judicieusement la face extérieure, le graphe H_n de la figure 59 peut être dessiné sur une grille de largeur $n/3$ et non $2n/3$.

Chapitre 6

Une majoration du nombre de graphes planaires à l'aide des réaliseurs

Introduction

Quelle quantité d'information peut contenir un graphe planaire à n sommets ? La réponse à cette question est hautement liée au nombre de graphes planaires à n sommets. Dénombrer les graphes planaires (non isomorphes) à n sommets est un problème bien connu, mais qui malheureusement est toujours non-résolu (cf. [1]). Il n'existe ni de formule clause, ni d'estimation asymptotique, ni même d'estimation asymptotique sur le log de ce nombre. Une estimation asymptotique sur le log donnerait une borne sur le nombre de bits aléatoires nécessaires pour générer aléatoirement et uniformément un graphe planaire (pas forcément en temps polynomial).

La génération d'objets aléatoires est un outil important pour l'analyse de la complexité moyenne d'algorithmes, ou le test d'algorithmes sur des cas typiques. A l'inverse des graphes aléatoires [41], on ne connaît que très peu de choses sur les graphes planaires aléatoires. En effet, l'ajout d'une arête dans un graphe planaire dépend fortement de l'emplacement des autres arêtes. Les graphes plans aléatoires ont été étudiés avec succès. Schaeffer [85], puis Banderier, Flajolet, Schaeffer et Soria [5] ont montré comment générer en temps polynomial plusieurs familles de cartes planaires, e.g. cartes planaires triconnexes. Malheureusement, cette génération ne donne que très peu d'informations sur les graphes planaires aléatoires car il y a de nombreuses manières de plonger un graphe dans le plan. Néanmoins, certaines familles de graphes planaires peuvent être générées de manière aléatoire : les arbres [2], les graphes planaires-extérieurs ("outerplanar" en anglais) maximaux [40, 6] (i.e. les triangulations d'un polygone), et plus récemment les graphes planaires-extérieurs étiquetés ou non [11]

En plus de l'aspect "combinatoire" et "génération aléatoire", une attention particulière est accordée en informatique aux représentations "*efficaces*" des objets discrets. Efficace signifie d'une part que la représentation occupe peu de place, et d'autre part que le temps nécessaire au calcul d'une telle représentation est polynomial (faible nombre

de bits et calcul rapide). Une manipulation rapide d'une telle représentation ainsi qu'un accès facile à des fragments d'information sont aussi des objectifs poursuivis. Au moins deux champs d'applications sont concernés par la représentation de graphes planaires : l'imagerie numérique et les réseaux.

La discrétisation d'un objet 3D donne une liste de coordonnées dans l'espace, ainsi qu'un ensemble de relations d'adjacences. Dans le cas d'objets convexes, cet ensemble de relations d'adjacences est un graphe planaire non étiqueté. En général, des graphes planaires dont les faces sont des triangles ou des quadrilatères sont utilisés pour discrétiser la surface de tels objets. Un algorithme de compression est appliqué sur le graphe ainsi obtenu. Les performances sont mesurées par le nombre moyen de bits par nœud et par arête. Elles sont exprimées en fonction de tests menés sur des générateurs d'exemples types ou sur des exemples de référence [66], faute d'avoir un "bon" générateur de graphes planaires aléatoires. Par exemple, King et Rossignac [67, 84] ont donné un algorithme de compression des triangulations qui garantit un codage en $3,67$ bits par sommet, le rapport optimal étant $\log_2(256/27) \approx 3,24$ bits par sommet (voir formule énumération des graphes plans maximaux [91], théorème 1.1.3).

Une table de routage est une structure de données associée à chaque nœud et qui indique, en fonction de la destination d'un message entrant, le port de sortie qui doit être utilisé pour atteindre le destinataire. Le principal objectif d'un système de routage est de minimiser la taille des tables de routage tout en maintenant les trajets aussi courts que possible. Les tables de routage construites pour un réseau ont été étudiées dans le cas de réseaux planaires [52, 54, 72, 89]. Le graphe sous-jacent est pré-calculé afin d'optimiser les tables de routage.

La stratégie utilisée par Gavoille et Hanusse [54], basée sur les plongements en k-page et améliorée par Lu [72] avec les arbres recouvrants ordonnés, démontre qu'une représentation compacte de graphes planaires aide à la construction de tables de routage compactes, plus particulièrement dans le cas d'un routage de plus court chemin.

Travaux précédents

Pour les raisons que l'on vient d'exposer, on cherche à représenter de manière succincte les graphes planaires à n sommets et m arêtes. Turán [90] a proposé un codage en $4m$ bits, qui fut amélioré plus tard par Keeler et Westbrook [65] pour obtenir un codage en $3,58m$ bits. Munro et Raman [76] ont proposé un codage en $2m + 8n$ bits s'appuyant sur les plongements en 4-page des graphes planaires (voir [99]). Dans une série d'articles, Lu et al. [26, 23] ont proposé un codage plus fin en $4m/3 + 5n$ bits, à l'aide des arbres recouvrants ordonnés. Comme nous l'avons vu dans le chapitre 3, il existe des codages pour les graphes planaires maximaux en $4n$ bits [26, 84, 14]. Une amélioration des techniques utilisées dans [84] donne un codage en $3,67n$ bits des graphes planaires maximaux [67], calculable en temps linéaire. Finalement, He, Kao et Lu [63] ont montré qu'un codage optimal des graphes planaires, maximaux ou non,

pouvait être obtenu avec une complexité en temps de $O(n\log(n))$. Pour cela ils utilisent une décomposition récursive du graphe à l'aide de séparateurs et un algorithme de codage exponentiel pour les dernières composantes de taille sous-logarithmique. Cependant la constante cachée derrière la notation "grand O" peut être limitante dans la pratique. De plus, l'implémentation de l'algorithme de codage nécessite l'implementation d'algorithmes complexes tels que ceux de reconnaissance de graphes isomorphes et de recherche de séparateurs [71]. Récemment, la complexité de cet algorithme de codage a été améliorée afin de la rendre linéaire [73]. Bien que la longueur du codage soit optimale, l'approche de [63] ne donne pas une borne explicite sur le nombre de bits utilisés dans cette représentation.

Si l'on s'intéresse uniquement au nombre de graphes planaires ou à certaines propriétés statistiques des graphes planaires (à quoi ressemble un graphe planaire aléatoire : nombre d'arêtes, connexité, etc.) d'autres outils peuvent être utilisés. Denise, Vasconcellos et Welsh [31] ont donné une chaîne de Markov dans l'espace de tous les graphes planaires étiquetés dont la distribution limite est uniforme. Leurs expérimentations ont montré qu'un graphe planaire aléatoire possède approximativement $2n$ arêtes. De plus, un tel graphe est généralement connexe mais pas biconnexe. Bien que leur chaîne de Markov converge vers une distribution uniforme, il n'est pas prouvé que la distribution obtenue après un nombre polynomial d'étapes est suffisamment proche de la distribution uniforme. Il est toutefois prouvé que presque tous les graphes planaires étiquetés possèdent au moins $3n/2$ arêtes et que le nombre $p(n)$ de graphes planaires non étiquetés vérifie le fait que $\frac{1}{n}\log_2(p(n))$ tende vers une constante γ telle que $\log_2(\frac{256}{27}) \leq \gamma \leq \log_2(\frac{256}{27}) + 3$. Ces bornes sur γ viennent de la formule de Tutte (voir théorème 1.1.3) : tous les graphes planaires maximaux sont des graphes planaires et tous les graphes planaires sont des sous-graphes des graphes planaires maximaux et par conséquent il y a 2^{3n-6} sous-ensembles d'arêtes possibles. Il y a aussi au plus $n!2^{\gamma n+o(n)}$ graphes planaires étiquetés puisqu'il y a au plus $n!$ manières d'étiqueter les sommets d'un graphe.

Osthus, Prömel et Taraz [80] ont étudié les triangulations possibles des graphes planaires et ont montré qu'il y avait au plus $n!2^{5,22n+o(n)}$ graphes planaires étiquetés. Ils ont également montré que presque tous les graphes planaires étiquetés ont au plus $2,56n$ arêtes. De leur côté, Gerke et McDiarmid [55] ont montré que ces même graphes possèdent presque tous au moins $\frac{13n}{7} \approx 1,85n$ arêtes, améliorant ainsi l'ancienne borne supérieure donnée dans [31] : $1,5n$ arêtes. Ils ont également montré que presque tous les graphes planaires non étiquetés ont au plus $2,69n$ arêtes.

Utilisant des séries génératrices, Bender, Gao et Wormald [8] ont prouvé que le nombre de graphes planaires 2-connexes étiquetés était asymptotiquement $n!2^{4,71n+O(\log(n))}$. Notons que l'énumération de cartes planaires donne une borne supérieure sur le nombre de graphes planaires. Récemment, Bousquet-Mélou [20], a montré que le nombre de cartes planaires simples était asymptotiquement $2^{5,098n+O(\log(n))}$, donnant ainsi une borne sur le nombre de graphes planaires non étiquetés et étiquetés $(n!2^{5,098n+O(\log(n))})$.

Résultats présentés

Dans ce chapitre nous montrons une borne supérieure de $2^{5,007n+O(\log(n))}$ sur le nombre de graphes planaires non étiquetés. Ce résultat implique que le nombre de graphes planaires étiquetés est d'au plus $n!2^{5,007n+O(\log(n))}$, améliorant ainsi la borne donnée dans [80].

Comme notre borne peut être paramétrée en fonction du nombre d'arêtes, nous pouvons montrer en utilisant la borne de [8] que la plupart des graphes planaires non étiquetés possèdent au moins $1,70n$ arêtes et au plus $2,54n$ arêtes, améliorant l'ancienne borne supérieure de $2,69n$ arêtes [80]. De plus, ce résultat est également vrai pour les graphes planaires étiquetés (améliorant légèrement l'ancienne borne, $2,56n$ [80]), connexes étiquetés et connexes non étiquetés.

Mis à part l'aspect fondamental de l'énumération des graphes planaires, notre technique s'appuie sur une représentation *explicite*, relativement simple, et calculable en temps linéaire. De plus, nous donnons un algorithme linéaire de codage en $3,37n$ bits des graphes plans maximaux et en $5,03n$ bits pour les graphes planaires. Notre représentation des graphes plans maximaux améliore la compression "Edgebreaker" [67], et il ne fait pas de doute que notre construction explicite d'un graphe planaire peut être utilisée pour des problèmes de routage dans les réseaux, en particulier pour améliorer le résultat de Lu [72].

Organisation du chapitre

La section 6.1 présente la notion d'*arbre recouvrant bien-ordonné*, une spécialisation des arbres recouvrants ordonnés introduite dans [26] et une généralisation des réaliseurs minimaux (voir chapitre 1, section 1.3). Nous présentons également un algorithme permettant de construire un tel arbre recouvrant en temps linéaire.

Dans la section 6.2 nous montrons que si G est connexe, le réaliseur minimal S d'un super-graphe particulier de G, appelé par la suite *super-triangulation*, possède la propriété que étant donné S et un sommet v, les arêtes de l'arbre recouvrant bien-ordonné de S et v sont également dans G. Cette propriété ainsi que quelques autres nous permettrons d'obtenir un codage compact de G. En effet, une des propriétés de $S = (T_0, T_1, T_2)$ est que toutes les arêtes de T_0 sont aussi des arêtes de G.

Dans la section 6.3 nous présentons un codage d'un graphe planaire G à l'aide de 8 chaînes binaires de densité différente (le rapport entre le nombre de "1" et la longueur de la chaîne) : 7 de ces chaînes servent à représenter la super-triangulation S (plus précisément 5 pour coder l'arbre T_2 et 2 pour coder les arêtes pertinentes) et une pour coder les arêtes manquantes. Chaque chaîne peut-être compressée à l'aide de l'algorithme de Pagh [81] (nous donnons également un algorithme linéaire et plus simple pour effectuer cette compression). Ceci nous permet de coder chaque chaîne en $\log_2(\binom{n}{k}) + o(n)$ bits où n est la longueur la chaîne et k le nombre de "1".

Enfin la section 6.4 nous analysons la taille de notre représentation en fonction du nombre de feuilles de T_2 dans un premier temps puis en fonction du nombre d'arêtes

du graphe. Cette analyse nous montre que notre codage utilise au plus $3,37n$ bits pour coder un graphe plan maximal et $5,03n$ bits pour un graphe planaire. Un codage en $5,007n$ bits est atteint à l'aide d'un codage un peu plus sophistiqué de S. Enfin nous montrons également que presque tous les graphes planaires, connexes ou non, étiquetés ou non, possèdent entre $1,70n$ et $2,54n$ arêtes.

6.1 Arbre recouvrant bien-ordonné

6.1.1 Définition

Nous proposons ici une spécialisation de la notion d'arbre recouvrant ordonné introduite dans [26] (voir chapitre 1 section 1.4). Un sommet u_i est *bien-ordonné* s'il est ordonné (voir définition 1.4.2) et si la première arête de $(u_i, u_j) \in B_>(u_i)$, si elle existe, est telle que le parent de u_j soit un ancêtre de u_i. Tout naturellement, un arbre recouvrant T de H est dit *bien-ordonné* si tous ses sommets sont bien-ordonnés. De plus, on dira que (T, H) est une paire bien-ordonnée. Remarquons qu'un arbre ordonné (bien ou pas) doit être recouvrant. Observons également qu'une arête adjacente à un sommet de T est soit dans T soit elle est non-apparentée. En particulier, si une arête de H est apparentée (i.e. une extrémité est descendante de l'autre dans T), alors elle appartient à T. Il s'en suit que tous les voisins de la racine de T sont dans T.

La figure 63 illustre les définitions ci-dessus.

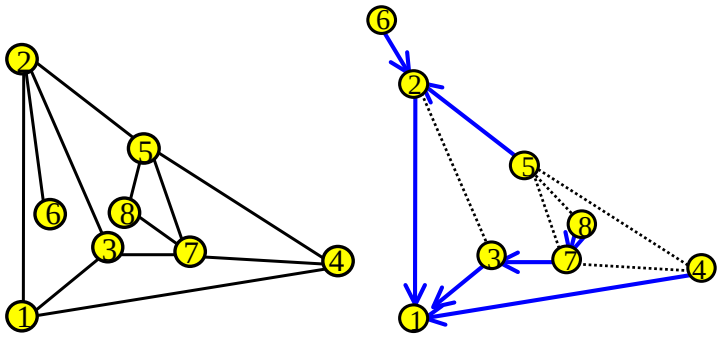

Figure 63 – Le graphe plan H du graphe G (à gauche), n'admet pas d'arbre recouvrant bien-ordonné. Le graphe plan H' de G (à droite), admet quant à lui un arbre recouvrant bien-ordonné T. La paire (T, H') est une paire bien-ordonnée de G.

