N° d'ordre : 1244

THÈSE

PRÉSENTÉE À

L'UNIVERSITÉ BORDEAUX I

ÉCOLE DOCTORALE DE MATHÉMATIQUES ET D'INFORMATIQUE

Par **François PELLEGRINI**

POUR OBTENIR LE GRADE DE

DOCTEUR

SPÉCIALITÉ : INFORMATIQUE

Application de méthodes de partition à la résolution de problèmes de graphes issus du parallélisme

---

Soutenue le : 25 janvier 1995

Après avis des rapporteurs :

Catherine ROUCAIROL	Professeur
Patrick SOLÉ	Chargé de Recherches CNRS

Devant la commission d'examen composée de :

André ARNOLD	Professeur	Président
André RASPAUD	Professeur	Rapporteur
Michel COSNARD	Professeur	Examinateurs
Jean ROMAN	Professeur	
Catherine ROUCAIROL	Professeur	
Patrick SOLÉ	Chargé de Recherches CNRS	

– 1995 –

Je tiens vivement à remercier ici :

André Arnold, professeur à l'Université Bordeaux I, pour l'honneur qu'il me fait en acceptant de présider ce jury ;

Michel Cosnard, professeur à l'E.N.S. Lyon, pour avoir bien voulu être membre de ce jury ;

André Raspaud, professeur à l'Université Bordeaux I, pour avoir accepté d'être membre de ce jury, et pour la grande rigueur scientifique qu'il a montrée chaque fois que j'ai eu l'occasion de travailler avec lui. Je lui suis très reconnaissant de l'attention qu'il a portée à mes travaux ;

Catherine Roucairol, professeur à l'Université Versailles Saint-Quentin, pour avoir bien voulu rapporter ce travail. Ses nombreuses remarques m'ont permis de compléter utilement ce document ;

Jean Roman, professeur à l'E.N.S.E.R.B., pour avoir encadré et dirigé mon travail avec sérieux et grande compétence pendant ces trois années, et pour avoir supporté avec stoïcisme le cosmos, la cotte de mailles, le pakistanais, l'esthétique, et bien d'autres surprises. Jean Roman et moi "étions sur un bateau..." ...qui touche terre maintenant ;

Patrick Solé, chargé de recherches au C.N.R.S., pour avoir accepté de rapporter cette thèse, et m'avoir fait découvrir l'existence des méthodes de point col.

Ce travail n'aurait pu se faire dans d'aussi bonnes conditions sans l'intervention, consciente ou inconsciente, de personnes que je veux également remercier ici.

Ma première pensée va à Dominique : ça y est, c'est mon tour, comme tu vois.

Merci à tous les membres successifs du $319^{ième}$ bureau, j'ai nommé Makan, Manu, Miguel, Mustapha, et Zorro, pour leurs conseils, astuces, et délires salvateurs.

Sous les prétextes les plus variés (mais toujours dans l'idée de fuir ma console `brol` et mon travail), j'ai effectué de nombreuses escapades, diurnes ou nocturnes, dans des bureaux amis.
Merci donc à Nadine, Isabelle, et Alain du 309 pour leur accueil chaleureux toutes les fois qu'il me prenait envie de les déranger. Le bureau 325 a su se forger une réputation non surfaite de halte gastronomique, avec des spécialités aussi recherchées que les sablés à l'alsacienne ou les Car-En-Sac. Merci donc à Sandrine (plus que 9 semaines et $\frac{1}{2}$!), Carole, Christophe, et Philippe ; mention spéciale Trois Étoiles au Michelin de l'amitié pour Gratex le purificateur, qui a su me remonter le moral dans la dernière ligne droite. Il existe une contrée étrange, dont peu sont revenus ; c'est le bureau 119, dont les pittoresques et délirants habitants se nomment Abdel, Cyrille, Ousmane, et Philippe. Salut à eux !

Merci a Pierre pour les discussions culturelles, et à David pour la pratique de la langue de Shakespeare à la mode Capone.

Je remercie les équipes parallélisme et GRI du LaBRI de m'avoir accueilli durant ces trois années. Bienvenue à Fred le petit nouveau (petit, mais costaud !).

Ami lecteur, vous n'auriez certainement pas pu lire ce que vous tenez entre vos mains sans l'aide ô combien précieuse de Pierre, capable de changer la cartouche de toner d'une imprimante en un temps négatif. Merci également à Anne (Bouh! Hé-hé-hé...), Marie-Laure, et Babb's, qui ont lutté contre la théorie "E.M." qui veut que tout plante quand on en a le plus besoin.
La théorie ci-dessus s'appliquant à tout le matériel, du photocopieur à l'agrafeuse, merci à tous les membres du secrétariat pour leur assistance.

Merci beaucoup aux membres de l'équipe système du service informatique du CESTA qui, pendant l'année que j'ai passée parmi eux, m'ont permis de ne pas interrompre totalement mes travaux, et ont contribué, par le temps qu'ils m'ont accordé, à la rédaction de cette thèse. Merci donc pour leur accueil à Nathalie, Eric, Patrice, Patrice, Philippe, Xavier, Jean-Claude Mignard, et Henri Dumont.

Merci à mon frère Yves pour ses conseils et références fort utiles.

Merci également, très beaucoup très fort, à Liliane, et bienvenue à tous les chatons et souricettes futurs...

Table des matières

Liste des figures

Liste des tables

Chapitre 1

Introduction générale

Une des grandes stratégies de résolution de problèmes est la technique appelée "*diviser pour résoudre*" ("*divide and conquer*"). Elle consiste à diviser le problème étudié en sous-problèmes plus petits, dont la résolution permet de construire la solution du problème initial.
Dans le cas où les sous-problèmes générés sont de même nature que le problème original, ce processus peut être appliqué récursivement, jusqu'à ce que leur taille permette leur résolution par une méthode directe.

Lorsque les problèmes étudiés peuvent être modélisés en termes de graphes, la méthode "diviser pour résoudre" se traduit par le partitionnement sous contraintes de ces graphes. Il est alors souvent intéressant algorithmiquement de travailler sur les graphes quotients des graphes originaux par rapport à ces partitions.

L'objectif de cette thèse est d'étudier l'impact des méthodes de quotientement sur plusieurs problèmes issus du parallélisme et que nous modélisons sous forme de graphes.

La première partie de la thèse, essentiellement théorique, est consacrée à l'étude de deux paramètres formels des graphes, qui sont la largeur de bande et la largeur de coupe. Ces paramètres interviennent dans de nombreux problèmes décrits en termes de graphes, tels l'optimisation de méthodes de résolution de systèmes linéaires ou l'implantation *V.L.S.I.*. Nous nous intéressons ici à leur encadrement pour des familles de graphes de réseaux d'interconnexion.
En nous basant sur la notion de graphe quotient, nous exprimons la largeur de bande d'un graphe en fonction de celle de ses graphes quotients et des sous-graphes induits par les quotientements, et faisons de même pour la largeur de coupe.
Ces résultats généraux sont appliqués, dans le cadre d'études spécifiques, à différentes familles de graphes. Nous retrouvons des bornes déjà connues pour les graphes de grille bidimensionnelle et les hypercubes, et démontrons de nouvelles majorations pour les graphes "*Butterfly*", FFT, CCC, et de De Bruijn binaires.
Pour expliciter certaines majorations, calculées sous la forme de sommes discrètes et inexploitables

en l'état, nous passons du domaine discret au domaine continu. Nous obtenons ainsi des équivalents asymptotiques des majorants étudiés. Ceci nous permet de proposer un équivalent asymptotique de nos majorations des largeurs de bande et de coupe du graphe de De Bruijn d-aire.

La deuxième partie traite du problème du placement statique sur machine parallèle. Le problème d'optimisation combinatoire consistant à placer les processus communicants d'un programme parallèle sur une machine parallèle de manière à minimiser le temps global d'exécution de celui-ci est, en toute généralité, NP-complet. Si l'on ne considère pas les dépendances logiques et temporelles entre processus, et si l'on suppose que tous les processus coexistent simultanément sur la machine parallèle pendant toute la durée d'exécution du programme, on se ramène à un problème appelé *placement statique*, lui aussi NP-complet. De fait, de nombreuses heuristiques de placement statique ont été proposées dans le but d'obtenir en un temps raisonnable des solutions sous-optimales acceptables.
Nous présentons ici un algorithme de placement basé sur le bipartitionnement récursif du graphe de communication des processus et du graphe modélisant l'architecture de la machine parallèle. Son originalité provient de la bipartition conjointe des graphes de processus et d'architecture effectuée à chaque étape de la récursion, qui permet de placer tout graphe de processus valué sur tout type d'architecture.
La présentation de notre algorithme et des méthodes de bipartitionnement qu'il utilise est suivie de multiples tests destinés à valider nos choix, qui montrent la grande efficacité de notre programme.

Dans le chapitre suivant, nous donnons les définitions et notations générales communes aux deux parties de la thèse. Chaque partie débute par une présentation du problème traité, suivie des définitions qui y sont propres, et finit par une conclusion spécifique.

Chapitre 2

Définitions et notations générales

Nous présentons dans ce chapitre les définitions et notations utilisées dans les deux parties de cette thèse.

2.1 Rappels de théorie des graphes

Le lecteur souhaitant une référence complète pourra consulter [10].

Un *graphe non-orienté* $G(V, E)$ est une structure composée d'un ensemble V d'éléments appelés *sommets*, que nous supposerons ici fini, et d'un ensemble E de paires de sommets appelées *arêtes*. Pour désigner respectivement les ensembles de sommets et d'arêtes d'un graphe G, on pourra utiliser les notations $V(G)$ et $E(G)$.
Le nombre de sommets d'un graphe est appelé *ordre* de ce graphe.

Une arête $\{v', v''\}$ est dite *incidente* à v' et à v''. v' et v'' sont les *extrémités* de $\{v', v''\}$, et sont dits *adjacents*. Une arête de la forme $\{v, v\}$ est appelée *boucle*. Une arête existant en plusieurs exemplaires dans l'ensemble des arêtes est une *arête multiple*.

Un graphe *simple* est un graphe sans boucles ni arêtes multiples. Par la suite, et sauf mention explicite, nous ne traiterons que de graphes simples.