Enfin, remarquons que si $R = (T_0, T_1, T_2)$ est un réaliseur de G, alors $\bar{T}_0$ est un arbre ordonné. Remarquons également que si R est un réaliseur minimal, alors $\bar{T}_0$, $\bar{T}_1$ et $\bar{T}_2$ sont des arbres bien-ordonnés.

Comme nous l'avons vu les arbres ordonnés sont une généralisation naturelle des réaliseurs aux graphes planaires connexes. De manière analogue, les arbres recouvrants bien-ordonnés sont une généralisation des réaliseurs minimaux. Cette généralisation est également une spécialisation des arbres recouvrants ordonnés.

Lemme 6.1.1. *Tout graphe plan bien-ordonné enraciné en un sommet v_0 admet un unique arbre recouvrant bien-ordonné.*

Démonstration. Supposons que le graphe plan H admet deux arbres recouvrants bien-ordonnés T, T' enracinés en v. Soit $u_1, u_2, \ldots, u_n$ (resp. $u'_1, u'_2, \ldots, u'_n$) les sommets H dans l'ordre préfixe anti-trigonométrique de T (resp. T'). Soit u_i un sommet tel que l'ensemble de ses voisins dans T diffère de l'ensemble de ses voisins dans T' et tel que i soit minimum. Nous avons donc $u_t = u'_t$ pour $t \leq i$, et $B_C(u_i) \neq B'_C(u_i)$, où B'_C désigne les enfants de u_i dans T'.

Supposons sans perte de généralité, que $|B_C(v_i)| \leq |B'_C(v_i)|$ (le cas symétrique étant obtenu en échangeant le rôle de T et de T'). Remarquons que le cas où $B_<(v_i) = B_>(v_i) = \emptyset$ est impossible, car dans un tel cas $B_C(v_i)$ serait constitué de tous les voisins de v_i (à l'exception peut-être du parent de v_i) et donc $B_C(v_i) \neq B'_C(v_i)$ et $|B_C(v_i)| \leq |B'_C(v_i)|$ seraient incompatibles. Soit e_1 (resp. e_2) la première (resp. dernière) arête de $B_C(v_i)$ dans l'ordre anti-trigonométrique. Soit e une arête quelconque de $B'_C(v_i)$. Par la suite, $e_1 \leq e$ signifie que $e_1 = e$ ou que e_1 est avant e dans l'ordre anti-trigonométrique autour de v_i.

Montrons que $e_1 \leq e$. Ceci est clairement vrai si $B_<(v_i) = \emptyset$. Si $B_<(v_i) \neq \emptyset$, alors considérons une arête $(v_i, v_h) \in B_>(v_i)$. Alors, l'arête (v_i, v_h) n'appartient pas à $B'_C(v_i)$. En fait, comme $h < i$, le chemin de v_h à v_i dans T existe également dans T', et l'arête (v_i, v_h) de T' créerait un cycle dans T'. Donc $e_1 \geq e$.

Si $B_>(v_i) = \emptyset$ alors $e \leq e_2$. Donc $e_1 \leq e \leq e_2$, ce qui est incompatible avec le fait que $B'_C(v_i)$ et $B_C(v_i)$ sont des bloques d'arêtes consécutives tels que $|B_C(v_i)| \leq |B'_C(v_i)|$. Nous avons donc $B_>(v_i) \neq \emptyset$.

Soit (v_i, v_j) la première arête de $B_>(v_i)$ dans l'ordre anti-trigonométrique. Alors, l'arête (v_i, v_j) n'appartient pas à $B'_C(v_i)$. En fait, comme T est bien-ordonné, le parent de v_j dans T, appelons-le v_k, est un ancêtre de v_i et donc $k < i$. Comme $B_C(v_k) = B'_C(v_k)$ pour $k < i$, l'arête (v_k, v_j) est dans T'. Donc le chemin de v_j à v_i dans T est également dans T' et l'arête (v_i, v_j) de T' créerait un cycle dans T'. On en déduit donc que toute arête $e \in B'_C(v_i)$ est telle qui $(v_i, v_j) \leq e$ et $e \neq (v_i, v_j)$. comme le chemin de v_i à v_j est dans T est également dans T', le sommet v_j est après v_i dans l'ordre préfixe anti-trigonométrique de T'. Donc le sommet v_i n'est pas bien-ordonné dans T' : d'où la contradiction.

□

Propriété 6.1.1. *Soit $R = (T_0, T_1, T_2)$ un réaliseur. Le réaliseur R est un réaliseur minimal si et seulement si $\bar{T}_i$ est bien-ordonné pour chaque $i \in \{0, 1, 2\}$.*

Démonstration. Supposons que R possède un cw-triangle (u, v, w), où $w = P_1(v)$, et que $\bar{T}_0$ soit bien-ordonné. Comme $w = P_1(v)$, alors le plus petit ancêtre commun à v et w dans $\bar{T}_0$ est u. On en déduit donc que $v = P_2(u)$ est un descendant de u dans $\bar{T}_0$, ce qui est en contradiction avec la propriété 1.2.2.

Supposons maintenant que $\bar{T}_0$ ne soit pas bien-ordonné. Ceci implique qu'il existe une arête (u_p, u_j), avec $u_p = P_1(u_j)$ telle que le plus petit ancêtre commun entre u_p et

u_j, appelons-le u_t, soit différent du sommet $u_i = P_0(u_j)$. Considérons le cycle C formé du chemin dans $\bar{T}_0$ entre u_p et u_j, et fermé par l'arête (u_p, u_j). Soit B la région connexe bornée de $\mathbb{R}^2 \setminus C$. Calculons maintenant $P_2(u_i)$ et $P_2(u_j)$. D'après la condition locale :

1. $P_2(u_j) \notin B$.
2. $P_j(u_j)$ ne peut appartenir au chemin entre u_j et u_t dans $\bar{T}_0$ avec l'arête $(u_j, P_2(u_j))$ à l'extérieur de B.
3. $P_2(u_j)$ ne peut appartenir au chemin allant de u_j à u_t dans $\bar{T}_0$ puisque $P_2(u_j)$ ne peut être un descendant de u_j dans $\bar{T}_0$ (voir propriété 1.2.2).

Donc, $P_2(u_j) \notin B \cup C$. Encore une fois, la condition locale nous montre que $P_2(u_i) \in B \cup C$. Il s'en suit que $P_2(u_i) = P_2(u_j)$ où $u_k = P_2(u_j)$ est un descendant de u_i tel que $i < k < j$ est impossible. Ceci implique donc que $\bar{T}_0$ ne possède pas la propriété branche. D'après la propriété 1.3.3 on en déduit également que R n'est pas minimal. Ceci termine donc notre preuve. □

6.1.2 Construction

Dans cette sous-section, nous prouvons le théorème suivant :

Théorème 6.1.1. *Soit G un graphe planaire connexe, et v un de ses sommets. Alors G admet une paire (H, T) bien-ordonnée, où T est enraciné en v. De plus, cette paire peut être calculée en temps linéaire.*

Démonstration. Nous donnons ici un algorithme simple pour construire une paire bien-ordonnée de G. Puis nous expliquons comment obtenir une implémentation linéaire.

Pour calculer la paire bien-ordonnée de G, nous commençons par calculer un graphe plan H de G tel que v se trouve sur la face extérieure de H. Ceci peut être effectué en temps linéaire [24]. Puis nous effectuons un parcours de H à partir de v dans le but de calculer un arbre couvrant bien-ordonné T. Toutefois, tous les graphes plans n'admettent pas un arbre recouvrant bien-ordonné (voir figure 63). Si, lors de la construction, T ne couvre pas tous les sommets de G alors la carte de G est modifiée et un nouveau parcours est lancé. Nous montrons qu'après un nombre fini de modifications de la carte H du graphe G le nombre de sommets couverts par T croit (T couvre plus de sommets) et que la construction converge vers une paire bien-ordonnée. Pour décrire plus précisément le parcours et la modification du graphe plan H nous avons besoin de quelques définitions.

Soit T un arbre de H (pas nécessairement recouvrant) enraciné en v. Un sommet est dit *libre*, s'il n'est pas couvert par T. Une arête est dite *libre*, si au moins une de ses extrémités est libre. Nous étendons la notion de sommet bien-ordonné de la manière suivante : un sommet est *partiellement bien-ordonné* (considérant H et T) s'il est bien-ordonné excepté le fait que les arêtes des blocs $B_<$ et $B_>$ (relatifs à T) autres que l'arête-frontale et l'arête dorsale peuvent être libres. L'arbre T est *partiellement bien-ordonné* si tous ses sommets sont partiellement bien-ordonnés. Un arbre bien-ordonné est un arbre partiellement bien-ordonné qui couvre H. Les quatre blocs d'arêtes autour d'un

sommet u partiellement bien-ordonné dans T sont notés $B_P(u,T), B_<(u,T), B_C(u,T)$ et $B_>(u,T)$.

Considérons l'algorithme 9. Soient $u_1 = v, u_2, \ldots, u_p$ les sommets de T (l'arbre

Algorithme 9 Parcours(H, v).

```
P : une pile
T : ensemble d'arêtes
T ← ∅
Mettre toutes les arêtes (u_l, v) dans T
Empiler tous les voisins u_l de v dans P (dans l'ordre trigonométrique)
tant que P ≠ ∅ faire
  u_a ← P.dépiler()
  pour chaque voisin libre u_l de u_a entre l'arête-frontale et l'arête-dorsale (dans
  l'ordre trigonométrique) faire
    Mettre (u_l, u_a) dans T
    P.empiler(u_l)
  fin pour
fin tant que
retourner T
```

résultat de l'algorithme 9) ordonnés dans l'ordre préfixe anti-trigonométrique. Considérons le sommet u_i et T_{u_i} l'arbre obtenu par l'algorithme 9 juste après le traitement du sommet u_i. L'observation clé est que $B_<(u_i, T_{u_i}) = B_<(u_i, T)$ et que $B_>(u_i, T_{u_i}) = B_>(u_i, T)$. En particulier l'arête-dorsale et l'arête-frontale de u_i dans T et dans T_{u_i} (si elles existent), sont les mêmes. Après le traitement de u_i, les arêtes autour de u_i dans T_{u_i} forment quatre blocs (éventuellement vides) : $B_P(u_i, T_{u_i}), B_<(u_i, T_{u_i}), B_C(u_i, T_{u_i})$ et $B_>(u_i, T_{u_i})$. Donc dans T, les blocs d'arêtes autour de u_i sont : $B_P(u_i, T), B_<(u_i, T), B_C(u_i, T)$ et $B_>(u_i, T)$. Pour montrer que T est partiellement bien-ordonné, il reste à montrer que si $(u_i, u_j) \in B_>(u_i, T)$ est une arête-frontale, alors le parent de u_j dans T est le plus petit ancêtre commun de u_i et u_j. Lorsque le sommet u_i est visité, les arêtes de l'arbre construit jusqu'à u_i (c'est à dire, $T_{u_{i-1}}$) sont soit entre des sommets u_t avec $t < i$, soit (u_k, u_j) avec $k < j$ et $j > i$. De plus, u_k appartient au chemin de u_i à v dans T. Donc l'arête-frontale (u_i, u_j) est telle que le parent de u_j est un ancêtre de u_i, le plus petit ancêtre commun. Donc T est partiellement ordonné.

Supposons que T ne couvre pas tous les sommets (si T couvre tous les sommets, le calcul est terminé). Soit u_i un sommet de T ayant des arêtes incidentes libres. Ces arêtes appartiennent aux blocs $B_<(u_i, T)$ et $B_>(u_i, T)$. Nous pouvons supposer, sans perte de généralité, que $B_>(u_i, T)$ contient une arête libre (voir figure 64). Le cas où u_i possède une arête libre dans $B_<(u_i, T)$ est symétrique. Soit $e_i = (u_i, u)$ la dernière arête libre du bloc $B_>(u_i, T)$. En fait, on peut choisir n'importe quelle arête qui soit la dernière, dans l'ordre anti-trigonométrique, d'un bloc d'arêtes libres dans $B_>(u_i, T)$. Par définition, $B_>(u_i, T)$ contient au moins une arête non-apparentée dans $B_>(u_i, T)$

avant e_i. Finalement soit $e_j = (u_j, w)$ la première arête libre de u_j avant e et telle qu'il n'y est pas d'arête non-apparentée entre e_j et e (donc u_j est la première arête du bloc d'arêtes libres juste avant e). Si une telle arête n'existe pas, alors on note $e_j = e$. En d'autres termes e, e_i et e_j sont choisies telles que les arêtes entre e et e_i autour de u_i et entre e et e_j autour de u_j forment un bloc maximal d'arêtes libres. Nous changeons le graphe plan H en appliquant un *basculement* :

1. Autour de u_i, e est déplacée et insérée juste après e_i (dans le sens anti-trigonométrique)
2. Autour de u_j, e est déplacée et insérée juste avant e_j (voir figure 64).

Par commodité, on dira que l'on effectue un basculement autour de e.

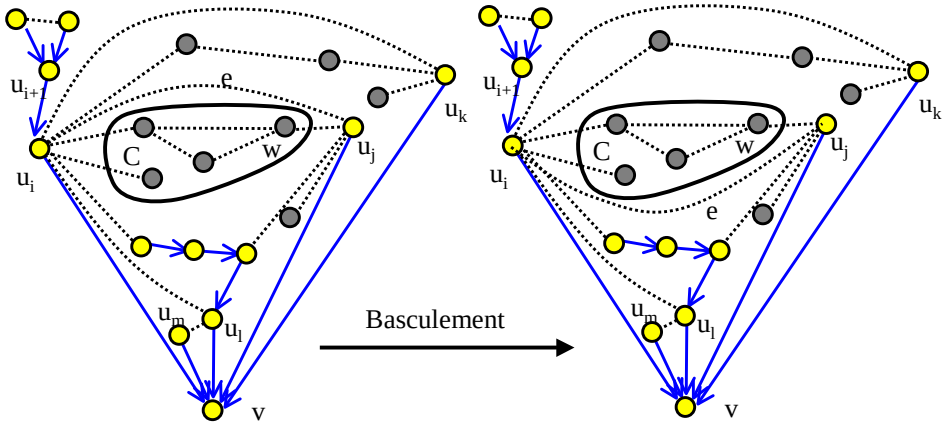

Figure 64 – Basculement du sous-graphe connexe composé de sommets libres par-dessus une arête critique.

Une fois le basculement effectué sur H, nous lançons de nouveau l'algorithme 9 sur le nouveau graphe plan H. Nous effectuons alternativement une exécution de l'algorithme 9 et un basculement jusqu'à obtenir un arbre couvrant de G. Pour compléter la validité de cet algorithme nous devons montrer que l'opération de basculement préserve la planarité et que l'arbre partiellement bien-ordonné obtenu après un appel à l'algorithme 9 converge bien vers un arbre recouvrant.

Soit X l'ensemble des sommets qui sont des extrémités des arêtes comprises entre e et e_i et entre e et e_j. Pour chaque $x \in X$, soit C_x la composante connexe contenant x dans le sous-graphe de H induit par les sommets libres. Soit $C = \cup_{x \in X} C_x$. Pour prouver que le basculement conserve la planarité, nous montrons que chaque chemin P de $y \in C$ à la racine v contient soit v_i, soit u_j. Soit C' le cycle composé du chemin dans T (l'arbre obtenu avant le basculement) allant de u_i à u_j et fermé par l'arête (u_i, u_j). Soit R la région connexe bornée de $\mathbb{R}^2 \setminus C'$. Supposons que P contienne un sommet $u_k \in T$ avec $u_k \in R \cup C'$,et $k \notin \{i, j\}$. Nous pouvons supposer, sans perte de généralité, que u_k est le premier sommet de T à partir de y dans P. Soit (u_k, z) l'arête libre de P. Nous avons $(u_k, z) \in B_<(u_k, T)$, ou $(u_k, z) \in B_>(u_k, T)$. Supposons que $(u_k, z) \in B_<(u_k, T)$, l'autre

cas étant symétrique. Comme $B_<(u_k, T) \neq \emptyset$, u_k possède une arête-dorsale. L'arête-dorsale est (u_k, u_i). En fait, si l'arête-dorsale est (u_k, u_t), avec t différent de i, alors le cycle composé du chemin dans T entre u_k et u_t, et fermé par l'arête (u_k, u_t), sépare C_z de $\{u_i, u_j\}$ puisqu'il est contenu dans R. Comme la suppression de l'arête (u_i, u_k) déconnecterait u_j de C_z, il doit exister une arête libre (u_i, s) pour un $s \in C_z$. Cette arête est après (u_i, u_k) dans l'ordre anti-trigonométrique. Ceci est en contradiction avec la définition de (u_i, u_j).

Donc la carte obtenue après le basculement est une carte planaire de G. On observe également que lorsque l'arête e est déplacée sous le sous-graphe C au cours d'un basculement, l'arbre calculé par l'algorithme 9 sur la nouvelle carte contient toutes les arêtes de T. En effet, C est connecté à $G \setminus C$ seulement par u_i et u_j. Donc, le déplacement de e ne peut créer d'arête non-apparentée (u_t, x) avec $x \in C$ et $t \notin \{i, j\}$. Supposons qu'après avoir déplacé e, l'algorithme 9 ne visite pas de nouveaux sommets. Alors, soit u_i possède une arête-frontale et une arête libre après cette arête-frontale, soit u_j contient une arête-dorsale et une arête libre avant cette arête-dorsale. En effet, si ce n'était pas le cas, tous les voisins de u_i et u_j seraient apparentés, et T contiendrait plus de sommets sans arêtes libres. Supposons donc que u_i possède une arête-frontale et une arête libre e'. Donc, dans au plus $\deg(u_i)$ basculements, un nouveau voisin libre de u_i (ou de u_j) est visité. Observons que pour chaque arête e il y a au plus deux basculements autour de e. Sur la figure 64, après le basculement autour de e, T est augmenté d'au moins une arête.

Pour finir cette preuve de l'algorithme de calcul d'une paire bien-ordonnée de G, analysons la complexité en temps. Il y a au plus $O(n)$ appels à l'algorithme 9 et au plus $O(n)$ basculements (pas plus de deux basculements autour de chaque arête), chacune de ces opérations s'effectuant en temps linéaire. Donc, une implémentation naïve donne un algorithme de calcul de paire bien-ordonnée quadratique.

Tout d'abord, on peut remarquer que l'opération de basculement peut être implémentée en $O(1)$, en utilisant des listes doublement chaînées pour stocker les listes d'adjacences des sommets, et pour chaque arête, un pointeur vers la position de l'arête dans la liste d'adjacences de chacune de ses extrémités. Cette représentation d'une carte est, par exemple déjà implémentée dans LEDA [74]. Comme l'arbre s'agrandit en ajoutant des arêtes, le coût total de la construction de T est linéaire. La seule difficulté est de gérer efficacement les arêtes e, e_i et e_j pour préparer le basculement.

Notons que lorsque l'on traite un sommet u_i avec une arête-frontale (u_i, u_k), nous avons le choix de continuer la construction de T (ces notations se réfèrent à celles représentées dans la figure 64) : soit en continuant la construction T à partir de u_i, en traitant u_{i+1}, soit en considérant le sous-graphe S délimité par le cycle composé du chemin entre u_i et u_k dans T et fermé par l'arête (u_i, u_k), en poursuivant avec les sommets de S (en traitant les sommets u_m, u_l et u_j dans la figure 64). Si dans T_{u_i}, les sommets de S ont été visités dans l'ordre $u_{i_1}, u_{i_2}, \ldots, u_{i_r}$, alors, les sommets sont récursivement traités dans l'ordre :$u_{i_r}, u_{i_{r-1}}, \ldots, u_{i_1}$. En effet, les deux parties du graphe plan (la partie après u_i et la partie à l'intérieur de S) sont indépendantes. La

partie de l'arbre composée des sommets après u_j peut être calculée après le calcul des arbres pour S et après avoir effectué le basculement. Il n'est pas difficile de voir qu'une version récursive de l'algorithme permet de gérer les arêtes e, e_i et e_j à la volée avec un coût total de $O(\sum_{i=1}^{n} \deg(u_i)) = O(n)$. □

6.2 Super-triangulation d'un graphe planaire

6.2.1 Définition

Définition 6.2.1. *Un réaliseur $S = (T_0, T_1, T_2)$ est une super-triangulation d'un graphe G si :*

1. $V(S) = V(G)$;
2. $E(T_0) \subset E(G)$.
3. $\bar{T_0}$ *est un arbre bien-ordonné de* S.
4. *Pour chaque nœud interne de* T_2, $(v, P_1(v)) \in E(G)$.

La figure 65 montre une super-triangulation du graphe de la figure 63.

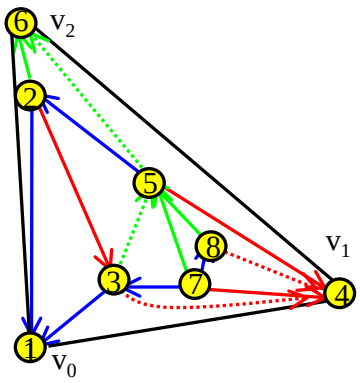

Figure 65 – Exemple de super-triangulation.

6.2.2 Construction

Lemme 6.2.1. *Soit T un arbre recouvrant bien-ordonné d'un graphe plan connexe H. Soit v_0 la racine de T. Supposons que T possède au moins 2 feuilles. Soit v_1 (resp. v_2) la première (resp. dernière) feuille de T dans l'ordre préfixe trigonométrique. Alors, il existe une unique super-triangulation $S = (T_0, T_1, T_2)$ de H, respectant le dessin de H, telle que T_i soit enraciné en v_i. De plus, $T_0 = T \setminus \{v_1, v_2\}$ et la super-triangulation S peuvent être calculés en temps linéaire.*

Démonstration. Soit T un arbre recouvrant bien-ordonné de H, enraciné en v_0 et soit $T_0 = T \setminus \{v_1, v_2\}$, où v_1 (resp. v_2) est la première (resp. dernière) feuille de T dans l'ordre préfixe trigonométrique.

Nous montrons d'abord que v_1 et v_2 appartiennent à la face extérieure de H. Considérons Q_i le chemin dans T allant de v_i à v_0 (pour $i \in \{1, 2\}$). Tous les sommets de Q_i doivent appartenir à la face extérieure de H, en particulier v_i. En fait, par induction (ceci est vrai pour v_0), un sommet v de Q_2 (resp. Q_1) possède un bloc $B_<(v)$ (resp. $B_>(v)$) vide. Donc le dernier (resp. premier) enfant de v dans l'ordre trigonométrique (s'il existe) doit appartenir à la face extérieure.

Soit H' (resp. G') le graphe plan (resp. graphe planaire) obtenu à partir de H (resp. G) en ajoutant les 3 arêtes entre les sommets v_i (ceci conserve la planarité car v_1, v_2 et v_0 appartiennent à la face extérieure) telles qu'elles forment la face extérieure de H'. Chaque arête n'est ajoutée que si elle ne crée pas d'arête-multiple. Comme les arêtes entre les sommets v_i appartiennent à n'importe quelle super-triangulation de G, il suffit de montrer que la super-triangulation $S = (T_0, T_1, T_2)$ de G' préservant H' est unique. Dans un premier temps nous allons montrer comment construire S puis dans un deuxième temps nous allons montrer que cette super-triangulation est unique.

Tout d'abord observons que $\bar{T}_0$ est enraciné en v_0 (en supprimant deux feuilles de T, $\bar{T}_0$ reste connexe). Clairement, $E(T_0) \subseteq E(G) \subseteq E(G')$, et $\bar{T}_0$ est un arbre recouvrant bien-ordonné de H' enraciné en v_0. D'après le lemme 6.1.1, $\bar{T}_0$ est unique.

Considérons l'algorithme 10 qui construit l'ensemble T_1.

Algorithme 10 Triangulation d'une paire bien-ordonnée.

```
triangulation(T_H, H)
T̄_0 ← T_H
pour chaque sommet v de G dans l'ordre préfixe trigonométrique de T_0 faire
  si B_>(v) ≠ ∅ alors
    Mettre l'arête-frontale dans T_1 (Affectation 1)
  sinon
    si v possède un frère gauche u alors
      Mettre (v, u) dans T_1 (Affectation 2)
    sinon
      Mettre (v, P_1(P_0(v))) dans T_1 (Affectation 3)
    fin si
  fin si
fin pour
```

Vérifions que $H' \cup T_1$ est toujours un graphe planaire. Remarquons tout d'abord que l'affectation 1 n'introduit pas de nouvelles arêtes. Lorsqu'on applique l'affectation 2, il n'y pas d'arête incidente à $P_0(u_i)$ entre u_i et son frère gauche, donc la planarité du graphe plan est préservée. Enfin pour l'affectation 3, $P_0(u_i)$ ne possède pas d'arête incidente entre u_i et $P_1(P_0(u_i))$, donc l'ajout de l'arête $(u_i, P_1(P_0(u_i)))$ préserve également la planarité.

Vérifions que $\{T_0, T_1\}$ sont deux arbres d'un réaliseur. Nous avons vu que $\bar{T}_0$ est bien-ordonné. Chaque sommet $u_i \neq v_0$ possède un parent dans T_1, donc T_1 est connexe. Vérifions que lors de chaque affectation, le parent de u_i est un sommet u_j avec $j > i$. Une conséquence de cela est que T_1 ne possède pas de cycle et donc T_1 est un arbre. La notation $P_1(u_i)$ a donc un sens. Vérifions aussi que l'arête $(u_i, P_1(u_i))$ est après les enfants de u_i (s'ils existent) et après l'arête $(u_i, P_0(u_i))$ dans l'ordre anti-trigonométrique. Ainsi, l'arbre T_1 est compatible avec la condition locale des réaliseurs.

A cette étape, $H' \cup T_1$ peut contenir des arêtes qui ne sont ni dans $\bar{T}_0$ ni dans $\bar{T}_1$. Soit $X = E(H') \setminus (E(\bar{T}_0) \cup E(\bar{T}_1))$ l'ensemble de ces arêtes. La construction de T_2 peut être effectuée grâce à la propriété 3.2.4. Comme il y a un unique moyen de construire l'arbre T_2 à partir de $\{T_0, T_1\}$, nous devons vérifier que les arêtes de X sont compatibles avec l'ensemble T_2 et la condition locale. Soit e une arête quelconque de X. Supposons que $e = (u_i, u_j)$ avec $i < j$. Puisque $e \notin T_0$, alors $e \in B_>(u_i)$. De plus, $e \notin T_1$ implique que e n'est pas l'arête-frontale de u_i. En fait, l'arête-frontale de u_i appartient à l'arbre T_1 (d'après l'affectation 1). Donc, $e \in T_2$ est compatible avec la condition locale.

On en déduit que $S = (T_0, T_1, T_2)$ est un réaliseur du graphe plan $H' \cup T_1 \cup T_2$. Nous avons vu que $E(T_0) \subseteq E(G) \subseteq E(G')$ et que $(\bar{T}_0, H')$ est une paire bien-ordonnée. D'après les règles d'affectation (1, 2 et 3), nous remarquons que si l'arête $(u, P_1(u)) \notin E(G')$ (affectation 2 ou 3), alors u ne peut avoir d'enfant dans T_2 (les arêtes $(u, P_1(u))$,$(u, P_0(u))$ forment un triangle avec une arête de $T_0 \cup T_1$). En d'autres termes, pour chaque nœud interne u de T_2, $(u, P_1(u)) \in E(G')$. En fait $(u, P_1(u)) \in E(G)$. Par conséquent S est une super-triangulation de G'.

Il reste à montrer que S est l'unique super-triangulation de G' qui préserve H' avec pour racine de T_i le sommet v_i. Comme T_0 est unique et comme, étant donné $\{T_0, T_1\}$ T_2 est unique, il suffit de prouver que T_1 est unique.