Soit v un sommet d'un graphe G. Le *degré* de v, noté $\delta(v)$, est le nombre d'arêtes de $E(G)$ incidentes à v. Cette notion est étendue au graphe entier : le degré minimal de G, $\delta(G)$, est le minimum sur tous les sommets v de $V(G)$ de $\delta(v)$. De manière similaire, le degré maximal de G, noté $\Delta(G)$, est le maximum de $\delta(v)$ sur tous les sommets v de $V(G)$.

Une *chaîne* entre deux sommets v' et v'' d'un graphe G est une suite $\{\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{k-1}, v_k\}\}$ d'arêtes de $E(G)$ telle que $v' = v_1$ et $v'' = v_k$. Le nombre d'arêtes d'une chaîne est appelé *longueur* de la chaîne.

La *distance* entre deux sommets v' et v'' d'un graphe G est la longueur de la plus courte chaîne existant entre v' et v''.

Un graphe G est dit *connexe* s'il existe une chaîne entre toute paire de sommets du graphe.
Le *diamètre* d'un graphe connexe G, noté $\mathrm{diam}(G)$, est le maximum, pour toutes les arêtes $\{v', v''\}$ de G, de la distance entre v' et v''.

Un sommet v' d'un graphe connexe G est *périphérique* s'il existe un sommet v'' appartenant à G tel que la distance entre v' et v'' est égale au diamètre de G.

Deux graphes G' et G'' sont dits *isomorphes* si et seulement si il existe une bijection φ de $V(G')$ sur $V(G'')$ telle que $\{v', v''\}$ appartient à $E(G')$ si et seulement si $\{\varphi(v'), \varphi(v'')\}$ appartient à $E(G'')$.

Le *plongement* d'un graphe G dans un graphe H est un couple $\xi_{G,H} = (\tau_{G,H}, \rho_{G,H})$, où $\tau_{G,H}$ est une application injective de $V(G)$ dans $V(H)$, et $\rho_{G,H}$ est une application injective associant à toute arête $\{v', v''\}$ de G une chaîne $\rho_{G,H}(\{v', v''\})$ de H liant $\tau_{G,H}(v')$ à $\tau_{G,H}(v'')$.
La *dilatation* d'un plongement $\xi_{G,H} = (\tau_{G,H}, \rho_{G,H})$ est définie par

$$\mathrm{dil}(\xi_{G,H}) \stackrel{\mathrm{def}}{=} \max_{\{v',v''\} \in E(G)} (|\rho_{G,H}(\{v', v''\})|) \ ,$$

où $|\rho_{G,H}(\{v', v''\})|$ représente la longueur de la chaîne $\rho_{G,H}(\{v', v''\})$. C'est la longueur de la plus grande chaîne produite par le plongement.
La *congestion* d'un plongement $\xi_{G,H} = (\tau_{G,H}, \rho_{G,H})$ est

$$\mathrm{cg}(\xi_{G,H}) \stackrel{\mathrm{def}}{=} \max_{\{w',w''\} \in E(H)} (|\{\rho_{G,H}(\{v', v''\}) \,/\, \{v', v''\} \in E(G) \text{ et } \{w', w''\} \in \rho_{G,H}(\{v', v''\})\}|) \ ,$$

où $\{w', w''\} \in \rho_{G,H}(\{v', v''\})$ indique que la chaîne $\rho_{G,H}(\{v', v''\})$ utilise l'arête $\{w', w''\}$ de H. C'est le nombre maximum, pour toute arête de H, du nombre de chaînes issues du plongement qui utilisent cette arête.

2.2 Rappels de théorie des langages

Nous présentons ici les quelques notions de théorie des langages utilisées pour définir les graphes que nous manipulons. Les définitions et notations usuelles sont empruntées à [64].

Un *alphabet* $\mathcal{A}$ de taille d est un ensemble de d éléments appelés *lettres*. On notera $\mathcal{A}_d$ l'alphabet pris sur $\mathbb{Z}/\mathrm{d}\mathbb{Z}$, c'est-à-dire $\mathcal{A}_d = \{0, 1, \ldots, (d-1)\}$; on identifiera alors la lettre à sa valeur numérique.

Un *mot* m sur un alphabet $\mathcal{A}$ est une séquence de lettres $m_1 m_2 \ldots m_j$ de $\mathcal{A}$. Le nombre de lettres d'un mot est appelé *taille* du mot, et noté $|m|$. Le mot de taille 0, appelé *mot vide*, est noté $\boldsymbol{\epsilon}$.

Étant donné un alphabet $\mathcal{A}$, on note $\mathcal{A}^*$ l'ensemble des mots sur l'alphabet, et $\mathcal{A}^j \subset \mathcal{A}^*$ l'ensemble des mots de taille j donnée. Un *langage* sur $\mathcal{A}$ est un sous-ensemble de $\mathcal{A}^*$.

Soient $m' = m'_1 m'_2 \dots m'_{j'}$ et $m'' = m''_1 m''_2 \dots m''_{j''}$ deux mots sur $\mathcal{A}$. La *concaténation* de m'' à m' est le mot $m'm'' = m'_1 m'_2 \dots m'_{j'} m''_1 m''_2 \dots m''_{j''}$ de taille $j' + j''$. Le mot formé par k concaténations successives de m au mot vide est noté m^k.

Soient m' et m'' deux mots sur $\mathcal{A}$. m' est un *sous-mot* de m'' si $m'' = pm's$, avec p et s sur $\mathcal{A}$.

Si $m = m_1 m_2 \dots m_j$ est un mot sur $\mathcal{A}_d$, on note $[m]$ la valeur du nombre dont m est la représentation en base d, c'est-à-dire $m_1 d^{j-1} + m_2 d^{j-2} + \dots + m_j d^0$.

Si $m = m_1 m_2 \dots m_j$ est un mot sur $\mathcal{A}_d$, le *complémentaire* de m est le mot $\overline{m} = \overline{m_1 m_2} \dots \overline{m_j}$, avec $\overline{m_i} = (d-1) - m_i$.

Nous définissons le *poids de Hamming étendu* d'un mot m sur $\mathcal{A}_d$ comme l'entier

$$\mathcal{H}(m) \stackrel{\text{def}}{=} \sum_{j=1}^{|m|} m_j \quad .$$

Dans le cas des mots binaires (c'est-à-dire pris sur $\mathcal{A}_2$), il est équivalent au poids de Hamming proprement dit, qui est défini comme le nombre de "1" contenus dans ces mots. Il ne doit pas être confondu avec la métrique de Hamming utilisée en théorie des codes, qui correspond au nombre de différences existant entre deux mots de même taille.
Pour m appartenant à $\mathcal{A}_d^k$, on a toujours $0 \leq \mathcal{H}(m) \leq k(d-1)$.

L'ensemble des mots de $\mathcal{A}_d^k$ de poids de Hamming étendu donné h est défini comme

$$A_{d,k}(h) \stackrel{\text{def}}{=} \{m \in \mathcal{A}_d^k \ / \ \mathcal{H}(m) = h\} \quad .$$

On notera naturellement $|A_{d,k}(h)|$ le cardinal de cet ensemble. Ces définitions sont valides pour tout $h \in \mathbb{Z}$, ce qui facilitera l'écriture de nos calculs ; ainsi, $|A_{d,k}(h)|$ vaut 0 pour $h < 0$ ou $h > k(d-1)$. On note $h_{d,k}$ la valeur de h pour laquelle $|A_{d,k}(h)|$ est maximal.
On s'intéressera également au nombre de mots contenus dans une "bande" de $(d-1)$ poids de Hamming consécutifs. Nous définissons donc

$$A_{d,k}^{d-1}(h) \stackrel{\text{def}}{=} \bigcup_{j=h-(d-2)}^{h} A_{d,k}(j) \quad ,$$

et $\left|A_{d,k}^{d-1}(h)\right|$ dénombre les mots dont le poids est compris entre $h-(d-2)$ et h. De manière analogue, on note $h_{d,k}^{d-1}$ la valeur de h maximisant $\left|A_{d,k}^{d-1}(h)\right|$.

2.3 Définition de quelques graphes usuels

Nous présentons ici les graphes que nous manipulons dans les parties suivantes. Ce sont tous des graphes non orientés, simples, et sans boucles. Une description concise de leurs propriétés peut

être trouvée dans [22, pages 269–282].
Pour tous les graphes définis sur alphabets, on identifiera les sommets et les mots qui les représentent. On pourra ainsi parler du complémentaire d'un sommet et, par extension, on appellera complémentaire d'une arête, l'arête dont les sommets sont les complémentaires des sommets de l'arête initiale.

Précisons tout d'abord le sens des opérateurs mathématiques que nous utilisons couramment. Nous notons $a \bmod b$ le reste de la division euclidienne de a par b, où a et b sont deux entiers naturels. Par extension, pour a strictement positif, $(-a) \bmod b = b - (a \bmod b)$.
De même, nous notons $\oplus$ l'opérateur *ou-exclusif* sur les représentations binaires de nombres entiers.

2.3.1 La chaîne

La chaîne $\mathrm{P}(k)$ est le graphe d'ordre k dont les sommets sont numérotés de 0 à $k-1$ et dont les arêtes sont de la forme $\{i, i+1\}$, avec $0 \leq i < k-1$.

2.3.2 Le cycle

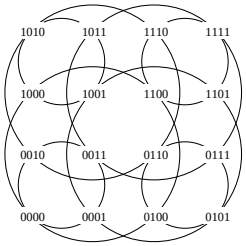
Le cycle $C(k)$ est le graphe d'ordre k dont les sommets sont numérotés de 0 à $k-1$ et dont les arêtes sont de la forme $\{i, (i+1) \bmod k\}$, avec $0 \leq i < k$.

2.3.3 Le graphe complet

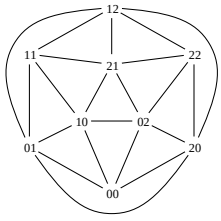
Le graphe complet $K(k)$ est le graphe d'ordre k tel que chaque sommet est lié à tous les autres.