L'arête-frontale de u_i doit appartenir à T_1 puisque le parent de u_i dans T_1 doit être avant (dans l'ordre anti-trigonométrique) les arêtes vers les enfants de u_i dans T_2. Si $B_>(u_i) = \emptyset$, et si u_i possède un frère gauche u_j, alors (u_i, u_j) doit être dans T_1. Sinon, (u_i, u_j) devrait être dans T_2 et $u_j = P_2(u_i)$, ce qui ferait que u_i serait un nœud interne de T_2. Cependant comme $B_>(u_i) = \emptyset$, $(u_i, P_1(u_i))$ n'est pas dans $E(G')$, ce qui contredit la définition de super-triangulation. Finalement, si $B_>(u_i) = \emptyset$ et si u_i est le dernier enfant de $P_0(u_i)$, alors $(u_i, P_1(P_0(u_i)))$ doit être dans T_1. Dans le cas contraire, $(u_i, P_1(P_0(u_i)))$ serait dans T_2 et de plus $(u_i, P_1(P_0(u_i)), P_2(P_1(P_0(u_i))))$ serait un cw-triangle (ce qui contredit la définition de super-triangulation).

Donc, T_1 et S sont uniques, ce qui complète la preuve. □

Théorème 6.2.1. *Soit G un graphe planaire triconnexe. Pour tout triplet v_0, v_1, v_2, il existe une unique super-triangulation (T_0, T_1, T_2) de G telle que T_i soit enraciné en v_i pour $i \in \{0, 1, 2\}$.*

Démonstration. D'après le théorème de Whitney (théorème 1.1.1), un graphe triconnexe n'admet qu'une seule carte planaire, où trois nœuds, appelons les v_0, v_1 et v_2, sont situés sur la face extérieure. Soit H un tel graphe plan de G. Supposons que les

sommets v_0, v_1 et v_2 apparaissent dans cet ordre lorsque l'on parcourt la face extérieure dans l'ordre trigonométrique. Soit H' (resp. G') le graphe plan (resp. graphe planaire) obtenu à partir de H (resp. de G) en ajoutant les arêtes (v_0, v_1), (v_1, v_2) et (v_2, v_0) (si elles n'existent pas).

L'arbre recouvrant bien-ordonné T de H' enraciné en v_0 possède au moins deux feuilles situées sur la face extérieure de H' : v_2 et v_1, qui sont respectivement la première feuille et la dernière feuille de T dans l'ordre préfixe anti-trigonométrique. L'arbre est unique d'après le lemme 6.1.1, une fois H' et v_0 fixés. On peut alors appliquer le lemme 6.2.1, et calculer en temps linéaire l'unique super-triangulation $S' = (T'_0, T'_1, T'_2)$ de G' préservant H' et où T_i est enraciné en v_i. Comme G' est triconnexe, H' est l'unique carte de G' avec v_i ayant les sommets v_i situés sur la face extérieure. Donc S' est finalement l'unique super-triangulation de G' (lemme 6.2.1). La super-triangulation S' est aussi une super-triangulation de G car les arêtes supplémentaires de la face extérieure ne créent pas de nœud interne dans T_2. Réciproquement, puisque toute super-triangulation $S = (T_0, T_1, T_2)$ de G avec T_i enraciné en v_i doit contenir les arêtes (v_0, v_1), (v_1, v_2) et (v_2, v_0), S est aussi une super-triangulation de G'. Ceci implique donc que la super-triangulation S de G est S'. La super-triangulation S est unique car S' l'est. De plus, elle peut être calculée en temps linéaire. □

6.3 Codage d'un graphe planaire à l'aide d'une super-triangulation

6.3.1 Représentation d'un graphe planaire à l'aide d'une chaîne binaire

Dans cette section, nous considérons $S = (T_0, T_1, T_2)$ une super-triangulation d'un graphe planaire connexe G à n sommets et m arêtes.

Pour montrer comment utiliser les super-triangulations pour représenter efficacement G, nous définissons deux ensembles d'arêtes. L'ensemble des *arêtes pertinentes* :

$$R_S = \{(v, p_1))|\text{v feuille de } T_2\}$$

et l'ensemble des arêtes manquantes :

$$M_S = E(G) \setminus (E(T_0) \cup \{(v, p_1(v))|v \text{ nud interne de } T_2\}).$$

M_S est en fait, l'ensemble des arêtes de G qui ne sont ni dans T_0 ni définies par la règle 4 de la définition 6.2.1.

Théorème 6.3.1. *Soit $S = (T_0, T_1, T_2)$ une super-triangulation de G.*

1. *Etant donné T_2 et R_S, on peut déterminer S en temps linéaire.*
2. *Etant donné S et M_S, on peut déterminer G en temps linéaire.*

Démonstration. D'après la propriété branche de T_2 (proposition 1.3.3), il suffit de déterminer $P_1(v)$ pour chaque feuille de T_2, pour construire $P_1(w)$ pour chaque sommet (autre que la racine) w de T_2 (rappelons que les branches gauches partitionnent les nœuds d'un arbre). Donc, T_2 et l'ensemble des arêtes pertinentes permettent de construire l'arbre T_1 en temps linéaire. Etant donné T_2 et T_1, les arêtes de T_0 peuvent être construites de manière unique en temps linéaire.

De la définition 6.2.1, pour déterminer G à partir de S et M_S, il suffit de calculer :

1. L'ensemble des arêtes de T_0.
2. L'ensemble des arêtes $(v, p_1(v))$ telles que v soit un nœud interne de T_2.
3. L'ensemble des arêtes de manquantes.

Ceci peut être effectué en temps linéaire. □

Maintenant nous allons voir comment coder de manière efficace l'arbre T_2 ainsi que les ensembles R_S et M_S.

Soit $u_1, u_2, \ldots, u_n$ les sommets d'un arbre T dans l'ordre préfixe anti-trigonométrique de T. Une feuille u_i de T est un *bourgeon* si le parent de u_i et de u_{i+1} est u_{i-1} (voir figure 66). En d'autres termes, u_i est un bourgeon s'il est le premier enfant de $P(u_i)$ et qu'il n'est pas enfant unique. Observons que pour chaque bourgeon u_i, u_{i+1} est un nœud d'une branche gauche de T se terminant par une feuille u_j, avec $j > i$. Donc un arbre contient au moins une feuille de plus que de bourgeons.

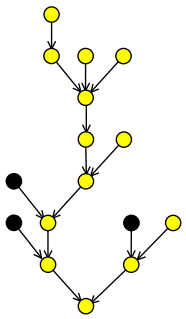

Figure 66 – Un arbre enraciné avec 8 feuilles et 3 bourgeons. La chaîne codant l'arbre (en gras le motif correspondant au bourgeon) 110**1101**1110010100100000110100.

Considérons le mot de Dyck codant l'arbre T. On peut observer qu'un bourgeon de T se traduit par un motif 1101 dans le code de T. Remarquons enfin que 2 occurrences du motif peuvent se superposer : 1101101.

Soit L une chaîne binaire. On note $\#L$ le nombre de chaînes binaires ayant la même longueur et le même nombre de "1" que L. Plus précisément, si L est de longueur x et possède y "1", alors

$$\#L = \binom{x}{y}.$$

Dans la suite de cette section, nous supposons que T_2 possède $n-2$ sommets (i.e., $\bar{T}_2$ possède n sommets), l feuilles et b bourgeons.

Propriété 6.3.1. *Soient $R = (T_0, T_1, T_2)$ un réaliseur et F un sous-ensemble des arêtes de T_2. Soient $u_1, u_2, \ldots, u_n$ les sommets de R dans l'ordre préfixe anti-trigonométrique de $\bar{T}_0$. Soient L l'ensemble des sommets u_i ayant des arêtes sortantes dans F et $D = (d_2, d_3, \ldots, d_{n-1})$ la séquence où d_i désigne le nombre d'arêtes entrantes dans F du sommet u_i. Alors, connaissant (T_0, L, D), F peut être construit en temps linéaire.*

Démonstration. On observe que u_1 et u_n n'ont pas d'arêtes entrantes dans T_2. Toutes les arêtes de F sont de la forme $(u, P_2(u))$ avec $u \in L$. Soit $u_i \in L$ où i est minimum et soit $u_j = P_2(u_i)$. Montrons que u_j a un degré entrant dans F, que u_i et u_j sont non-apparentés dans $\bar{T}_0$ et que j est le plus grand indice inférieur à i.

On observe que si une telle propriété est vérifiée, cela donne un algorithme de construction de F à partir de (T_0, L, D) (voir Algorithme 11). Montrons que cette propriété est donc vérifiée. D'après la propriété 1.2.2, u_j est non-apparenté à u_i et $j < i$. Supposons que j soit maximal. Il existe donc une arête $(u_t, u_k) \in F, u_k = P_2(u_t), u_t$ étant non-apparenté à u_i et $j < t < i$. Soit C le cycle composé du chemin entre u_j et u_i dans $\bar{T}_0$ et fermé par l'arête (u_i, u_j). Soit R la région connexe bornée de $\mathbb{R}^2 \setminus C$. Comme $j < t < i$, alors $u_t \in R \cup C$. D'après la condition locale sur le sommet u_t toutes les arêtes de T_2 appartiennent à R, en particulier (u_t, u_k). Comme $u_k \in L$ et que i est le plus petit indice tel que $u_i \in L$ alors $k > i$. On en déduit donc que $u_k \notin R \cup C$, ce qui contredit le fait que le graphe est planaire. Donc j est maximal. □

Algorithme 11 Construction de F à partir de (T_0, L, D).

$F \leftarrow \emptyset$
pour chaque sommet u_i dans l'ordre préfixe anti-trigonométrique de T_0 **faire**
  Trouver le plus grand $j < i$ tel que $d_j > 0$ et que u_j ne soit pas apparenté avec u_i
  Mettre (u_i, u_j) dans F
  d_j–
fin pour

Lemme 6.3.1. *Connaissant S et m, l'ensemble M_S peut être codé à l'aide d'une chaîne binaire C_1 telle que :*

$$\#C_1 = \binom{n+l+3}{2n-m-9}.$$

De plus, connaissant S et m, le codage et le décodage de M_S s'effectuent en temps linéaire.

Démonstration. M_S est composé d'arêtes de T_2 et d'arêtes $(v, p_1(v))$ où v est feuille de T_2. La chaîne C_1 est construite de la manière suivante.

Les $n-3$ premiers bits représentent les arêtes de T_2 qui sont dans M_S (rappelons que T_2 contient $n-2$ sommets). Plus précisément, si le i-ième bit vaut 1, alors l'arête

$(u_{i+1}, P_2(u_{i+1}))$ (où u_{i+1} désigne le $i+1$-ième sommet de T_2 dans l'ordre préfixe anti-trigonométrique) se trouve dans M_S. Si le i-ième bit vaut 0 l'arête ne s'y trouve pas.

Les l bits suivants de C_1 représentent les arêtes $(v, P_1(v))$ avec v feuille de T_2. Concrètement, en effectuant un parcours préfixe de T_2, le $(i-n-3)$-ième bit vaut 1 si et seulement si la i-ième feuille rencontrée possède une arête $(v, P_1(v))$ dans M_S. La chaîne C_1 peut être codée et décodée en temps linéaire.

La longueur de C_1 vaut $n-3+l$ et le nombre de 1 est égal à $|M_S|$. Nous avons $|M_S| = m-(n-3)-k$, où $k = n-3-l$ est le nombre de nœuds internes de T_2. □

Lemme 6.3.2. *Connaissant T_2, l'ensemble R_S peut être codé à l'aide de deux chaînes binaires B_1, B_2 et un entier $t \in [0, b]$ tels que :*

$$\#B_1 = \binom{b}{t} \text{ et } \#B_2 = \binom{n-t+l-b-3}{l-b-1}$$

De plus, connaissant T_2, le codage et le décodage de R_S s'effectuent en temps linéaire.

Démonstration. Nous considérons les bourgeons de T_2 de la manière suivante. Soient $u_1, u_2, \ldots, u_n$ les sommets de $\bar{T}_2$ dans l'ordre préfixe anti-trigonométrique de $\bar{T}_2$. Pour chaque bourgeon u_i, notons $l(u_i)$ la feuille de la branche gauche contenant u_{i+1} (sur la figure 66 $l(u_i) = u_j$). Soit $B = \{l(u_i) | u_i \text{ est un bourgeon de } T_2\}$. Puisque $\bar{T}_2$ vérifie la propriété branche, $P_1(l(u_i))$ est égal à $P_1(u_{i-1})$ ou $P_1(l(u_i))$ est un descendant de u_{i-1} avant u_{i+1} dans l'ordre anti-trigonométrique. Ce descendant est unique et est le bourgeon u_i. Donc, pour chaque bourgeon u_i, soit $P_1(l(u_i)) = P_1(u_{i-1})$ soit $P_1(l(u_i)) = u_i$.

Soit $A = \{(v, P_1(v)) | v \in B\} \subset R_S$. Nous représentons différemment les arêtes de A et celles de $R_S \setminus A$. Soit t le nombre de bourgeons de u_i tel que $P_1(l(u_i)) = P_1(u_{i-1})$. La chaîne binaire B_l est de longueur b et est définie de la manière suivante : le j-ième bit de B_l vaut 1 si et seulement si pour le j-ième bourgeon u_i de T_2, $P_1(l(u_i)) = P_1(u_{i-1})$. Clairement, $\#B_l = \binom{b}{t}$. Comme il n'y a que deux cas, pour $P_1(l(u_i))$, on peut reconstruire complètement R_S connaissant $T_2, R_S \setminus A$, et B_l.

Pour représenter $R_S \setminus A$, nous considérons la propriété 6.3.1 appliquée au réaliseur (T_2, T_0, T_1) et pour $F = R_S \setminus A$. L'ensemble F peut être déterminé en temps linéaire à partir du triplet (T_2, L, D) où L désigne l'ensemble des feuilles de T_2 qui ne sont pas dans B et $D = d_2, d_3, \ldots, d_{n-1}$ est la séquence des degrés entrants des arêtes de F. Comme T_2 est connu L peut être calculé et seule D a besoin d'être représentée pour reconstruire $F = R_S \setminus A$. Remarquons que pour chaque bourgeon u_i tel que $P_1(l(u_i)) = P_1(u_{i-1})$, u_i ne possède pas d'arête entrante dans F, i.e. $d_i = 0$. Connaissant T_2 et B_l, un tel cas peut être détecté. Donc, on peut supprimer de D tous les d_i correspondant à ce cas. La sous-séquence résultante est composée de $n-2-t$ entiers (éventuellement nuls) dont la somme vaut : $|F| = |R_S \setminus A| = |R_S| - |B| = l - b$.

Toute séquence de $k \geq 1$ entiers (éventuellement nuls) dont la somme vaut $s \geq 1$ peut être représentée (en temps linéaire) par une chaîne binaire de longueur $k-1+s$

possédant $k-1$ bits à "1" : chaque entier i est codé par une séquence de i "0" délimitée par un "1".