2.3.4 La grille k-dimensionnelle

Les sommets de la grille k-dimensionnelle $\mathrm{M}_k(d_1, d_2, \ldots, d_k)$ sont les k-uplets d'entiers, notés $v = (v_1, v_2, \ldots, v_k)$, tels que $0 \leq v_i < d_i$ pour tout i compris entre 1 et k. Deux sommets sont liés par une arête si et seulement si leurs coordonnées sont respectivement égales dans $k-1$ dimensions, et diffèrent d'exactement 1 dans la dimension restante.
Par convention, on supposera toujours $d_i \leq d_{i+1}$, pour $1 \leq i < k$.



a. Le graphe H(4).



b. Le graphe UB(3, 2).

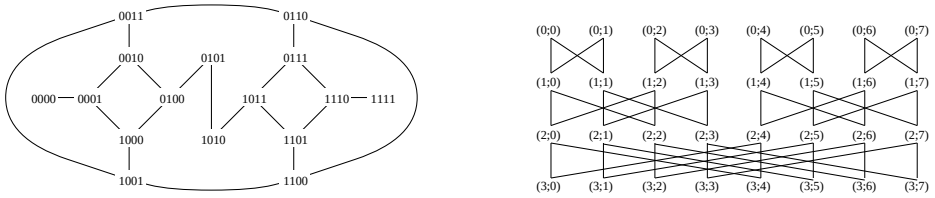
Figure 2.1: Les graphes H(4) et UB(3, 2).

2.3.5 L'hypercube

L'hypercube de dimension k, noté $\mathrm{H}(k)$, est le graphe dont les sommets sont les mots de k lettres $v = v_1 v_2 \dots v_k$ appartenant à $\mathcal{A}_2^k$, et tel qu'il existe une arête entre deux sommets si et seulement si ceux-ci diffèrent d'exactement une lettre. Il est représenté en figure 2.1.a.

2.3.6 Le graphe de De Bruijn non-orienté d-aire

Les sommets v de $\mathrm{UB}(d,k)$ sont les mots de k lettres pris dans $\mathcal{A}_d^k$, notés $v = v_1 v_2 \dots v_k$. Il existe une arête entre deux sommets distincts v' et v'' si et seulement si les $k-1$ lettres les plus à gauche de l'un des sommets sont égales aux $k-1$ lettres les plus à droite de l'autre sommet, c'est-à-dire $v'' = v'_2 v'_3 \dots v'_k x$ ou bien $v'' = x v'_1 \dots v'_{k-2} v'_{k-1}$, avec $x \in \mathcal{A}_d$. Ce graphe est dessiné en figure 2.1.a.



a. Le graphe SE(4). **b.** Le graphe FFT(3).

Figure 2.2: Les graphes SE(4) et FFT(3).

2.3.7 Le graphe "*Shuffle-Exchange*"

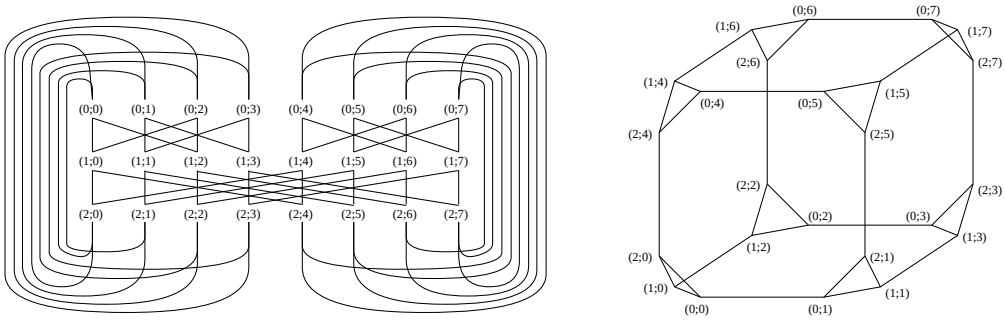
Le graphe "*Shuffle-Exchange*", noté $\mathrm{SE}(k)$, a pour sommets les mots $v = v_1 v_2 \dots v_k$ de $\mathcal{A}_2^k$, et il existe une arête entre deux sommets v' et v'' si et seulement si v'' s'obtient par un décalage cyclique des lettres de v' vers la gauche ($v'' = v'_2 \dots v'_k v_1$) ou la droite ($v'' = v'_k v'_1 \dots v'_{k-1}$), ou bien par complémentation de la dernière lettre de v' ($v'' = v'_1 v'_2 \dots v'_{k-1} \overline{v'_k}$). Un représentant de cette famille est dessiné en figure 2.2.a.

2.3.8 Le graphe "*Fast Fourier Transform*"

Les sommets de $\mathrm{FFT}(k)$ sont les couples d'entiers $(l;m)$, avec $0 \leq l \leq k$ et $0 \leq m < 2^k$. Tout sommet $(l;m)$ de $\mathrm{FFT}(k)$ est lié aux sommets $(l-1;m)$, $(l-1;m \oplus 2^{l-1})$, $(l+1;m)$, et $(l+1;m \oplus 2^l)$, s'ils existent, comme dessiné en figure 2.2.b. On dira que deux sommets $(l';m')$ et $(l'';m'')$ appartiennent au même niveau si et seulement si $l' = l''$.

2.3.9 Le graphe "*Butterfly*"

Les sommets de $\mathrm{BF}(k)$ sont les couples d'entiers $(l;m)$, avec $0 \leq l < k$ et $0 \leq m < 2^k$. Tout sommet $(l;m)$ de $\mathrm{BF}(k)$ est lié aux sommets $((l-1) \bmod k;m)$, $((l-1) \bmod k;m \oplus 2^l)$, $((l+1) \bmod k;m)$, et $((l+1) \bmod k;m \oplus 2^{(l+1) \bmod k})$, comme dessiné en figure 2.3a.
On dira des sommets $(l;m)$ ayant même valeur l qu'ils appartiennent au même niveau.



a. Le graphe BF(3). **b.** Le graphe CCC(3).

Figure 2.3: Les graphes BF(3) et CCC(3).

2.3.10 Le graphe "*Cube-Connected Cycles*"

Les sommets de CCC(k) sont les couples d'entiers $(l;m)$, avec $0 \leq l < k$ et $0 \leq m < 2^k$, et tout sommet $(l;m)$ de CCC(k) est lié aux sommets $((l-1) \bmod k; m)$, $((l+1) \bmod k; m)$, et $(l; m \oplus 2^l)$, comme illustré en figure 2.3.b. Une autre représentation de ce graphe est donnée en figure 5.13, page 44.

On dira des sommets $(l;m)$ ayant même valeur l qu'ils appartiennent au même niveau.

Partie I

Largeur de bande, largeur de coupe, et graphes quotients

Chapitre 3

Présentation du problème

3.1 Largeur de bande et largeur de coupe

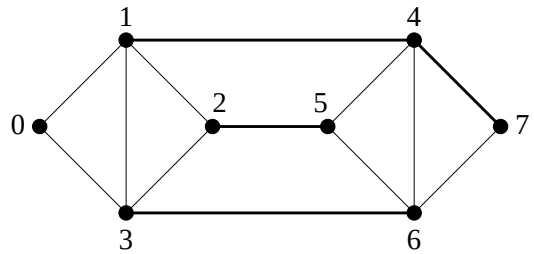
La largeur de bande et la largeur de coupe sont deux paramètres fondamentaux qui interviennent dans la formulation de nombreux problèmes modélisés par des graphes.
Soit $\Phi(G)$ l'ensemble des numérotations des sommets d'un graphe G d'ordre n, c'est-à-dire l'ensemble des bijections $\varphi : V(G) \longrightarrow \{0, \ldots, n-1\}$. La largeur de bande de G est

$$\mathrm{Bd}(G) \stackrel{\text{def}}{=} \min_{\varphi \in \Phi(G)} \left(\max_{\{v', v''\} \in E(G)} (|\varphi(v') - \varphi(v'')|) \right) \quad ,$$

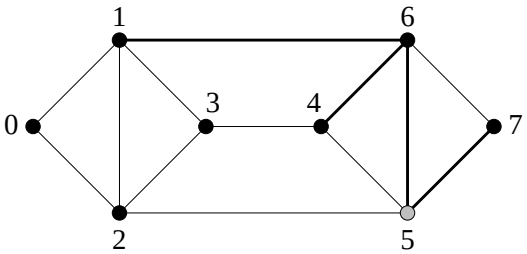
et la largeur de coupe de G est

$$\mathrm{Cw}(G) \stackrel{\text{def}}{=} \min_{\varphi \in \Phi(G)} \left(\max_{l \in \{0, \ldots, n-1\}} (|\{\{v', v''\} \in E(G) \,/\, \varphi(v') \leq l < \varphi(v'')\}|) \right) \quad .$$

Ces définitions, illustrées en figure 3.1, sont basées sur la numérotation explicite des sommets du graphe. Il en existe une autre formulation, relative au plongement de G dans la chaîne de même ordre $\mathrm{P}(n)$ [66].
Soit $\Xi(G)$ l'ensemble de tous les plongements ξ de G dans $\mathrm{P}(n)$. Les largeurs de bande et de coupe



a. $\mathrm{Bd}(\mathrm{UB}(2,3)) = 3$. Exemple de numérotation donnant cette valeur, avec en gras les arêtes concernées.



b. $\mathrm{Cw}(\mathrm{UB}(2,3)) = 4$. Exemple de numérotation donnant cette valeur, avec en gras les arêtes dont les valeurs des extrémités encadrent la valeur du sommet gris.

Figure 3.1: Largeurs de bande et de coupe de $\mathrm{UB}(2,3)$.

de G s'écrivent alors

$$\begin{aligned} \mathrm{Bd}(G) &= \min_{\xi \in \Xi(G)} (\mathrm{dil}(\xi)) \quad , \\ \mathrm{Cw}(G) &= \min_{\xi \in \Xi(G)} (\mathrm{cg}(\xi)) \quad . \end{aligned}$$

Pour retrouver les premières définitions à partir de celles-ci, il suffit de numéroter consécutivement les sommets de la chaîne en partant d'une de ses extrémités, et d'en déduire la numérotation φ qui associe à chaque sommet de G le numéro du sommet de $\mathrm{P}(n)$ qui est son image par ξ.
Inversement, les numérotations φ_G peuvent être vues comme des plongements de G dans $\mathrm{P}(n)$, ce qui fait que nous pouvons étendre les définitions de la dilatation et de la congestion à φ :

$$\mathrm{dil}(\varphi) \stackrel{\mathrm{def}}{=} \max_{\{v',v''\} \in E(G)} (|\varphi_G(v') - \varphi_G(v'')|) \quad , \tag{3.1}$$

$$\begin{aligned} \mathrm{cg}(\varphi) &= \max_{\{w',w''\} \in E(\mathrm{P}(n))} (|\{\rho_{G,\mathrm{P}(n)}(\{v',v''\}) \,/\, \{v',v''\} \in E(G) \text{ et} \\ &\qquad \{w',w''\} \in \rho_{G,\mathrm{P}(n)}(\{v',v''\})\}|) \\ &\stackrel{\mathrm{def}}{=} \max_{v \in V(G)} (|\{\{v',v''\} \in E(G) \,/\, \varphi_G(v') \leq \varphi_G(v) < \varphi_G(v'')\}|) \quad . \end{aligned} \tag{3.2}$$

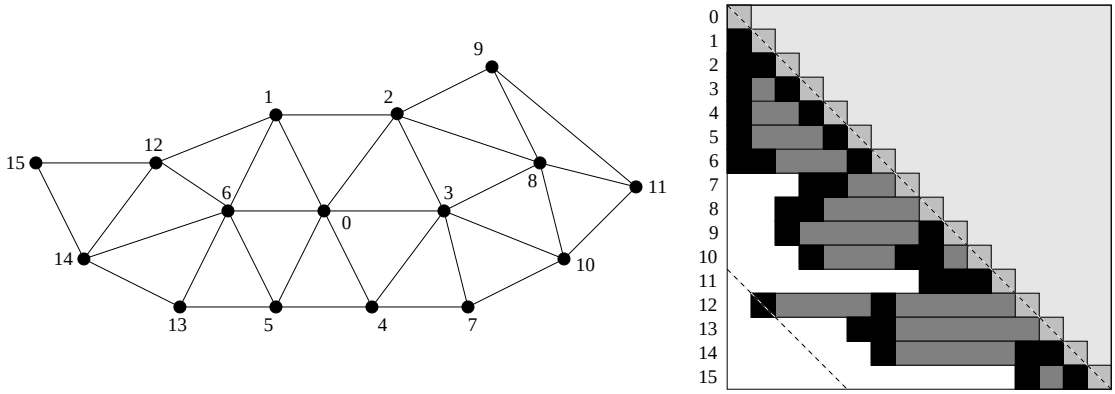
3.2 Applications

Les premiers travaux sur la largeur de bande ont été motivés par l'optimisation des algorithmes de calcul matriciel utilisés dans la conduite de simulations numériques de phénomènes physiques par méthodes de différences et d'éléments finis [35, 40, 62, 87]. Les résultats obtenus ont ensuite été étendus à la théorie des codes [46], puis à l'implantation de circuits *V.L.S.I.* [56, 66, 94], pour laquelle la largeur de coupe est également utilisée [28, 66, 78, 90, 91].
Nous présentons succintement ici deux domaines d'application des largeurs de bande et de coupe.

3.2.1 Application au calcul matriciel

La méthode habituelle pour simuler informatiquement un phénomène physique continu consiste à le discrétiser dans l'espace et dans le temps. Pour cela, on construit un graphe maillé G dont les sommets portent les quantités physiques calculées en un point de l'espace, et dont les arêtes indiquent les dépendances spatiales (voisinage) entre sommets.

Lorsque les dépendances entre les variables situées aux sommets sont linéaires, le problème peut être traité comme un système linéaire, et stocké sous forme matricielle. Comme chaque sommet du graphe n'est lié qu'à un petit nombre de voisins, chaque ligne de la matrice ne possède que peu de coefficients non nuls. Le nombre de coefficients non nuls de la matrice est donc borné par $\Delta(G)\,|V(G)|$, qui est très petit devant $|V(G)|^2$; la matrice est dite *creuse*.
Pour éviter de stocker dans une matrice pleine un grand nombre de termes nuls et d'effectuer sur eux des opérations inutiles, plusieurs méthodes ont été proposées. Le stockage du *profil* consiste à stocker chaque ligne de la matrice sous forme d'un vecteur contenant tous les termes compris entre le premier terme non-nul et la diagonale, comme illustré en figure 3.2.



a. Un graphe de maillage, et une numérotation des inconnues portées par ses sommets.

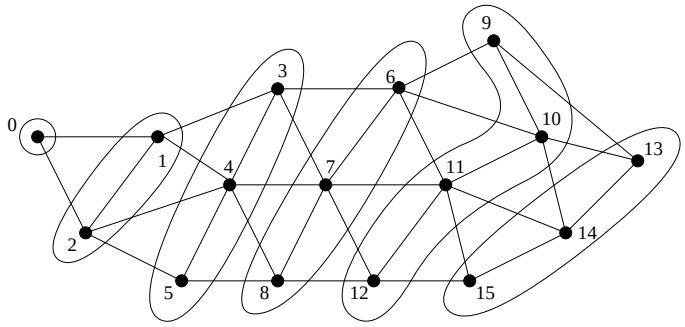
b. La matrice associée, dont seule la partie diagonale inférieure est représentée.

Figure 3.2: Graphe de maillage et matrice profil symétrique associée à la numérotation des inconnues. Les termes non-nuls sont représentés en noir, et les vecteurs profil en gris foncé.

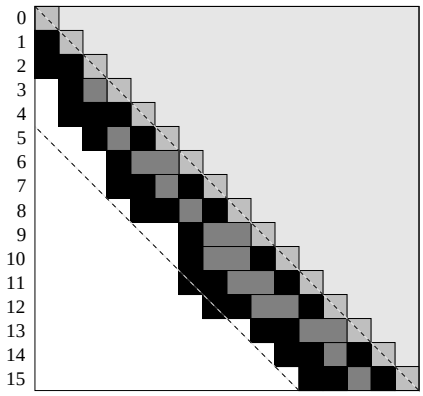
L'inconvénient des méthodes de profil est qu'elles sont très sensibles à la numérotation des inconnues, dont dépendent les tailles des vecteurs. Pour réduire la taille totale du profil, il faut minimiser la *somme* sur l'ensemble des sommets de l'écart maximal entre le numéro d'un sommet et les numéros de tous ses voisins. Cette formulation est très proche de celle de la largeur de bande, qui minimise le *maximum* des écarts maximaux.
Historiquement, avant d'utiliser le profil, les numériciens stockaient la plus petite bande sous-diagonale utile (représentée en figure 3.2.b par les traits pointillés), dont ils cherchaient à minimiser la largeur ; de là vient le nom du paramètre. Le stockage du profil a rapidement supplanté celui de la bande, du fait de la place qu'il économise lorsque les écarts maximaux de numérotation par rapport aux sommets voisins varient d'un sommet à l'autre.
L'une des heuristiques les plus utilisée pour minimiser tant la largeur de bande que le profil est celle de Gibbs, Poole, et Stockmeyer, que nous présentons plus bas (voir figure 3.3).

3.2.2 Application à l'implantation de circuits *V.L.S.I.*

Un autre grand domaine d'application des largeurs de bande et de coupe est l'implantation de circuits *V.L.S.I.*. La fabrication à moindre coût de circuits logiques complexes nécessite de respecter les contraintes inhérentes à la technologie employée. Ces contraintes incluent la minimisation de l'aire de silicium nécessaire au gravage du circuit, du nombre de couches utilisées pour router les pistes conductrices, et de la longueur maximale de celles-ci, qui conditionne la fréquence de cadencement du circuit.
On modélise habituellement le circuit électronique à implanter par un graphe G dont les sommets représentent les composants (transistors à l'échelle du micro-processeur, composants à l'échelle de la carte à circuits imprimés), et les arêtes les pistes conductrices.



a. Numérotation par couches du graphe à partir d'un sommet pseudo-périphérique.



b. La matrice associée.

Figure 3.3: Renumérotation par la méthode de Gibbs, Poole, et Stockmeyer des sommets du graphe de la figure 3.2.a, et nouvelle matrice profil associée.

Pour modéliser l'implantation *V.L.S.I.*, Monien et Sudborough [66] proposent d'ordonner les composants du circuit à placer sur la première ligne d'une grille bidimensionnelle, et de router les pistes en suivant les arêtes de la grille, sans que deux pistes partagent la même arête (voir figure 3.4). Cette implantation ne nécessite que deux couches (l'une pour router les portions horizontales des pistes, l'autre pour les portions verticales), et son aire est bornée par $\sum_{v \in V(G)} \delta(v) = 2\,|E(G)|$ multiplié par le nombre maximal d'arêtes passant les unes au dessus des autres.

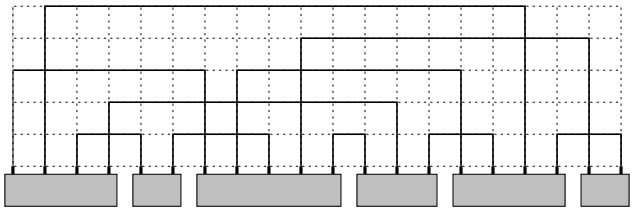


Figure 3.4: Modélisation de l'implantation *V.L.S.I.* d'un circuit.

Si l'on contracte les composants ainsi alignés, on se ramène au problème du plongement du graphe G dans la chaîne de même ordre. Le nombre d'arêtes qui se croisent est donc minimisé par la largeur de coupe de G, et l'aire de l'implantation par $2\,|E(G)|\,\mathrm{Cw}(G)$. De même, la longueur de la plus grande piste est au moins égale à $\mathrm{Bd}(G)$.
Notons cependant que les numérotations qui minimisent la largeur de bande permettent rarement de minimiser simultanément la largeur de coupe, et réciproquement.

3.3 Résultats connus

Les largeurs de bande et de coupe ont été très étudiées [16, et références contenues]. Citons en particulier quelques théorèmes généraux.