Par exemple, la séquence $2,1,0,1,1,0,1,3$ de 8 entiers dont la somme fait 9 peut être codé par la chaîne binaire 00101110101101000.

De manière similaire, la séquence D' est codée par une chaîne binaire B_2 telle que $\#B_2 = \binom{n-t+l-b-3}{l-b-1}$. Donc, connaissant T_2, R_S peut être représenté par les chaînes B_1 et B_2. □

Lemme 6.3.3. *Connaissant n, l, b, l'arbre T_2 peut être codé à l'aide de cinq chaînes binaires $A_1, \ldots, A_5$, et par des entiers $p \in [1, l]$ et $w \in [b, n-l]$, tels que :*

$$A_1 = \binom{l-1}{p-1}, \#A_2 = \binom{p}{b}, \#A_3 = \binom{w}{b-1},$$

$$\#A_4 = \binom{n-l-w+p-b-3}{p-b-1}, \#A_5 = \binom{n-l-2}{p-b-1}.$$

De plus, connaissant n, l, b le codage et le décodage de T_2 s'effectuent en temps linéaire.

Démonstration. Soit s le mot de Dyck codant T_2 où 4 bits ont été ajoutés : une séquence 01 au début et à la fin de s. La longueur de s est donc $2n-2$ (rappelons que T_2 possède $n-2$ sommets. La chaîne s contient :

- $n-1$ "0".
- $n-1$ "1".
- l séquences 10 (le mot de Dyck codant T_2 commence par un "1" et se termine par un "0" et à chaque feuille correspond une séquence 10).
- $l+1$ séquences 01 (le mot de Dyck codant T_2 contient exactement $l-1$ séquences 01).

Le nombre de séquences 1101 dans s (se chevauchant potentiellement) vaut b, le nombre de bourgeons.

Soit X_p l'ensemble des chaînes binaires de longueur $2n-2$, et de la forme $(A'_i B_i)^p A'_{p+1}$ où les A'_i sont les blocs maximaux de séquences 01, et $A'_1 = A'_{p+1} = 01$. Par exemple :

$$W = \boxed{01} B_1 \boxed{010101} B_2 \boxed{01\ldots} \ldots B_p \boxed{01}, W \in X_p.$$

Notons que $S \in X_p$ pour un certain paramètre $p \geq 1$. Les chaînes $A_1, \ldots, A_5$ sont utilisées pour représenter les membres de X_p, et donc en particulier s.

Soit $a'_i \geq 1$ l'entier tel que $A'_i = (01)^{a'_i}$, pour chaque $i \in \{1, 2, \ldots, p+1\}$. La séquence d'entiers non nuls $a'_2, a'_3, \ldots, a'_p$ (rappelons que $a'_1 = a'_p = 1$) peut être utilisée pour représenter tous les blocs A'_i. Cette séquence peut être décrite par une chaîne binaire A_1 telle que $\#A_1 = \binom{s}{p-1}$ où $s = \sum_{i=2}^{p} a'_i$. Comme toutes les séquences de 01

de s sont localisées dans les A'_i, $s = (l+1) - 2$. Donc $\#A_1 = \binom{l-1}{p-1}$. Il reste maintenant à décrire les blocs B_i.

Pour chaque bloc B_i deux cas seulement peuvent se produire. Soit B_i contient uniquement des "1" : $B_i = 1^+$, soit B_i se termine par au moins un "0" : $B_i = 1^*0^+$. Dans le premier cas ($B_i = 1^+$), $A'_iB_iA'_{i+1}$ contient exactement un bourgeon (une séquence 1101) , et exactement une séquence 10 (entre les blocs B_i et A'_{i+1}). Dans le deuxième cas ($B_i = 1^*0^+$), $A'_iB_iA'_{i+1}$ contient exactement une séquence 10.

Une chaîne binaire A_2 indique pour chaque bloc B_i, s'il se trouve dans le premier cas ou le deuxième cas. Le premier cas apparaît b fois, donc $\#A_2 = \binom{p}{b}$. Soit w le nombre de "1" utilisés dans les blocs B_i qui sont dans le premier cas ($B_i = 1^+$). Les longueurs des blocs B_i du premier cas peuvent être décrites à l'aide d'une séquence de b entiers non nuls dont la somme vaut précisément w. Donc, une chaîne binaire A_3 telle que $\#A_3 = \binom{w}{b-1}$ permet de représenter cette séquence.

Pour compléter la description des membres de X_p, il reste à définir les blocs B_i qui se trouvent dans le deuxième cas. Il y a $p - b$ blocs dans le deuxième cas et ils sont de la forme 1^*0^+. Il suffit de décrire le nombre de "0" et de "1" de chaque bloc B_i du deuxième cas. Le nombre de "1" restants dans ces blocs est $(n-1) - (l+1) - w$. Comme le cas 2 apparaît $p - b$ fois, une séquence contenant la longueur des séquences de "1" (éventuellement égale à "0") peut-être représentée par une chaîne binaire A_4 telle que $\#A_4 = \binom{(n-1)-(l+1)-w+p-b-1}{p-b-1}$. Les longueurs des séquences de "0" peuvent être représentées par une chaîne binaire A_5 telle que $\#A_5 = \binom{n-l-2}{p-b-1}$, car le nombre de "0" des blocs B_i dans le cas 2 est $(n-1) - (l+1)$ (il y a $l+1$ "0" dans les blocs A'_i).

Ces chaînes binaires peuvent être clairement construites en temps linéaire. □

Lemme 6.3.4. *Chaque chaîne binaire S de longueur n peut être codée par chaîne S' de longueur $\log_2(\#S) + O(n\log(\log(n))/\log(n))$. De plus, connaissant n, le codage de S en S' et le décodage de S' peuvent être effectués en temps et en espace linéaire à l'aide d'un ordinateur RAM sur des mots de $w \geq \log_2(n)$ bits.*

Démonstration. L'idée principale est de partager S en blocs de taille identique b et de coder chaque bloc de manière optimale. Le codage de chaque bloc prend un temps exponentiel en b. Cependant, le codage de tous les blocs possibles peut être placé dans un tableau une fois pour toutes en $O(2^{O(b)}) = O(n)$, pour un b suffisamment petit. L'optimalité du codage de S' dérive de l'optimalité du codage de chaque bloc par super-additivité des binomiaux. Plus précisément, le codage s'effectue de la manière suivante.

Soit $b = \lfloor \log_2(n) - \log_2(\log_2(n)) \rfloor$. Donc, les opérations classiques sur les entiers inférieurs à 2^b peuvent être effectuées en temps constant puisque $w \geq b$. Par la suite k_p dénotera $\binom{b}{p}$. Nous devons construire quelques tableaux.

Nous construisons tout d'abord un tableau L tel que pour chaque $p \in [0, b]$, $L[p] = \lceil \log_2(k_p) \rceil$. Tous les nombres $k_0, k_1, \ldots, k_b$ peuvent être calculés dynamiquement à l'aide $O(b^2)$ nombres codés sur b bits (Méthode de Pascal). Au total, la construction de L coûte en temps de $O(b^2 + \sum_p \log(\log(k_p))) = O(\log^2(n))$, où $O(\log(\log(k_p)))$ est le

coût du calcul de$\lceil \log(k_p) \rceil$ à partir de la représentation binaire de k_p (en utilisant une recherche binaire et des masques).

Nous construisons un tableau P d'entiers plus petits que b tel que pour chaque $i \in [0..2^b[$, $P[i]$ est le nombre de "1" dans la représentation binaire de i. Le tableau P peut être construit en temps et en espace $O(b2^b) = O(n)$. Toutefois, le temps peut être réduit à $O(b2^{b/2}) = O(\sqrt{n \log(n)})$ (ou même plus petit encore) en utilisant un tableau P' pour des demi-mots de taille $\lceil b/2 \rceil$ bits. En effet, nous avons $P[i] = P'[i/2^{\lfloor b/2 \rfloor}] + P'[i \mod 2^{\lfloor b/2 \rfloor}]$.

Pour chaque $p \in \{0, 1, \ldots, b\}$, nous calculons le tableau D_p (utilisé pour le décodage) tel que, pour chaque $i \in [0, k_p[$, $D_p[i]$ est une chaîne binaire distincte de longueur b possédant p "1". Les chaînes de D_p sont ordonnées dans l'ordre lexicographique. La génération de toutes les chaînes de D_p coûte en temps $O(2^b) = O(n/\log(n))$ et en espace $O(b2^b) = O(n)$ en parcourant toutes les chaînes binaires $s \in [0, 2^b[$ en incrémentant la valeur et en remplissant la case correspondante de $D_{P[s]}[i_p]$ (et en mettant à jour l'indice i_p).

Finalement, nous construisons un tableau C (utilisé pour le codage) tel que pour chaque $s \in [0, 2^b[$, $C[s]$ contient l'indice i tel que $D_P[i] = s$, où $p = P[s]$. L'indice $i = C[s]$ est stocké sur b bits, bien que seulement les $L[p] = \lfloor log_2(k_p) \rfloor$ bits les moins significatifs de i sont utiles puisque $i \in [0, k_p[$. Pour construire C, il suffit d'itérer pour chaque $p \in [0, b]$ et pour chaque $i \in [0, k_p[: C[D_p[i]] = i$. Une fois D_p et P calculés, la construction de C coûte $O(\sum_{p=0}^{b} k_p) = O(2^b) = O(n/\log(n))$ en temps et $O(b2^b) = O(n)$ en espace.

Soit $t = \lfloor n/b \rfloor$ le nombre de blocs de b bits de S. Si n n'est pas divisible par b, les derniers $(n \mod b)$ bits sont traités séparément. Pour les procédures de codage et de décodage, nous itérons t fois les étapes suivantes :

1. Lire le bloc suivant s de b bits de S (s peut être manipulé comme un index de $[0, 2^b[$).
2. Ecrire dans S' la valeur de $P[s]$ comme un nombre binaire sur $\lceil \log_2(b) \rceil$ bits.
3. Ecrire dans S' la chaîne composée des $L[s]$ bits les plus significatifs de $C[s]$.

Nous finissons le processus de codage en écrivant dans S' les $n(\mod b)$ bits de S (s'ils existent).

La procédure de décodage est la suivante :

1. Lire dans S' les $\lceil \log_2(b) \rceil$ bits pour former la valeur p.
2. Lire dans S' les $L[p]$ bits suivants, représentant un entier $i \in [0, k_p[$.
3. Ecrire dans S la chaîne $D_p[i]$.

Nous finissons le processus de décodage en écrivant dans S les $(n \mod b)$ derniers bits de S' (s'ils existent).

Les procédures de codage et de décodage prennent clairement un temps $O(t) = O(n/\log(n))$, une fois que les tableaux L, P, D_p et C ont été générés. La validité du codage et du décodage est assurée par la symétrie des procédures ci-dessus.

Il reste à montrer que la longueur de la chaîne S' n'excède pas $\log_2(\#S) + O(n\log(\log(n))/\log(n))$. Soit p_i le nombre de "1" dans le i-ième bloc de b bits de S, pour tout $i \in \{1, 2, \ldots, p\}$. D'après la procédure de codage, le nombre de bits écrits dans S' pour le i-ième bloc est : $\lceil \log_2(b) \rceil + \lceil \log_2(k_{p_i}) \rceil$. En sommant sur tous les blocs, nous obtenons la borne supérieure suivante sur la longueur de S' :

$$\sum_{i=1}^{t} (\lceil \log_2 b \rceil + \lceil \log_2(k_{p_i}) \rceil) + (n \bmod b) \;=\; \left(\sum_{i=1}^{t} \log_2(k_{p_i}) \right) + O(b + t \log_2 b) \;.$$

Observons que par super-additivité $\binom{a}{b}.\binom{a'}{b'} \leq \binom{a+a'}{b+b'}$ donc

$$\prod_{i=1}^{t} k_{p_i} \;=\; \prod_{i=1}^{t} \binom{b}{p_i} \;\leq\; \binom{bt}{\sum_i p_i} \;=\; \#\overline{S},$$

où $\bar{S}$ est la chaîne composée des tb premiers bits de S. Comme la longueur et le nombre de "1" de S et de $\bar{S}$ ne diffèrent que d'au plus b, il s'en suit que $|\log_2(\#S) - \log_2(\#\bar{S})| = O(b\log_2(n))$. Par conséquent, nous obtenons que la longueur de S' est au plus de :

$$\left(\log_2 \prod_{i=1}^{t} k_{p_i} \right) + O(b + t\log b) \;\leq\; \log_2(\#S) + O(n\log_2(\log_2(n))/\log_2(n)).$$

□

6.4 Nombre de graphes planaires non étiquetés

Cette section est dédiée aux résultats suivants :

Théorème 6.4.1. *Tout graphe plan maximal et tout graphe planaire connexe à n sommets peuvent être représentés par une chaîne binaire de longueur au plus $3,37n$ bits et $5,03n$ bits respectivement. De plus, le codage et le décodage peuvent être effectués en temps linéaire.*

Théorème 6.4.2. *Le nombre $p(n)$ de graphes planaires non étiquetés à n sommets, satisfait, pour n suffisamment grand la double inégalité suivante :*

$$\beta n - \Theta(\log n) \;\leq\; \log_2 p(n) \;\leq\; \alpha n + O(\log n)$$

avec $\alpha \approx 5,007$ et $\beta \approx 4,710$.

La borne inférieure du théorème 6.4.2 vient du nombre $g(n)$ de graphes planaires 2-connexes étiquetés. Clairement, $p(n) \geq (g(n)/n!)$. L'asymptotique suivante a été prouvée dans [8] :

$$g(n) \;\sim\; \Theta(n^{-7/2})\, c^{-n}\, n!$$

où $c \approx 0,03819$. Il en suit que pour n suffisamment grand,

$$\log_2 p(n) \;\geq\; \log_2 \frac{g(n)}{n!} \;=\; \beta n - \frac{7}{2}\log_2 n + O(1), \quad \text{with } \beta = -\log_2 c \approx 4,71066.$$

Majorons maintenant le nombre de graphes planaires. Soit $q(n)$ le nombre de graphes planaires connexes non étiquetés à n sommets. Pour relier $p(n)$ à $q(n)$, nous représentons chaque graphe planaire G à $k \geq 1$ composantes connexes par le triplet $(k, t(G), v)$, où $t(G)$ et v sont définis de la manière suivante. Soit u_i un sommet (qui ne soit pas un sommet d'articulation) de la i-ième composante connexe de G. Il est clair qu'un tel sommet peut toujours être trouvé en prenant, par exemple, une feuille d'un arbre recouvrant de la composante connexe. Le graphe $t(G)$ est obtenu en fusionnant toutes les composantes connexes de G : on identifie tous les sommets u_i en un seul sommet v. Clairement, $t(G)$ est un graphe planaire connexe. On peut obtenir le graphe G à partir de $(k, t(G), v)$ en divisant le sommet v de $t(G)$. Toutes les $k' \leq k$ composantes connexes obtenues de cette manière sont comprises dans G (on ne risque pas de déconnecter une composante connexe de G, car les sommets u_i ne sont pas des sommets d'articulation). Pour reconstruire complètement G, on peut être amené à ajouter $k' - k$ sommets isolés. Le nombre de sommets de $t(G)$ est $n - (k' - k) - k' + 1 = n - k + 1$.

La figure 67, montre un graphe non connexe G et sa représentation par le triplet $(k, t(G), v)$.

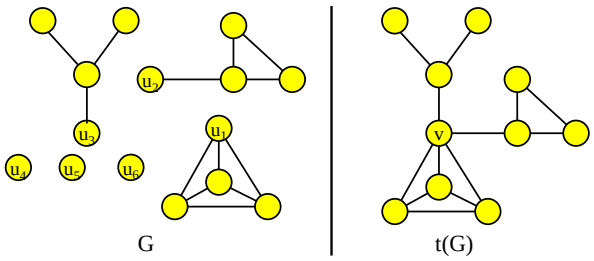

Figure 67 – Représentation d'un graphe planaire non connexe par le triplet $(k, t(G), v)$.

De cette représentation, on déduit l'inégalité suivante :

$$p(n) \;\leq\; \sum_{k=1}^{n} k \cdot q(n-k+1) \cdot (n-k+1) \;\leq\; n^3 q(n) \tag{4}$$

car $q(n)$ est une fonction croissante de n. Donc, pour prouver le théorème 6.4.2, il reste à prouver que pour n suffisamment grand,

$$\log_2 q(n) \;\leq\; \alpha n + O(\log n), \qquad \text{pour une constante } \alpha \approx 5,007. \tag{5}$$

6.4.1 Fonction entropie

Considérons la majoration suivante (voir par exemple [13, P. 255]) :

$$\binom{x}{y} < \left(\frac{x}{y}\right)^y \left(\frac{x}{x-y}\right)^{x-y} = \left(\frac{x}{rx}\right)^{rx} \left(\frac{x}{x-rx}\right)^{x-rx} = \left(\left(\frac{1}{r}\right)^r \left(\frac{1}{1-r}\right)^{1-r}\right)^x = \left(2^{\mathrm{H}(r)}\right)^x$$

où $r = \frac{y}{x}$ et H est la *fonction entropie* de r, définie pour $r \in [0,1]$ par :

$$\mathrm{H}(r) := -r\log_2 r - (1-r)\log_2(1-r), \quad \text{et avec} \quad \mathrm{H}(0) := \mathrm{H}(1) := 0 \ .$$

Donc pour tous $n, c_1, c_2 > 0$:

$$\frac{1}{n}\log_2 \binom{c_1 n}{c_2 n} \ \leq \ c_1 \mathrm{H}(c_2/c_1) \ . \tag{6}$$

Du théorème 6.3.1, il ressort que chaque graphe planaire connexe G à n sommets et m arêtes peut être représenté par huit chaînes binaires $A_1, \ldots, A_5, B_1, B_2, C_1$ décrites dans les lemmes 6.3.3, 6.3.2 et 6.3.1 ainsi qu'un nombre constant d'entiers sur $O(\log(n))$ bits. Soit Q défini de la manière suivante :

$$Q \ := \ \max_{\ell,b,t,w,p} \log_2 \left(\#A_1 \cdots \#A_5 \cdot \#B_1 \cdot \#B_2 \cdot \#C_1\right)$$

où ℓ, b, t, w, p sont des entiers de l'intervalle $[0,n]$ définis dans les lemmes 6.3.3, 6.3.2 et 6.3.1, et $q(n,m)$ est le nombre de graphes planaires connexes à n sommets et m arêtes. De cette représentation par huit chaînes binaires, on en déduit

$$\log_2 q(n,m) \ \leq \ Q + O(\log_2 n) \ . \tag{7}$$

Comme $q(n) \leq \sum_{m=n-1}^{3n-6} q(n,m) \leq 2n \max_m q(n,m)$, pour prouver l'équation (5) et calculer α, il suffit d'analyser la valeur de Q. Soient $\lambda = \frac{l}{n}$ et

$$f(\lambda) := \max_{b,t,w,p} \left\{ \frac{1}{n}\log_2 \left(\#A_1 \cdots \#A_5 \cdot \#B_1 \cdot \#B_2\right) \right\} \ .$$

Plus simplement, $f(\lambda)n$ représente le nombre de bits utilisés pour coder la paire (T_2, R_S) et donc par le théorème 6.3.1, le nombre de bits qui codent la super-triangulation S. Notons que seulement n, m et l (rappelons que $l = \lambda n$) apparaissent dans l'expression de $\#C_1$. Donc, par définition de Q, nous avons :

$$Q \ = \ \max_{0\leq\lambda\leq 1} \left\{ f(\lambda)n \ + \max_{n<m\leq 3n-6} \log_2(\#C_1) \right\} + O(\log_2 n) \ . \tag{8}$$

Soit $\mu = \frac{m}{n}$. Remarquons que $\#C_1 = \binom{n+l}{3n-6-m} \leq (n+l)^6 \binom{n+l}{3n-m} \leq (2n)^6 \binom{(1+\lambda)n}{(3-\mu)n}$. De l'équation (6), la borne supérieure sur $\#C_1$ avec la fonction entropie donne :

$$\log_2(\#C_1) \ \leq \ \left((1+\lambda)\cdot \mathrm{H}\left(\frac{3-\mu}{1+\lambda}\right)\right) n + 6\log_2(2n) \ .$$

Ainsi, en utilisant cette majoration de $\#C_1$ dans l'équation (8), nous obtenons :

$$\frac{1}{n}\left(Q - O(\log n)\right) \;\leq\; \max_{\substack{0\leq\lambda\leq 1\\ 1\leq\mu\leq 3}} \left\{ f(\lambda) + (1+\lambda)\cdot \mathrm{H}\left(\frac{3-\mu}{1+\lambda}\right)\right\} \;. \tag{9}$$

L'étude de la fonction f peut être effectuée en exprimant chaque binomial à l'aide de la fonction entropie. Dériver la fonction et résoudre les équations est un calcul lourd. L'expression clause de $f(\lambda)$ est assez longue. A l'aide d'un solveur formel (voir en annexe la feuille de calcul complète [1]), nous obtenons le comportement de f (voir figure 68).

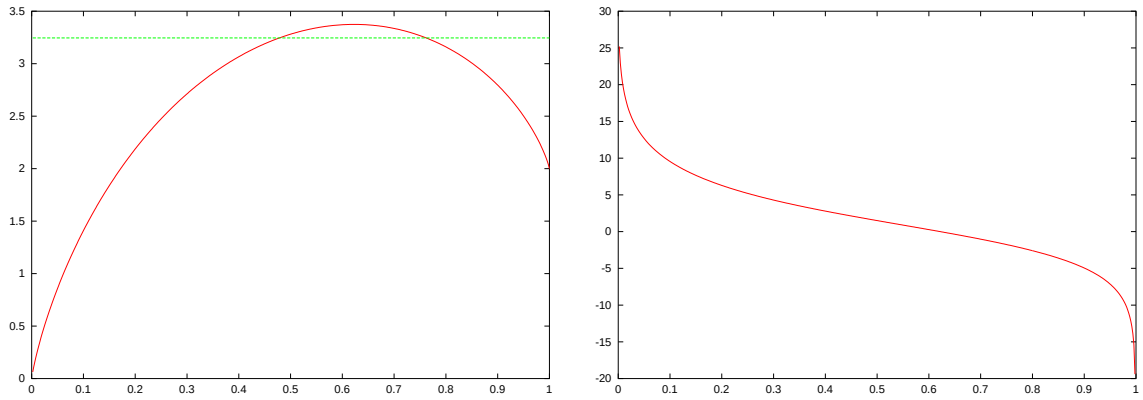

Figure 68 – Comportement de $f(\lambda)$ et $f'(\lambda)$.

Le maximum de $f(\lambda)$ et atteint pour $\lambda = \lambda_0$, solution de l'équation de degré 5 suivante :

$$21\lambda^5 - 49\lambda^4 + 47\lambda^3 - 24\lambda^2 + 7\lambda - 1 = 0 \;.$$

Dans l'intervalle $[0,1]$, nous obtenons une unique solution $\lambda_0 \approx 0,622900$ et $f(\lambda_0) \approx 3,374449$.

Démonstration *du théorème 6.4.1.* D'après le lemme 6.3.4, chacune des chaînes apparaissant dans Q peut être compressée de manière asymptotiquement optimale. Le graphe G peut alors être codé en temps linéaire avec au plus $Q + O(\frac{n\log(\log(n))}{\log(n)})$ bits, car la somme des longueurs des chaînes codant G est en $O(n)$.

Si G est une triangulation, alors $m = 3n-6$. Donc $\mu = 3$ et l'équation (9) se réécrit en :

$$\frac{1}{n}\left(Q - O(\log n)\right) \;\leq\; \max_{0\leq\lambda\leq 1} f(\lambda) \;=\; f(\lambda_0) \;\approx\; 3,37$$

ce qui prouve que les triangulations peuvent être représentées en temps linéaire avec asymptotiquement $3,37n$ bits.

[1] Cette feuille est également disponible à l'adresse suivante :
http : //www.labri.fr/ gavoille/article/BGH02.mws

Le maximum de la borne supérieure sur Q dans l'équation (9) est atteint pour $\mu = \frac{1}{2}(5-\lambda)$, puisque $H(r)$ est maximum pour $r = \frac{1}{2}$ (et nous avons $H(\frac{1}{2}) = 1$). Donc les graphes planaires connexes peuvent être représentés en temps linéaire asymptotiquement avec Q bits tels que :

$$\frac{1}{n}\left(Q - O(\log n)\right) \;\leq\; \max_{0\leq\lambda\leq 1}\left\{f(\lambda) + 1 + \lambda\right\}.$$

Les calculs nous ramènent à la résolution de l'équation de degré 7 suivante :

$$275\lambda^7 - 1430\lambda^6 + 2884\lambda^5 - 2986\lambda^4 + 1767\lambda^3 - 614\lambda^2 + 118\lambda - 10 = 0\ .$$

Dans l'intervalle $[0,1]$, nous obtenons une unique solution $\lambda_1 \approx 0,699017$. Nous avons donc $f(\lambda_1)+1+\lambda_1 \approx 5,036013$. Donc un graphe planaire connexe peut être représenté en temps linéaire à l'aide de $5,03n$ bits. Notons que les graphes qui atteignent un codage en $5,03n$ bits possèdent asymptotiquement $\frac{1}{2}(5-\lambda_1)n \approx 2,15n$ arêtes. □

Démonstration *du théorème 6.4.2.* D'après le théorème 6.4.1 , nous avons déjà $5,03$ comme borne supérieure de α. Pour améliorer un peu cette borne, nous observons qu'une triangulation possède une seule super-triangulation (voir théorème 6.2.1). Donc nous pouvons alternativement représenter la super-triangulation soit en utilisant notre représentation (donnée dans le théorème 6.3.1) à l'aide de (T_2, R_S) (ceci pouvant être codé en $f(\lambda)n+O(\log(n))$ bits), soit nous pouvons utiliser un codage théorique optimal en nombre de bits de la triangulation sous-jacente de S en $\lceil \log_2(T_n)\rceil$ bits où T_n désigne le nombre de triangulations enracinées. Le théorème 6.2.1 nous assure que l'on peut reconstruire de manière unique S à partir d'une triangulation.

D'après la formule de Tutte (voir chapitre 1 section 1.1), pour tout $n \geq 3$,

$$T_n \;=\; \frac{2(4n-11)!}{(n-2)!(3n-7)!} \;=\; \frac{2}{(3n-8)(3n-7)}\binom{4n-11}{n-2} \;\leq\; \binom{4n}{n}\ .$$

Donc,

$$\frac{1}{n}\log_2(T_n) \;\leq\; 4\mathrm{H}(1/4) \;=\; 8 - 3\log_2 3 \;\approx\; 3,245112$$

A partir des deux représentations d'une triangulation, l'équation (9) se réécrit en

$$\frac{1}{n}\left(Q - O(\log n)\right) \;\leq\; \max_{\substack{0\leq\lambda\leq 1\\ 1\leq\mu\leq 3}}\left\{\min\left\{f(\lambda), 4\mathrm{H}(1/4)\right\} + (1+\lambda)\cdot \mathrm{H}\left(\frac{3-\mu}{1+\lambda}\right)\right\}. \qquad (10)$$

Comme $H(\frac{3-\mu}{1+\lambda}) \leq 1$ et que $\log_2(q(n,m)) \leq Q + O(\log(n))$, l'équation (10) s'écrit

$$\frac{1}{n}\left(\log_2 q(n) - O(\log n)\right) \;\leq\; \max_{0\leq\lambda\leq 1}\left\{\min\left\{f(\lambda), 4\mathrm{H}(1/4)\right\} + 1 + \lambda\right\}\ .$$

Dans l'intervalle $[0,1]$, l'équation $f(\lambda) = 4H(\frac{1}{4})$ possède deux solutions (voir figure 68). Numériquement, ces solutions sont $\lambda_2 \approx 0,478207$ et $\lambda_3 \approx 0,762116$. Les valeurs λ_0 et

λ_1 maximisent respectivement $f(\lambda)$ et $f(\lambda)+1+\lambda$ donc $f'(\lambda_0)=0$ et $f'(\lambda_1)=-1$. Les fonctions f et f' décroissent pour $\lambda \in [\lambda_0, 1]$ (voir figure 68). Donc, $f'(\lambda_3) \leq -1$ puisque $\lambda_3 > \lambda_1$. Ceci signifie que pour $\lambda \in [\lambda_3, 1]$, le terme $f(\lambda)$ décroît plus que le terme $1+\lambda$ ne croît. En d'autres termes :

$$\forall \lambda > \lambda_3, \quad f(\lambda)+1+\lambda \ < \ f(\lambda_3)+1+\lambda_3 \ = \ 4\mathrm{H}(1/4)+1+\lambda_3 \ .$$

Donc,

$$\max_{0\leq\lambda\leq 1} \{\min\{f(\lambda), 4\mathrm{H}(1/4)\}+1+\lambda\} \ = \ 4\mathrm{H}(1/4)+1+\lambda_3 \approx 5,007228 \ .$$

En fixant $\alpha = 4H(\frac{1}{4})+1+\lambda_3$, nous avons donc prouvé que $\log_2(q(n)) \leq \alpha n + O(\log(n))$, et aussi que $\log_2(q(n)) \leq \alpha n + O(\log(n))$ avec $\alpha \approx 5,007$. Ceci complétant la preuve du théorème 6.4.2. □

6.4.2 Nombre d'arêtes d'un graphe planaire aléatoire

Théorème 6.4.3. *Presque tous les graphes non étiquetés et presque tous les graphes étiquetés de n sommets ont au moins $1,70n$ arêtes et au plus $2,54n$ arêtes. Ce résultat est également valable pour les graphes connexes étiquetés ou non étiquetés.*

Démonstration. Soient $m_1 = 1,70n$, $m_2 = 2,54n$ et $I = [m_1, m_2]$. Soit $l(n,m)$ (resp. $p(n,m)$) le nombre de graphes étiquetés (resp. non étiquetés) à n sommets et m arêtes. On notera $l(n)$ le nombre de graphes planaires étiquetés à n sommets.