Théorème 3.1. [19] *Pour tout graphe G,*

$$\mathrm{Bd}(G) \geq |V(G)| - \frac{1+\sqrt{(2|V(G)|-1)^2 - 8|E(G)|}}{2} \ .$$

Théorème 3.2. [17] *Pour tout graphe G,*

$$\mathrm{Bd}(G) \geq \left\lceil \frac{|V(G)|-1}{\mathrm{diam}(G)} \right\rceil \ .$$

Papadimitriou [73, 36] et Gavril [38] ont montré respectivement que déterminer la largeur de bande et la largeur de coupe d'un graphe quelconque était NP-complet. De fait, l'obtension d'encadrements serrés des largeurs de bande et de coupe nécessite de restreindre le cadre de travail.

Une première approche consiste à ne s'intéresser qu'à des familles de graphes spécifiques, dont on utilise les propriétés, que ce soit dans des calculs analytiques débouchant sur l'obtension de théorèmes [8, 17, 20, 47, 70, 69], ou bien sur la construction d'algorithmes polynomiaux [18].

Pour obtenir des majorations des largeurs de bande et de coupe de graphes quelconques en temps polynomial, de nombreuses heuristiques ont également été proposées, telles celles présentées dans [35, 43, 87], ou bien celle de Gibbs, Poole, et Stockmeyer [40].

L'algorithme de Gibbs, Poole, et Stockmeyer est une heuristique de renumérotation des sommets de graphes dont le but est de réduire l'écart entre les numéros de sommets voisins. Elle s'applique autant à la réduction de profil qu'à la réduction de bande, puisque la réduction de la somme des écarts entraîne celle de leur maximum.
La première phase de l'algorithme consiste à découper le graphe considéré en couches. Partant d'un sommet quelconque du graphe, que l'on affecte à la couche de numéro 0, on construit successivement les autres couches en affectant à la couche i tous les voisins des sommets de la couche $i-1$ n'ayant pas encore été affectés (ceci revient à construire un arbre recouvrant en largeur ayant comme racine le sommet initial). Lorsque tous les sommets du graphe ont été affectés, un sommet de plus petit degré appartenant à la couche de plus grand numéro devient le nouveau sommet de départ, et le processus de découpage est réitéré tant que cela conduit à augmenter le nombre de couches. Comme l'algorithme cherche à maximiser la hauteur des arbres construits à partir des sommets de départ choisis, ceux-ci sont appelés *nœuds pseudo-périphériques*.
Une fois la phase de découpage terminée, l'algorithme numérote les sommets consécutivement, couche après couche, en partant de la couche de numéro 0, comme illustré en figure 3.3.a. La dilatation de la numérotation ainsi calculée est donc majorée par le double de la taille de la plus grande couche. En cherchant à maximiser le nombre de couches, la première phase a en fait pour but de réduire leurs tailles.

3.4 Notre travail

À plusieurs reprises [17, 40], les méthodes utilisées pour calculer la largeur de bande mettent en œuvre le regroupement de sommets afin de numéroter en deux étapes ces groupes puis les sommets

qui les constituent. Le principe de ces méthodes est de diminuer la complexité du problème de numérotation en orientant la recherche des solutions par le groupement préalable et judicieux des sommets.

En formalisant cette approche au moyen des graphes quotients, nous exprimons la largeur de bande d'un graphe en fonction de celle de ses graphes quotients et des sous-graphes induits par les quotientements (c'est-à-dire des sous-graphes partiels du graphe initial contractés en un unique sommet des graphes quotients). Nous faisons de même pour la largeur de coupe.

Ces résultats généraux sont appliqués, dans le cadre d'études spécifiques, à différentes familles de graphes modélisant les réseaux d'interconnexion de machines parallèles (c'est-à-dire dont les sommets représentent les nœuds du réseau –routeurs et processeurs de calcul– et les arêtes les liens de communication entre nœuds). Les graphes quotients utilisés à cette occasion sont soit des chaînes, soit des cycles.
Lorsque la structure des graphes étudiés s'y prête, il est possible de trouver des quotientements induisant des sous-graphes de même topologie que le graphe initial, ce qui permet d'appliquer récursivement la méthode. Les majorations que nous démontrons de cette manière ne concernent que la largeur de coupe. En effet, pour la largeur de bande, les valeurs obtenues par ce moyen se sont avérées moins bonnes que celles provenant d'un quotientement direct.
Lors du calcul des majorations de la largeur de bande, nous utilisons parfois la numérotation induite par notre quotientement pour calculer directement et plus finement la dilatation. Nos résultats généraux servent alors plus en tant que cadre méthodologique pour la recherche de numérotations que comme théorèmes directement applicables.

Nous retrouvons des bornes déjà connues sur les graphes de grille bidimensionnelle et les hypercubes, et démontrons des majorations, inédites à notre connaissance, des largeurs de bande et de coupe des graphes "*Butterfly*", FFT, CCC, et de De Bruijn binaires. Ce travail a fait l'objet d'un rapport interne [6], et a été soumis pour publication.

Pour expliciter certaines majorations, calculées sous la forme de sommes discrètes et inexploitables en l'état, nous passons du domaine discret au domaine continu. Nous utilisons pour cela une écriture du symbole de Kronecker comme intégrale d'une exponentielle complexe, ce qui nous permet de ramener des sommes imbriquées de termes entiers à l'intégrale simple de la puissance d'une somme unique. En appliquant le théorème de Laplace à ces expressions intégrales, nous obtenons des équivalents asymptotiques des majorants étudiés. Ceci nous permet de proposer un équivalent asymptotique de nos majorations des largeurs de bande [74] et de coupe du graphe de De Bruijn d-aire.

Nous prouvons également des minorations des largeurs de bande et de coupe, en utilisant les bissections sommet et arête.

Chapitre 4

Définitions et notations

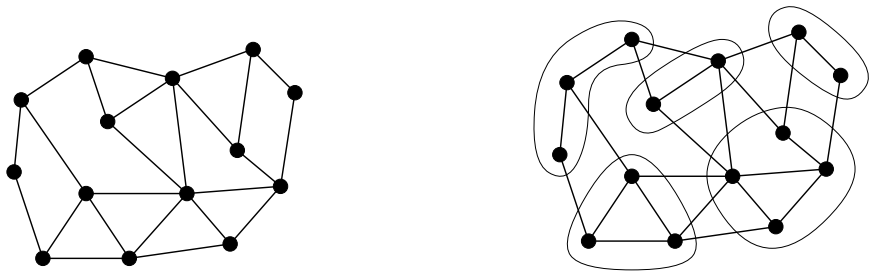
Ce chapitre est consacré à la définition des objets et outils mathématiques spécifiques que nous allons utiliser dans la suite de cette partie.

4.1 Partitionnements de graphes

Soit $V(G)$ l'ensemble non-vide des sommets d'un graphe G d'ordre n. Une *partition* Π de $V(G)$ est le découpage de $V(G)$ en N parties π, tel que :

- aucune partie n'est vide ;
- toutes les parties sont deux à deux disjointes ;
- l'union de toutes les parties contient tous les sommets de $V(G)$ (on a donc $1 \leq N \leq n$).

Un exemple de partition est donné en figure 4.1.
Nous noterons $\mathcal{P}_V(G)$ l'ensemble de toutes les partitions de $V(G)$.



a. Un graphe G. **b.** Une partition Π de G.

Figure 4.1: Partition d'un graphe.

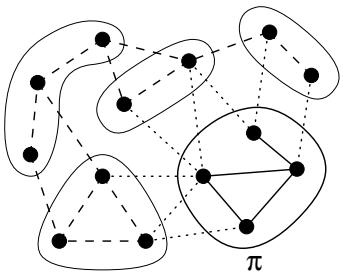
Pour tout sommet v de G, $\pi(v)$ représente la partie de Π contenant v. La taille de la plus grande partie d'une partition est notée $\max_\pi \stackrel{\text{def}}{=} \max_{\pi \in \Pi}(|\pi|)$.

Le *cocycle* $\omega(\pi)$ d'une partie π est l'ensemble des arêtes ayant exactement une de leurs extrémités dans π. Le cardinal du plus grand cocycle de la partition est $\max_\omega \stackrel{\text{def}}{=} \max_{\pi \in \Pi}(|\omega(\pi)|)$.

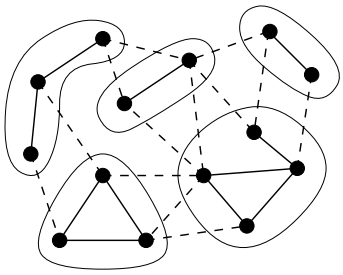
Pour toute partie π d'une partition Π de $\mathcal{P}_V(G)$, on notera $E_I(\pi)$ et $E_E(\pi)$ les ensembles des arêtes de $E(G)$ ayant respectivement leurs deux extrémités dans π (arêtes *internes* aux parties), et aucune de leurs extrémités dans π (arêtes *externes* à π). Afin d'harmoniser les notations dans les calculs, on notera parfois $E_C(\pi)$ le cocycle d'une partie π. Ces différents types d'arêtes sont représentés en figure 4.2.a.
Par extension, on définit

$$E_I(\Pi) \stackrel{\text{def}}{=} \bigcup_{\pi \in \Pi} E_I(\pi) \qquad \text{et} \qquad E_E(\Pi) \stackrel{\text{def}}{=} E(G) - E_I(\Pi) \ ,$$

qui représentent respectivement les arêtes internes et externes à l'ensemble des parties de la partition, comme illustré en figure 4.2.b.



a. Arêtes du cocycle (en pointillés), externes (en tirets), et internes (en traits pleins) à la partie π de G.



b. Extension à la partition.

Figure 4.2: Les trois types d'arêtes.

4.2 Graphes quotients et sous-graphes induits

Le *graphe quotient* $Q = G_{/\Pi}$ d'un graphe G par rapport à une partition Π de $V(G)$ (figure 4.3.b) est défini par :

$$\begin{cases} V(Q) = \Pi \\ (\{\pi', \pi''\} \in E(Q)) \iff (\pi' \neq \pi'', \exists v' \in \pi', \exists v'' \in \pi'' \,/\, \{v', v''\} \in E(G)) \ . \end{cases}$$

Pour toute partie π de Π, nous noterons $G[\pi]$ le *sous-graphe* de G *induit* par π, c'est-à-dire le graphe dont l'ensemble de sommets est π et l'ensemble d'arêtes $E_I(\pi)$ (figure 4.3.a).