Nous concentrons notre attention dans un premier temps sur la première affirmation du théorème 6.4.3. Nous avons à prouver que :

$$\lim_{n\to+\infty} \sum_{m\in I} \frac{p(n,m)}{p(n)} \ = \ 1 \quad \text{et} \quad \lim_{n\to+\infty} \sum_{m\in I} \frac{l(n,m)}{l(n)} \ = \ 1 \ .$$

Comme $p(n) = \sum m \in Ip(n,m) + \sum m \notin Ip(n,m)$, et $l(n) = \sum m \in Il(n,m) + \sum m \notin Il(n,m)$, nous devons prouver que :

$$\sum_{m\notin I} p(n,m) \ = \ o(p(n)) \quad \text{et} \quad \sum_{m\notin I} l(n,m) \ = \ o(l(n)) \ . \tag{11}$$

Pour prouver l'équation (11), nous relions $p(n,m)$ et $l(n,m)$ à $q(n,m)$, nombres introduits dans l'équation (7).

Comme nous l'avons vu précédemment, un graphe planaire non étiqueté G peut être représenté par le triplet $(k, t(G), v)$ où $t(G)$ est un graphe connexe qui possède le même nombre d'arête que G.c Comme le nombre d'arêtes d'une composante connexe de G avec η sommets dans l'intervalle $[\eta-1, 3\eta-m]$, le nombre k de composantes

connexes de G est au moins $k_1 = \max\{1, n-m\}$, et au plus $k_2 = n - \lceil \frac{m}{3} \rceil - 1$ si $m \geq 2$ et $k_2 = n - \lceil \frac{m}{3} \rceil$ sinon. Ainsi l'équation (4) s'étend de la manière suivante :

$$\forall n, m \geq 0, p(n,m) \;\leq\; \sum_{k=k_1}^{k_2} k \cdot q(n-k+1, m) \cdot (n-k+1) \;\leq\; n^2 \sum_{k=k_1}^{k_2} q(n-k+1, m) \; .$$

Pour les graphes planaires étiquetés, nous utilisons la représentation de G par un quadruplet $(k, t(G), v, L)$, où L est un tableau de n entiers compris entre 1 et n. Chaque élément du tableau contient l'étiquette de chaque sommet de G. La construction est similaire au cas non étiqueté. Toutes les k composantes connexes de G sont fusionnées à l'aide d'un sommet v (le sommet fusionné de chaque composante connexe est choisi pour ne pas être un sommet d'articulation). Le graphe $t(G)$ est un graphe planaire connexe non étiqueté, et possède $n-k+1$ sommets et m arêtes. Soient $u_1, u_2, \ldots, u_{n-k+1}$ les sommets de $t(G)$ ordonnés de manière arbitraire. Soient $C_1, C_2, \ldots, C_{k'}$ les composantes connexes de G qui ne sont pas des sommets isolés. Nous devons ordonner ces composantes connexes sur la base de $t(G)$, sans information sur les étiquettes de G. Les sous-graphes C_i sont ordonnés tels que $C_i \setminus \{v\}$ contienne le sommet u_z ayant le plus petit indice z parmi les sommets de $\{u_1, u_2, \ldots, u_{n-l+1}\} \setminus \cup_{j=1}^{i-1} C_{j-1}$. Par convention, $C_0 = \{v\}$. Par exemple, $C_1 \setminus \{v\}$ doit contenir u_1 ou u_2 si $v \neq u_1$. Pour chaque $i \in \{1, 2, \ldots, n-k+1\}$, $L[i]$ contient l'étiquette de u_i dans G, avec la convention que l'étiquette de $v = u_{i_0}$ est affectée comme si v appartient à C_1 dans G. Alors, pour chaque $i \in \{2, 3, \ldots, k'\}$, $L[n-k+i]$ est l'étiquette de v car v appartient à C_i dans G. Pour finir les éléments $L[n-k+k'+1], \ldots, L[n]$ sont les étiquettes des sommets isolés de G. La reconstruction du graphe sous-jacent non étiqueté de G peut être effectuée comme précédemment à l'aide du triplet $(k, t(G), v)$ (voir figure 67). Alors, il n'est pas difficile de voir que L permet d'assigner correctement les étiquettes de G. A partir de cette représentation, nous avons alors les inégalités suivantes :

$$l(n,m) \;\leq\; \sum_{k=k_1}^{k_2} k \cdot q(n-k+1, m) \cdot (n-k+1) \cdot n! \;\leq\; n^2 \cdot n! \cdot \sum_{k=k_1}^{k_2} q(n-k+1, m) \; .$$

Maintenant, établissons une borne supérieure pour $q(n,m)$, pour chaque $m \in [n-1, m_1[\cup]m_2, 3n-6]$. Par commodité, nous définissons $h(\mu)$:

$$h(\mu) \;:=\; \max_{0 \leq \lambda \leq 1} \left\{ \min\{f(\lambda), 4\mathrm{H}(1/4)\} + (1+\lambda) \cdot \mathrm{H}\left(\frac{3-\mu}{1+\lambda}\right) \right\} \; .$$

A partir de l'équation (7), $\log_2(q(n,m)) \leq Q + O(\log n)$, et à partir de l'équation (10) du théorème 6.4.1 nous obtenons :

$$\log_2 q(n,m) \;\leq\; h(m/n) \cdot n + O(\log n) \; .$$

Nous vérifions que $h(m_1/n)$ et $h(m_2/n)$ sont strictement plus petits que $\beta \approx 4,71066$ (constante définie dans l'équation (5)) (voir figure 6.4.2). De plus, pour chaque $m \in$

$[n-1, m_1[, h(m/n) \leq h(m_1/n)$ et pour chaque $m \in]m_2, 3n-6], h(m/n) \leq h(m_2/n)$. Donc,

$$\forall\, m \in [n-1, m_1[\cup]m_2, 3n-6], \quad q(n,m) \;\leq\; 2^{(\beta-\delta)n} \;\leq\; \frac{g(n)}{n!} 2^{-\delta n}$$

où δ est une constante telle que $0 < \delta < \beta$. Observons que pour $n' = n - k + 1$ et $k \in [k_1, k_2]$, nous avons $m \in [n'-1, 3n'-6]$. Puisque la fonction $(\frac{g(n)}{n!}).2^{-\delta n}$ est croissante en n, et que $k_2 - k_2 \leq 3n$, on obtient la majoration suivante :

$$\sum_{k=k_1}^{k_2} q(n-k+1, m) \;\leq\; 3n \cdot \frac{g(n-k+1)}{(n-k+1)!} \cdot 2^{-\delta(n-k+1)} \;\leq\; 3n \cdot \frac{g(n)}{n!} \cdot 2^{-\delta n}\ .$$

Donc, en utilisant le fait que $|I| \leq 2n$, et que $p(n) \leq g(n)/n!$,

$$\begin{aligned} \sum_{m \notin I} p(n,m) \;&\leq\; \sum_{m \notin I} \left(n^2 \sum_{k=k_1}^{k_2} q(n-k+1, m) \right) \;\leq\; \sum_{m \notin I} 3n^3 \cdot \frac{g(n)}{n!} \cdot 2^{-\delta n} \\ &\leq\; 6n^4 \cdot \frac{g(n)}{n!} \cdot 2^{-\delta n} \;=\; o\left(\frac{g(n)}{n!}\right) \;=\; o(p(n))\ . \end{aligned}$$

De manière analogue, comme $l(n) \geq g(n)$,

$$\sum_{m \notin I} l(n,m) \;\leq\; 6n^4 \cdot g(n) \cdot 2^{-\delta n} \;=\; o(g(n)) \;=\; o(l(n))\ .$$

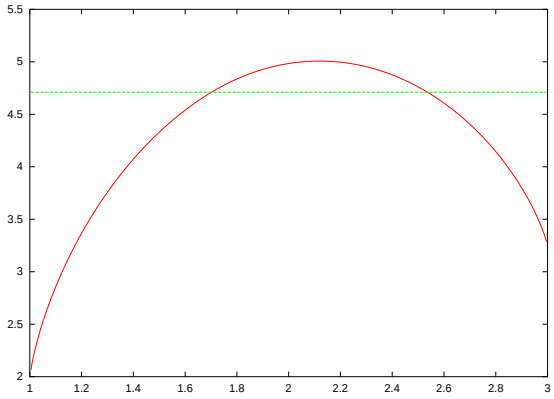

Figure 69 – Comportement of $h(\mu)$.

Ceci prouvant l'équation (11).

Evidemment, ce résultat peut être étendu à tout ensemble $A_{n,m}$ (resp. $B_{n,m}$) de graphes planaires non étiquetés (resp. étiquetés) à n sommets et m arêtes vérifiant :

$$\sum_{m \geq 0} |A_{n,m}| \;=\; \Omega\left(\frac{g(n)}{n!}\right) \quad \text{or} \quad \sum_{m \geq 0} |B_{n,m}| \;\geq\; \Omega\left(g(n)\right)\ .$$

En fait nous avons de manière similaire,

$$\sum_{m \notin I} |A_{n,m}| \;\leq\; \sum_{m \notin I} p(n,m) \;=\; o\left(\frac{g(n)}{n!}\right) \;=\; o\left(\sum_{m \geq 0} |A_{n,m}|\right) \;.$$

La même équation est également vérifiée pour $B_{n,m}$. En particulier elle est vérifiée pour les graphes planaires connexes étiquetés et connexes non étiquetés.

□

6.5 Conclusion

Un problème intéressant laissé ouvert dans ce chapitre est de déterminer si l'approche utilisée ici (représenter un graphe planaire par une triangulation appropriée et l'ensemble des arêtes devant être supprimées) permet d'atteindre la borne optimale sur le nombre de bits suffisants pour coder d'un graphe planaire à n sommets.

Nous pensons que la borne peut être améliorée en utilisant à nouveau l'approche par triangulation. En effet, notre codage des triangulations utilise $3,37n$ bits alors que $3,24n$ bits sont théoriquement suffisants. Un gain hypothétique de $0,13n$ bits serait possible nous amenant ainsi à un codage des graphes planaires en $(5,00 - 0,13)n = 4,87n$ bits. Indépendamment le nombre de feuilles l de T_2 dans une triangulation aléatoire est proche de la valeur obtenue dans notre codage dans le pire des cas : $l \approx 0,62n$. L'ensemble M_S pourrait donc être codé avec $n + l = 1,62n$ bits (nous avons besoin de n bits pour décrire les arêtes de T_2 qui sont dans G et l bits pour les arêtes de T_1 quittant une feuille de T_2 qui sont également dans G). Ceci nous donnerait un codage en $(3,24 + 1,62)n = 4,86n$ bits pour les graphes planaires aléatoires. Comme une conjecture doit être simple, et puisque nous avons la coïncidence numérique suivante, $4,85 \approx 5H(2/5)$, nous proposons :

Conjecture 6.5.1. *Pour n assez grand,*

$$p(n) \leq \binom{5n}{2n},$$

où $p(n)$ désigne le nombre de graphes planaires non étiquetés à n sommets.

Bibliographie

[1] V. A.Liskovets and T. R. Walsh. Ten steps to counting planar graphs. *Congressus Numerantium*, 60 :269–277, 1987.

[2] L. Alonso, J. L. Rémy, and R. Schott. A linear-time algorithm for the generation of trees. *Algorithmica*, 17(2) :162–182, 1997.

[3] L. Alonso and R. Schott. *Random generation of trees*. Kluwer academic publishers, 1995.

[4] D. B. Arnold and M. R. Sleep. Uniform random generation of balanced parenthesis strings. *ACM Trans. Programming Languages and Systems*, 2(1) :122–128, 1980.

[5] C. Banderier, P. Flajolet, G. Schaeffer, and M. Soria. Planar maps and airy phenomena. In 27^{th} *International Colloquium on Automata, Languages and Programming (ICALP)*, volume 1853 of Lecture Notes in Computer Science, pages 388–402. Springer, July 2000.

[6] E. Barcucci, A. Del Lungo, and E. Pergola. Random generation of trees and other combinatorial objects. *Theoretical Computer Science*, 218(2) :219–232, 1999.

[7] G. Di Battista, R. Tamassia, and L. Vismara. Output-sensitive reporting of disjoint paths. *Algorithmica*, 23(4) :302–340, 1999.

[8] E. A. Bender, Z. Gao, and N. C. Wormald. The number of labeled 2-connected planar graphs, 1999. Manuscript.

[9] T. C. Biedl, G. Kant, and M. Kaufmann. On triangulating planar graphs under the four-connectivity constraint. *Algorithmica*, 19(4) :427–446, 1997.

[10] G. Birkhoff. On the combination of subalgebras. *Proc. Cambridge Phil. Soc.*, 29 :441–464, 1933.

[11] M. Bodirsky and M. Kang. Generating random outerplanar graphs. the 1st Workshop on Algorithms for Listing, Counting, and Enumeration (ALICE03), January 12-14, 2003.

[12] H. L. Bodlaender and G. Kant. Triangulating planar graphs while minimizing the maximum degree. *Information and Computation*, 135(1) :1–14, 1997.

[13] B. Bollobás. *Extremal Graph Theory*. Academic Press, New York, 1978.

[14] N. Bonichon. A bijection between realizers of maximal plane graphs and pairs of non-crossing dyck paths. In *Proceedings of FPSAC'02*, pages 123–132, 2002.

[15] N. Bonichon, C. Gavoille, and N. Hanusse. An information-theoretic upper bound of planar graphs using triangulation. In 20^{th} *Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume Lecture Notes in Computer Science. Springer, February 2003. To appear.

[16] N. Bonichon, B. Le Saëc, and M. Mosbah. Optimal area algorithm for planar polyline drawings. In *Graph-Theoretic Concepts in Computer Science (WG 2002)*, volume 2573 of *LNCS*, 2002.

[17] N. Bonichon, B. Le Saëc, and M. Mosbah. Wagner's theorem on realizers. In *International Colloquium on Automata, Languages and Programming 2002 (ICALP'02)*, volume 2380 of *LNCS*, pages 1043–1053, 2002.

[18] N. Bonichon and M. Mosbah. Watermelon uniform random generation with applications. *Theoretical Computer Science*, to appear.

[19] N. Bonichon, B. Le Saëc, and M. Mosbah. Orthogonal drawings based on the stratification of planar graphs. Technical Report RR-1246-00, LaBRI, 2000.

[20] M. Bousquet-Mélou, 2002. Communication personnelle.

[21] R. Brak and J. W. Essam. Return polyomials for non-intersecting paths above a surface on a directed square lattice. *J. Phys. A - Math. Gen.*, 34 :10763–10782, 2001.

[22] C. C. Cheng, C. A. Duncan, M. T. Goodrich, and S. G. Kobourov. Drawing planar graphs with circular arcs. In *Graph Drawing (Proc. GD '99)*, volume 1731 of *Lecture Notes in Computer Science*, pages 117–126. Springer-Verlag, 1999.

[23] Y.-T. Chiang, C.-C. Lin, and H.-I. Lu. Orderly spanning trees with applications to graph encoding and graph drawing. In *Proc. 12th Symp. Discrete Algorithms*, pages 506–515. ACM and SIAM, 2001.

[24] N. Chiba, T. Nishizeki, S. Abe, and T. Ozawa. A linear algorithm for embedding planar graphs using pq-trees. *Journal of Computer and System Sciences*, 30(1) :54–76, 1985.

[25] M. Chrobak and S. Nakano. Minimum-width grid drawings of plane graphs. In *Graph Drawing (Proc. GD '94)*, pages 104–110, 1995.

[26] RCN Chuang, A. Garg, X. He, MY Kao, and HI Lu. Compact encodings of planar graphs via canonical ordering and multiple parentheses. In *Proc. 25th International Colloquium on Automata, Languages, and Programming (ICALP'98)*, volume 1443, pages 118–129, 1998.

[27] N.G. De Bruijn, D. E. Knuth, and S. O. Rice. The average height of planted plane trees. *In Graph Theory and Computing*, pages 15–22, 1972.

[28] M. P. Delest and G. Viennot. Algebraic languages and polyominoes enumeration. *Theoret. Comput. Sci.*, 34 :169–206, 1984.

[29] A. Denise. Génération aléatoire et uniforme de mots. *Discrete Mathematics*, 153 :69–84, 1996.

[30] A. Denise. Génération aléatoire uniforme de mots de langages rationnels. *Theoret. Comput. Sci.*, 159 :43–63, 1996.

[31] A. Denise, M. Vasconcellos, and D.J.A. Welsh. The random planar graph. *Congressus Numerantium*, 113 :61–79, 1996.

[32] M. Desainte–Catherine. A honeycomb graph perfect matchings enumeration. *Journal of Mathematical Chemistry*, 13 :133–143, 1993.

[33] M. Desainte–Catherine and G. Viennot. Enumeration of certain Young tableaux with bounded height. *Combinatoire énumérative*, pages 58–67, 1986.

[34] L. Devroye. *Non-uniform random variate generation.* Springer Verlag, 1986.

[35] A. K. Dewdney. Wagner's theorem for torus graphs. *Discrete Math.*, 4 :139–149, 1973.

[36] S. Dulucq and O. Guibert. Stack words, standard tableaux and Baxter permutations. *Discrete Mathematics*, 157 :91–106, 1996.

[37] S. Dulucq and O. Guibert. Baxter permutations. *Discrete Mathematics*, 180 :139–150, 1998.

[38] S. Eliahou. Signed diagonal flips and the four color theorem. *Europ. J. Combinatorics*, 20 :641–646, 1999.

[39] S. Eliahou, S. Gravier, and C. Payan. Three moves on signed surface triangulations. *Les cahiers du laboratoire leibniz*, 2000.

[40] P. Epstein and J.-R. Sack. Generating triangulations at random. *ACM Trans. Model. and Comput. Simul.*, 4 :267–278, 1994.

[41] P. Erdös and A. Renyi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci.*, 5 :17–61, 1960.

[42] J. W. Essam and A. J. Guttmann. Vicious walkers and directed polymer networks in general dimensions. *Phys. Rev.*, E 52 :5849–5862, 1995.

[43] I. Fary. On straight lines representation of planar graphs. *Acta Sci. Math. Szeged*, 11 :229–233, 1948.

[44] S. Felsner. Convex drawings of planar graphs and the order dimension of 3-polytopes. *Order*, 18 :19–37, 2001.

[45] S. Felsner. Geodesic embeddings of planar graphs, 2001. Draft.

[46] S. Felsner. Lattice structures from planar graphs. Preprint, 2002.

[47] M. E. Fisher. Walks, walls, wetting, and melting. *J. Stat. Phys.*, 34 :667–729, 1984.

[48] H. de Fraysseix and P. Ossona de Mendez. Regular orientations, arboricity and augmentation. In *proc. of Graph Drawing '94*, pages 111–118, 1995.

[49] H. de Fraysseix and P. Ossona de Mendez. On topological aspects of orientations. *Discrete Mathematics*, 229(1-3) :57–72, 2001.

[50] H. de Fraysseix and P. Ossona de Mendez, 2002. Communication personnelle.

[51] H. de Fraysseix, J. Pach, and J. Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10 :41–51, 1990.

[52] G. N. Frederickson and R. Janardan. Efficient message routing in planar networks. *SIAM Journal on Computing*, 18(4) :843–857, August 1989.

[53] Z. Gao, J. Urrutia, and J. Wang. Diagonal flips in labelled planar triangulations. *Graphs and Combinatorics*, 17(4) :647–657, 2001.

[54] C. Gavoille and N. Hanusse. Compact routing tables for graphs of bounded genus. In Jiří Wiedermann, Peter van Emde Boas, and Mogens Nielsen, editors, 26^{th} *International Colloquium on Automata, Languages and Programming (ICALP)*, volume 1644 of Lecture Notes in Computer Science, pages 351–360. Springer, July 1999.

[55] S. Gerke and C. McDiarmid. Adding edges to planar graphs, 2001. Preprint.

[56] E.N. Gilbert. Gray codes and paths on the n-cube. *Bell Systems Technical Journal*, 37 :815–826, 1958.

[57] D. Gouyou-Beauchamps. Chemins sous-diagonaux et tableaux de Young. In *Colloque de Combinatoire Enumérative, Montréal, UQAM 1985, Lecture Notes in Mathematics*, volume 1234, pages 112–125, 1986.

[58] D. Gouyou-Beauchamps. Standard Young tableaux of height 4 and 5. *Europ. J. Combinatorics*, 10 :69–82, 1989.

[59] Ronald L. Graham, Martin Grötschel, and Lászlo Lovász, editors. *Handbook of Combinatorics, Volume I.* Elsevier (North-Holland) ; The MIT Press, Cambridge, Massachusetts, 1995.

[60] T. Granlund. The GNU Multiple Precision arithmetic library, http ://www.gnu.org/manual/gmp/ps/gmp.ps.gz, 1996.

[61] S. Gravier and C. Payan. Flips signés et triangulations d'un polygone. *Les cahiers du laboratoire leibniz*, 2000.

[62] C. Gutwenger and P. Mutzel. Planar polyline drawings with good angular resolution. In S. Whitesides, editor, *Graph Drawing (Proc. GD '98)*, volume 1547 of *Lecture Notes in Computer Science*, pages 167–182. Springer-Verlag, 1998.

[63] X. He, MY Kao, and HI Lu. A fast general methodology for information-theoretically optimal encodings of graphs. *SIAM Journal on Computing*, 30(3) :838–846, 2000.

[64] G. Kant. Drawing planar graphs using the canonical ordering. *Algorithmica*, 16 :4–32, 1996.

[65] K. Keeler and J. Westbrook. Short encodings of planar graphs and maps. *Discrete Applied Mathematics*, 58 :239–252, 1995.

[66] A. Khodakovsky, P. Alliez, M. Desbrun, and P. Schröder. Near-optimal connectivity encoding of 2-manifold polygon meshes. *Graphical Models*, 2002. To appear in a special issue.

[67] D. King and J. Rossignac. Guaranteed 3.67V bit encoding of planar triangle graphs. In 11*th Canadian Conference on Computational Geometry (CCCG)*, pages 146–149, August 1999.

[68] C. Krattenthaler. Another involution principle-free bijective proof of stanley's hook-content formula. *J. Combin. Theory Ser. A*, 88 :66–92, 1999.

[69] C. Krattenthaler, A. J. Guttmann, and X. G. Viennot. Vicious walkers, friendly walkers and Young tableaux ii : with a walls. *J. Phys. A ; Math. Gen.*, 33 :8123–8135, 2000.

[70] C. Liao, H. Lu, and H. C. Yen. Floor-planning via orderly spanning trees. In *Proce. 9th International Symposium on Graph Drawing (GD 2001). September 23-26 2001, Vienna, Austria.*, volume 2265 of *Lecture Notes in Computer Science*, pages 367–377. Springer-Verlag, 2002.

[71] R. J. Lipton and R. E. Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36(2) :177–189, April 1979.

[72] HI Lu. Improved compact routing tables for planar networks via orderly spanning trees. In 8^{th} *Annual International Computing & Combinatorics Conference (COCOON)*, volume 2387 of *LNCS*, pages 57–66. Springer, August 2002.

[73] HI Lu. Linear-time compression of bounded-genus graphs into information-theoretically optimal number of bits. In 13^{th} *Symposium on Discrete Algorithms (SODA)*, pages 223–224. ACM-SIAM, January 2002.

[74] K. Mehlhorn and St. Näher. *The LEDA Platform of Combinatorial and Geometric Computing*. Cambridge University Press, 1999.

[75] E. Miller. Planar graphs as minimal resolutions of trivariate monomial ideals. *Documenta Mathematica*, 7 :43–90, 2002.

[76] J. I. Munro and V. Raman. Succinct representation of balanced parentheses, static trees and planar graphs. In 38^{th} *Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 118–126. IEEE Computer Society Press, October 1997.

[77] T. V. Narayana. A partial order and its application to probability. *Sankhya*, 21 :91–98, 1959.

[78] S. Negami and S. Watanabe. Diagonal transformations of triangulations of surfaces. *Tsukuba J. Math.*, 135 :225–232, 1990.

[79] P. Ossona de Mendez. *Orientations bipolaires*. PhD thesis, Ecole des Hautes Etudes en Sciences Sociales, 1994.

[80] D. Osthus, H. J. Prömel, and A. Taraz. On random planar graphs, the number of planar graphs and their triangulations. J. Combinatorial Theory, Series B, à paraître.

[81] R. Pagh. Low redundancy in static dictionaries with constant query time. *SIAM Journal on Computing*, 31(2) :353–363, 2001.

[82] J.G. Penaud and O. Roques. Tirage à pile ou face de mots de Fibonacci. *Publications du LaCIM*, 27, 2000.

[83] J. Propp. Lattice structure for orientations of graphs. Manuscript, 1993.

[84] J. Rossignac. Edgebreaker : Connectivity compression for triangle meshes. *IEEE Transactions on Visualization and Computer Graphics*, 5(1) :47–61, 1999.

[85] G. Schaeffer. Random sampling of large planar maps and convex polyhedra. In 31^{st} *Annual ACM Symposium on Theory of Computing (STOC)*, pages 760–769, Atlanta, Georgia, May 1999.

[86] W. Schnyder. Planar graphs and poset dimension. *Order*, 5 :323–343, 1989.

[87] W. Schnyder. Embedding planar graphs on the grid. In *Proc. 1st ACM-SIAM Symp. Discrete Algorithms*, pages 138–148, 1990.

[88] S.K. Stein. Convex maps. In *Proc. Amer. Math. Soc.*, volume 2, pages 464–466, 1951.

[89] M. Thorup. Compact oracles for reachability and approximate distances in planar digraphs. In 42^{th} *Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 242–251. IEEE Computer Society Press, October 2001.

[90] G. Turán. Succinct representations of graphs. *Discrete Applied Mathematics*, 8 :289–294, 1984.

[91] W. T. Tutte. A census of planar triangulations. *Canadian Journal of Mathematics*, 14 :21–38, 1962.

[92] G. Viennot. A bijective proof for the number of Baxter permutations. In *Séminaire Lotharingien de Combinatoire, Le Klebach (1981)*, 1981.

[93] K. Wada, Y. Nagata, and W. Chen. An optimal fault-tolerant routing for triconnected planar graphs. In *Graph-Theoretic Concepts in Computer Science (WG'1999)*, volume 1665 of *LNCS*, pages 191–201, 1999.

[94] K. Wagner. Bemerkungen zum vierfarbenproblem. In *Jahresber. Deutsche Math. -Verein.*, volume 46, pages 26–32, 1936.

[95] H. Whitney. A set of topological invariants for graphs. *Amer. J. Math*, 55 :231–235, 1933.

[96] H.S. Wilf. A unified setting for sequencing, ranking, and selection algorithms for combinatorial objects. *Advances in Mathematics*, 24 :281–291, 1977.

[97] D. Wilson. Mixing times of lozenge tiling and card shuing Markov chains. The Annals of Applied Probability, à paraître.

[98] D. Wilson. Determinant algorithms for random planr structures. In *Proc. 18th Symp. Discrete Algorithms*, pages 258–267. ACM and SIAM, 1997.

[99] M. Yannakakis. Embedding planar graphs in four pages. *Journal of Computer and System Sciences*, 38 :36–67, 1989.

Index

S

T

V

W