4.3 Numérotations

Une *numérotation* φ des sommets d'un graphe G est une bijection de $V(G)$ vers $\{0, 1, \ldots, n-1\}$. De φ découle une orientation implicite des arêtes de G, à partir de laquelle on définit les demi-degrés

a. Le graphe $G[\pi]$. **b.** Le graphe $G_{/\Pi}$.

Figure 4.3: Graphe induit et graphe quotient issus du graphe G représenté en figure 4.2.a.

inférieurs et supérieurs des sommets du graphe. Pour tout v appartenant à $V(G)$,

$$\begin{aligned} \delta^-_\varphi(v) &\stackrel{\text{def}}{=} |\{\{v', v\} \in E(G) \,/\, \varphi(v') < \varphi(v)\}| \ , \\ \delta^+_\varphi(v) &\stackrel{\text{def}}{=} |\{\{v, v''\} \in E(G) \,/\, \varphi(v) < \varphi(v'')\}| \ . \end{aligned}$$

Si φ et Π sont respectivement une numérotation des sommets et une partition d'un graphe G, la *restriction* de φ à une partie π de G, notée $\varphi_{|\pi}$, est la numérotation de $G[\pi]$ telle que pour tous sommets v' et v'' appartenant à π, $(\varphi(v') < \varphi(v'')) \Longleftrightarrow (\varphi_{|\pi}(v') < \varphi_{|\pi}(v''))$.

Nous noterons $\Phi_Q(G, \Pi)$ l'ensemble des numérotations φ_Q du graphe quotient $Q = G_{/\Pi}$; numéroter les sommets de Q est équivalent à numéroter les parties de Π.

Soit φ_Q une numérotation d'une partition Π d'un graphe G. Pour toute partie π de Π et tout sommet v de π, on définit les *degrés entrant* et *sortant* de v *externes à* π par :

$$\begin{aligned} \delta^-_{\Pi,\varphi_Q}(v) &\stackrel{\text{def}}{=} |\{\{v', v\} \in \omega(\pi(v)) \,/\, \varphi_Q(\pi(v')) < \varphi_Q(\pi(v))\}| \ , \\ \delta^+_{\Pi,\varphi_Q}(v) &\stackrel{\text{def}}{=} |\{\{v, v''\} \in \omega(\pi(v)) \,/\, \varphi_Q(\pi(v'')) > \varphi_Q(\pi(v))\}| \ . \end{aligned}$$

Cette notation est étendue à la partie toute entière :

$$\begin{aligned} \delta^-_{\Pi,\varphi_Q}(\pi) &\stackrel{\text{def}}{=} \min_{v \in \pi} \left(\delta^-_{\Pi,\varphi_Q}(v)\right) \ , \\ \delta^+_{\Pi,\varphi_Q}(\pi) &\stackrel{\text{def}}{=} \min_{v \in \pi} \left(\delta^+_{\Pi,\varphi_Q}(v)\right) \ . \end{aligned}$$

Étant donné un graphe G, une partition Π de ses sommets en N parties, et une numérotation φ_Q des sommets du graphe quotient $Q = G_{/\Pi}$, nous appelons *numérotation induite* des sommets de G par Π et φ_Q une numérotation φ_G des sommets de G telle que

$$\forall\, \pi', \pi'' \in \Pi, \ \ (\varphi_Q(\pi') < \varphi_Q(\pi'')) \iff (\forall\, v' \in \pi',\ \forall\, v'' \in \pi'',\ \varphi_G(v') < \varphi_G(v'')) \ ,$$

ce qui équivaut à l'ensemble de propriétés suivant :

$$\begin{cases} \bigcup\limits_{v \in \pi} \{\varphi_G(v)\} = \{p_{\varphi_Q(\pi)}, \ldots, P_{\varphi_Q(\pi)}\} & \text{dans chaque partie, les numéros forment des intervalles,} \\ \bigcup\limits_{i=0}^{N-1} \{p_i, \ldots, P_i\} = \{0, \ldots, n-1\} & \text{les intervalles sont disjoints et couvrent } \{0, \ldots, n-1\}, \\ \forall i \in \{1, \ldots, N-1\},\ p_i = P_{i-1} + 1 & \text{les parties de numéros croissants donnent des intervalles de numéros croissants.} \end{cases}$$

En particulier, $p_0 = 0$, $P_0 = |\varphi_Q^{-1}(0)| - 1$, $p_1 = |\varphi_Q^{-1}(0)|$, $P_1 = |\varphi_Q^{-1}(0)| + |\varphi_Q^{-1}(1)| - 1$, ..., $P_{N-1} = n - 1$.
Nous noterons $\Phi_G(G, \Pi, \varphi_Q)$ l'ensemble de toutes les numérotations φ_G induites par un φ_Q de $\Phi_Q(G, \Pi)$. Nous appellerons alors *configuration* de G tout triplet $(\Pi, \varphi_Q, \varphi_G)$ tel que $\Pi \in \mathcal{P}_V(G)$, $\varphi_Q \in \Phi_Q(G, \Pi)$, et $\varphi_G \in \Phi_G(G, \Pi, \varphi_Q)$.

Puisque la partition à n parties $\Pi \cong V(G)$ autorise toujours des numérotations $\varphi_G \cong \varphi_Q$ qui permettent d'atteindre les largeurs de bande et de coupe de G, nous avons

$$\mathrm{Bd}(G) = \min_{\substack{(\Pi, \varphi_Q, \varphi_G) \\ \text{configuration de } G}} (\mathrm{dil}(\varphi_G)) \ , \tag{4.1}$$

$$\mathrm{Cw}(G) = \min_{\substack{(\Pi, \varphi_Q, \varphi_G) \\ \text{configuration de } G}} (\mathrm{cg}(\varphi_G)) \ . \tag{4.2}$$

4.4 Bissections

La *bissection arête* $\mathrm{Bis}_e(G)$ d'un graphe G est le cardinal du plus petit ensemble d'arêtes déconnectant G en deux sous-graphes de même ordre, à un près si l'ordre de G est impair.
La *bissection sommet* $\mathrm{Bis}_v(G)$ d'un graphe G est le cardinal du plus petit ensemble de sommets déconnectant G en deux sous-graphes de même ordre.

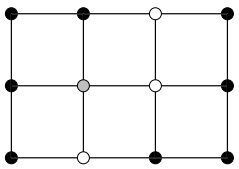


Figure 4.4: Les sommets blancs déconnectent le graphe en deux parties de cardinaux égaux à un près. Afin de les égaliser, on compte dans la bissection sommet le sommet gris.

La définition de la bissection sommet n'est pas, *mutatis mutandis*, strictement équivalente à celle de la bissection arête. En effet, la bissection sommet assure l'égalité stricte des ordres des deux sous-graphes résultants, ce qui implique l'ajout éventuel à l'ensemble déconnectant d'un sommet ayant pour fonction d'assurer cette égalité, alors que l'égalité à un près ne l'aurait pas nécessité (voir figure 4.4). Nous définissons donc la *bissection sommet à-un-près* $\mathrm{Bis}'_v(G)$ d'un graphe G comme le cardinal du plus petit ensemble de sommets déconnectant G en deux sous-graphes de même ordre, *à un près*.
Il est à noter que $\mathrm{Bis}_v(G)$ s'obtient directement à partir de $\mathrm{Bis}'_v(G)$, alors que la réciproque est fausse. En effet, on a

$$\mathrm{Bis}_v(G) = \begin{cases} \mathrm{Bis}'_v(G) & \text{si } n - \mathrm{Bis}'_v(G) \text{ est pair} \\ \mathrm{Bis}'_v(G) + 1 & \text{si } n - \mathrm{Bis}'_v(G) \text{ est impair} \ . \end{cases}$$

Chapitre 5

Résultats généraux et applications

Dans ce chapitre, nous donnons des majorations et minorations générales des largeurs de bande et de coupe. Le calcul des bornes supérieures est basé sur l'expression des numérotations du graphe considéré par rapport à celles de ses graphes quotients. Les bornes inférieures sont, elles, liées aux bissections sommet et arête du graphe.
La seconde partie du chapitre est consacrée à l'application de ces résultats à plusieurs familles de graphes de réseaux d'interconnexion.

5.1 Résultats sur la largeur de bande

Théorème 5.3. *Si G est un graphe, Q le graphe quotient obtenu à partir d'une partition Π de G, φ_Q une numérotation de Q appartenant à $\Phi_Q(G,\Pi)$ donnant la largeur de bande de Q, et φ_G une numérotation de $\Phi_G(G,\Pi,\varphi_Q)$, alors*

$$\mathrm{Bd}(G) \leq \max\left(\epsilon_i(\varphi_G), (\mathrm{Bd}(Q)-1)\max_\pi + \epsilon_e(\varphi_G)\right) \quad ,$$

$$o\grave{u} \begin{cases} \epsilon_i(\varphi_G) &= \max\limits_{\pi\in\Pi}\left(\mathrm{dil}(\varphi_{G|\pi})\right) \quad , \\ \epsilon_e(\varphi_G) &= \max\limits_{\substack{\{v',v''\}\in E_E(\Pi) \\ \varphi_G(v')<\varphi_G(v'')}} \left(\left(P_{\varphi_Q(\pi(v'))} - \varphi_G(v')\right) + \left(\varphi_G(v'') - p_{\varphi_Q(\pi(v''))}\right) + 1\right) \quad . \end{cases}$$

Preuve. Afin de faire apparaître des termes dépendant de Q dans l'expression de la largeur de bande de G, nous réécrivons celle-ci en différenciant les termes internes aux parties de la partition des termes externes à celles-ci.
Pour toutes les configurations $(\Pi,\varphi_Q,\varphi_G)$ de G respectant l'hypothèse, nous avons, en reprenant (3.1),

$$\begin{aligned} \mathrm{dil}(\varphi_G) &= \max_{\{v',v''\}\in E(G)} \left(|\varphi_G(v') - \varphi_G(v'')|\right) \\ &= \max\left(\max_{\{v',v''\}\in E_I(\Pi)} \left(|\varphi_G(v') - \varphi_G(v'')|\right), \max_{\{v',v''\}\in E_E(\Pi)} \left(|\varphi_G(v') - \varphi_G(v'')|\right)\right) \quad . \end{aligned}$$

Évaluons séparément chacun de ces deux termes.

- Comme φ_G est une numérotation induite par φ_Q, les numéros des sommets de chaque partie forment un intervalle, d'où

$$\begin{aligned}
\max_{\{v',v''\}\in E_I(\Pi)}(|\varphi_G(v')-\varphi_G(v'')|) &= \max_{\pi\in\Pi}\left(\max_{\{v',v''\}\in E_I(\pi)}(|\varphi_G(v')-\varphi_G(v'')|)\right)\\
&= \max_{\pi\in\Pi}\left(\max_{\{v',v''\}\in E_I(\pi)}(|\varphi_{G|\pi}(v')-\varphi_{G|\pi}(v'')|)\right)\\
&= \max_{\pi\in\Pi}\left(\mathrm{dil}(\varphi_{G|\pi})\right)\\
&= \epsilon_i(\varphi_G)\ .
\end{aligned}$$

- Quand E_E n'est pas vide, $\mathrm{Bd}(Q)\geq 1$, et pour toutes les arêtes $\{v',v''\}$ appartenant à E_E, avec $q'=\varphi_Q(\pi(v'))<\varphi_Q(\pi(v''))=q''$, on a $q''-q'\leq \mathrm{Bd}(Q)$ et

$$\begin{aligned}
\varphi_G(v'')-\varphi_G(v') &= (\varphi_G(v'')-p_{q''}+1)+(P_{q''-1}-p_{q'+1}+1)+(P_{q'}-\varphi_G(v'))\\
&= \sum_{q'<q<q''}|\varphi_Q^{-1}(q)|+(P_{q'}-\varphi_G(v'))+(\varphi_G(v'')-p_{q''})+1\\
&\leq (\mathrm{Bd}(Q)-1)\max_\pi+(P_{q'}-\varphi_G(v'))+(\varphi_G(v'')-p_{q''})+1\ ,
\end{aligned}$$

  d'où $\max_{\{v',v''\}\in E_E(\Pi)}(|\varphi_G(v')-\varphi_G(v'')|)\leq(\mathrm{Bd}(Q)-1)\max_\pi+\epsilon_e(\varphi_G)$.

En combinant ces deux résultats avec l'équation (4.1), on obtient la borne annoncée. □

Corollaire 5.4. *Soit G un graphe d'ordre n, et Q le graphe quotient obtenu à partir d'une partition* Π *de $V(G)$.*

$$\mathrm{Bd}(G)\leq(\mathrm{Bd}(Q)+1)\max_\pi-1\ .$$

Preuve. Soit φ_Q une numérotation de Q donnant la largeur de bande de Q. Pour toute numérotation φ_G de G induite par Π et φ_Q, on a $\epsilon_i(\varphi_G)\leq\max_\pi-1$ et $\epsilon_e(\varphi_G)\leq 2(\max_\pi-1)+1$. En appliquant ces majorations à l'énoncé du théorème 5.3, on obtient le résultat souhaité. □

Théorème 5.5. *Pour tout graphe G,*

$$\mathrm{Bd}(G)\geq \mathrm{Bis}'_v(G)\ .$$

Preuve. Soit G un graphe d'ordre n, et φ une numérotation de G donnant la largeur de bande de G, c'est-à-dire telle que $\mathrm{dil}(\varphi)=\mathrm{Bd}(G)$. Posons

$$b_1=\left\lfloor\frac{n-\mathrm{Bd}(G)}{2}\right\rfloor \qquad\text{et}\qquad b_2=\left\lfloor\frac{n+\mathrm{Bd}(G)}{2}\right\rfloor\ ,$$

et soit $V_d=\{v\in V(G)\ /\ b_1\leq\varphi(v)<b_2\}$.
Il ne peut exister dans $E(G)$ d'arête $\{v',v''\}$ telle que $\varphi(v')<b_1$ et $\varphi(v'')\geq b_2$, puisque $|\varphi(v')-\varphi(v'')|\leq\mathrm{Bd}(G)=b_2-b_1$. De ce fait, V_d déconnecte G en deux sous-graphes G_1 et G_2 tels que $|V(G_1)|=|V(G_2)|$ ou bien $|V(G_1)|=|V(G_2)|-1$, et donc $\mathrm{Bis}'_v(G)\leq|V_d|=\mathrm{Bd}(G)$. □

5.2 Résultats sur la largeur de coupe

Théorème 5.6. *Si G est un graphe, Q le graphe quotient obtenu à partir d'une partition Π de $V(G)$, et φ_Q une numérotation de Q appartenant à $\Phi_Q(G,\Pi)$ donnant la largeur de coupe de Q, alors*

$$\mathrm{Cw}(G) \leq \left(\mathrm{Cw}(Q) - \left\lceil \tfrac{\delta(Q)}{2} \right\rceil\right) \cdot \max_\omega + \max_{\pi\in\Pi} \left(\mathrm{Cw}(G[\pi]) + |\omega(\pi)| - |\pi| \cdot \min\left(\delta^-_{\Pi,\varphi_Q}(\pi), \delta^+_{\Pi,\varphi_Q}(\pi)\right)\right) \quad .$$

Preuve. D'après l'équation (4.2), nous savons que, pour tout φ_G appartenant à $\Phi_G(G,\Pi,\varphi_Q)$, $\mathrm{Cw}(G) \leq \mathrm{cg}(\varphi_G)$. Pour faire apparaître des termes dépendant de Q dans l'expression de $\mathrm{cg}(\varphi_G)$, nous décomposons celle-ci en trois termes prenant en compte les arêtes de G respectivement externes à une partie donnée, appartenant au cocycle de cette partie, et internes à la partie, comme illustré en figure 5.1. Pour toute numérotation φ_G appartenant à $\Phi_G(G,\Pi,\varphi_Q)$, on a, en reprenant (3.2),

$$\begin{aligned}\mathrm{Cw}(G) \leq \mathrm{cg}(\varphi_G) &= \max_{v\in V(G)} (|\{\{v',v''\} \in E(G) \,/\, \varphi_G(v') \leq \varphi_G(v) < \varphi_G(v'')\}|) \\ &= \max_{\pi\in\Pi} \Big(\max_{v\in\pi}(|\{\{v',v''\} \in E_E(\pi) \cup E_C(\pi) \cup E_I(\pi) \,/\, \\ &\qquad\qquad \varphi_G(v') \leq \varphi_G(v) < \varphi_G(v'')\}|)\Big) \quad .\end{aligned}$$

Posons

$$\begin{aligned} f_E(\pi) &= \max_{v\in\pi} (|\{\{v',v''\} \in E_E(\pi) \,/\, \varphi_G(v') \leq \varphi_G(v) < \varphi_G(v'')\}|) \quad , \\ f_C(\pi) &= \max_{v\in\pi} (|\{\{v',v''\} \in E_C(\pi) \,/\, \varphi_G(v') \leq \varphi_G(v) < \varphi_G(v'')\}|) \quad , \text{ et} \\ f_I(\pi) &= \max_{v\in\pi} (|\{\{v',v''\} \in E_I(\pi) \,/\, \varphi_G(v') \leq \varphi_G(v) < \varphi_G(v'')\}|) \quad . \end{aligned}$$

Ainsi, $\mathrm{cg}(\varphi_G) \leq \max_{\pi\in\Pi}(f_E(\pi) + f_C(\pi) + f_I(\pi))$.
Étudions séparément chacun de ces trois termes. Soit π un élément de Π, et v un sommet de π.

- $f_E(\pi)$ dénombrant des arêtes externes à π, le choix de v dans π n'est en fait pas significatif. Considérons alors le graphe quotient Q. Soient π^- et π^+ les sommets de Q tels que $\varphi_Q(\pi^-) = \varphi_Q(\pi) - 1$ et $\varphi_Q(\pi^+) = \varphi_Q(\pi) + 1$, s'ils existent. Le nombre d'arêtes de Q dont les numéros des extrémités encadrent strictement $\varphi_Q(\pi)$, que nous notons $g_E(\pi)$, peut s'écrire de deux manières :

$$\begin{aligned} g_E(\pi) &= |\{\{w',w''\} \in E(Q) \,/\, \varphi_Q(w') \leq \varphi_Q(\pi^-) < \varphi_Q(w'')\}| - \delta^-_{\varphi_Q}(\pi) \quad , \text{ et} \\ g_E(\pi) &= |\{\{w',w''\} \in E(Q) \,/\, \varphi_Q(w') \leq \varphi_Q(\pi) < \varphi_Q(w'')\}| - \delta^+_{\varphi_Q}(\pi) \quad . \end{aligned}$$

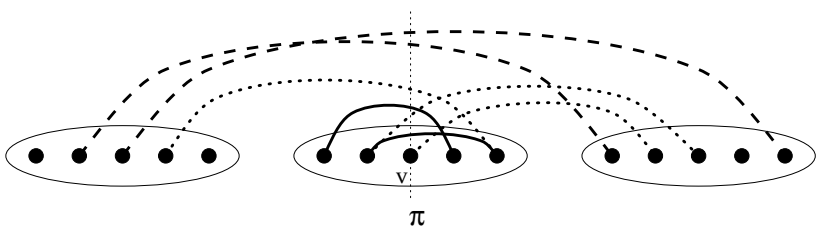


Figure 5.1: Contribution à la congestion des arêtes externes (en tirets), du cocycle (en pointillés), et internes (en traits pleins) à la partie π

Comme le premier terme du membre droit de chacune des égalités est, par définition, majoré par $\mathrm{cg}(\varphi_Q)$, on a $g_E(\pi) \leq \mathrm{cg}(\varphi_Q) - \min\left(\delta^-_{\varphi_Q}(\pi), \delta^+_{\varphi_Q}(\pi)\right)$, et puisque $\delta^-_{\varphi_Q}(\pi) + \delta^+_{\varphi_Q}(\pi) = \delta(\pi)$, le minimum des degrés partiels est toujours supérieur ou égal à $\left\lceil \frac{\delta(\pi)}{2} \right\rceil$, d'où

$$g_E(\pi) \leq \mathrm{cg}(\varphi_Q) - \left\lceil \frac{\delta(\pi)}{2} \right\rceil \ .$$

Comme une arête de Q quotiente au plus $\max_\omega$ arêtes de G,

$$f_E(\pi) \leq \max_\omega \cdot \left(\mathrm{cg}(\varphi_Q) - \left\lceil \frac{\delta(\pi)}{2} \right\rceil\right) \ .$$

- Le deuxième terme est trivialement borné par $|E_C(\pi)| = |\omega(\pi)|$. Remarquons cependant que, de tout sommet v' de π tel que $\varphi_G(v') \geq \varphi_G(v)$, au moins $\delta^+_{\Pi,\varphi_Q}(\pi)$ arêtes de $E_C(\pi)$ sont incidentes à v'' et telles que $\varphi_G(v'') > \varphi_G(v')$; elles ne doivent donc pas être prises en compte. De même, pour tout sommet v' de π tel que $\varphi_G(v') < \varphi_G(v)$, au moins $\delta^-_{\Pi,\varphi_Q}(\pi)$ arêtes ne doivent pas être comptées. Ainsi,

$$|\{\{v', v''\} \in E_C(\pi) \,/\, \varphi_G(v') \leq \varphi_G(v) < \varphi_G(v'')\}| \leq |\omega(\pi)| - |\pi| \cdot \min\left(\delta^-_{\Pi,\varphi_Q}(\pi), \delta^+_{\Pi,\varphi_Q}(\pi)\right) \ .$$

- Le troisième terme est, par définition, égal à la congestion de $G[\pi]$ numéroté par la restriction de φ_G à π, c'est-à-dire $\mathrm{cg}(\varphi_{G|\pi})$.

En combinant ces trois bornes, nous arrivons à

$$\begin{aligned}\mathrm{Cw}(G) \leq \max_{\pi\in\Pi} \Bigg(&\left(\mathrm{cg}(\varphi_Q) - \left\lceil \tfrac{\delta(\pi)}{2} \right\rceil\right) \cdot \max_\omega + \mathrm{cg}(\varphi_{G|\pi}) + \\ &\left(|\omega(\pi)| - |\pi| \cdot \min\left(\delta^-_{\Pi,\varphi_Q}(\pi), \delta^+_{\Pi,\varphi_Q}(\pi)\right)\right)\Bigg) \ .\end{aligned}$$

Puisque, dans chaque partie, la numérotation φ_G des sommets de G est continue et indépendante des numérotations dans les autres parties, il est possible, une fois donné un φ_Q, d'obtenir une numérotation φ_G induite telle que la restriction de φ_G à chacune des parties π de Π donne la largeur de coupe de $G[\pi]$. Nous concluons la preuve en combinant cette numérotation avec l'inégalité ci-dessus. □

Théorème 5.7. *Pour tout graphe G,*

$$\mathrm{Cw}(G) \geq \mathrm{Bis}_e(G) \ .$$

Preuve. Soit G un graphe d'ordre n, et φ une numérotation de G donnant la largeur de coupe de G, c'est-à-dire telle que $\mathrm{cg}(\varphi) = \mathrm{Cw}(G)$, et soit $E_d = \left\{\{v', v''\} \in E(G) \,/\, \varphi_G(v') \leq \left\lfloor \frac{n-1}{2} \right\rfloor < \varphi_G(v'')\right\}$. Par définition de la largeur de coupe, $|E_d| \leq \mathrm{Cw}(G)$, et E_d déconnecte G en deux sous-graphes G_1 et G_2 tels que $|V(G_1)| = \left\lfloor \frac{n-1}{2} \right\rfloor$ et $|V(G_2)| = \left\lceil \frac{n-1}{2} \right\rceil$. On a donc $\mathrm{Bis}_e(G) \leq |E_d| \leq \mathrm{Cw}(G)$. □

5.3 Applications

Nous nous consacrons ici à l'application des résultats de la section précédente à différentes familles de graphes de réseaux d'interconnexion. Ces graphes, dont les largeurs de bande et de coupe ont été étudiées principalement dans le cadre de l'implantation *V.L.S.I.* [20, 47, 70, 69], possèdent des propriétés de structure dont nous tirons parti pour construire leurs partitions.

5.3.1 La grille bidimensionnelle

La largeur de bande de la grille bidimensionnelle a été calculée par Chvàtalovà.

Théorème 5.8. [20]

$$\mathrm{Bd}(\mathrm{M}_2(d_1,d_2)) = d1 \ .$$

Proposition 5.9.

$$\begin{array}{rcccl} d_1 & \leq & \mathrm{Cw}(\mathrm{M}_2(d_1,d_2)) & \leq & d_1+1 \quad si\ d_2\ est\ pair\ , \\ & & \mathrm{Cw}(\mathrm{M}_2(d_1,d_2)) & = & d_1+1 \quad si\ d_2\ est\ impair\ . \end{array}$$

Preuve. Pour majorer la largeur de coupe, une intuition immédiate est de partitionner la grille en regroupant les sommets appartenant à une même dimension. Nous définissons donc la configuration $(\Pi, \varphi_Q, \varphi_G)$ suivante :

- $\Pi = \{\pi_0, \pi_1, \ldots, \pi_{d_2-1}\}$, où $\pi_q = \{(v_1,v_2) \in V(\mathrm{M}_2(d_1,d_2)) \ / \ v_2 = q\}$;
- $\varphi_Q(\pi_q) = q$;
- $\varphi_G(v_1,v_2) = \varphi_Q(\pi_{v_2}) \cdot d_1 + v_1$.

Pour tout q appartenant à $\{0,1,\ldots,d_2-1\}$, $|\pi_q| = d_1$, et $\mathrm{M}_2(d_1,d_2)[\pi_q]$ est isomorphe à la chaîne $\mathrm{P}(d_1)$, donc $\mathrm{Cw}(\mathrm{M}_2(d_1,d_2)[\pi_q]) = 1$.
$\mathrm{M}_2(d_1,d_2)_{/\Pi}$ est isomorphe à $\mathrm{P}(d_2)$, d'où $\mathrm{Cw}(\mathrm{M}_2(d_1,d_2)_{/\Pi}) = 1$. De plus, tout sommet v d'une partie π_q est relié à un sommet au plus de π_{q-1} quand $q > 0$, et à un sommet au plus de π_{q+1} quand $q < (d_2-1)$. De fait,

$$\begin{array}{lccc} & |\omega(\pi_0)| = d_1, & \delta^-_{\Pi,\varphi_Q}(\pi_0) = 0, & \delta^+_{\Pi,\varphi_Q}(\pi_0) = 1 \ ; \\ \text{pour } 0 < q < d_2-1, & |\omega(\pi_q)| = 2d_1, & \delta^-_{\Pi,\varphi_Q}(\pi_q) = 1, & \delta^+_{\Pi,\varphi_Q}(\pi_q) = 1 \ ; \\ & |\omega(\pi_{d_2-1})| = d_1, & \delta^-_{\Pi,\varphi_Q}(\pi_{d_2-1}) = 1, & \delta^+_{\Pi,\varphi_Q}(\pi_{d_2-1}) = 0 \ . \end{array}$$

Il en découle que, pour tout π_q, $|\omega(\pi_q)| - |\pi_q| \cdot \min(\delta^-_{\Pi,\varphi_Q}(\pi_q), \delta^+_{\Pi,\varphi_Q}(\pi_q)) = d_1$, et donc, par le théorème 5.6, que $\mathrm{Cw}(\mathrm{M}_2(d_1,d_2)) \leq d_1+1$.

Pour la minoration, on peut montrer, d'après [57, pages 223–226], que $\mathrm{Bis}_e(\mathrm{M}_2(d_1,d_2)) = d_1$ si d_2 est pair, et d_1+1 si d_2 est impair. Le théorème 5.7 nous permet alors d'obtenir le résultat annoncé. □

5.3.2 L'hypercube

Une expression de la largeur de bande de l'hypercube a été donnée par Harper [47]. Lai et Sprague [55] ont ultérieurement calculé son équivalent asymptotique.

Théorème 5.10. [47, 55]

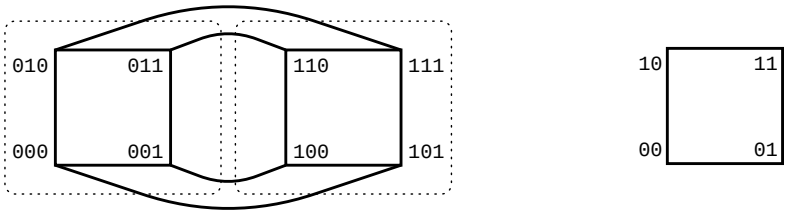
$$\begin{array}{rcl} \mathrm{Bd}(\mathrm{H}(k)) & = & \displaystyle\sum_{j=0}^{k-1} \binom{j}{\lceil \frac{j}{2} \rceil} \ , \\ \mathrm{Bd}(\mathrm{H}(k)) & \in & \Theta\left(\frac{2^k}{\sqrt{\pi k}}\right) \ . \end{array}$$

Proposition 5.11.

$$2^{k-1} \leq \mathrm{Cw}(\mathrm{H}(k)) \leq 2^k - 1 \ .$$

Preuve. La preuve de la borne supérieure du théorème est basée sur la construction récursive des hypercubes. Nous définissons une partition dont chaque partie contient un sous-hypercube, comme illustré en figure 5.2 :

- $\Pi = \{\pi_0, \pi_1\}$, où $\begin{cases} \pi_0 = \{v \in V(\mathrm{H}(k)) \,/\, v = 0v_2 \dots v_k\} \ , \\ \pi_1 = \{v \in V(\mathrm{H}(k)) \,/\, v = 1v_2 \dots v_k\} \ ; \end{cases}$
- $\varphi_Q(\pi_q) = q$;
- $\varphi_G(v) = [v]$.



a. H(3) et sa partition : deux H(2). **b.** H(2) est isomorphe à $C(4)$.

Figure 5.2: Structure récursive des hypercubes.

Par définition, $\mathrm{H}(k)[\pi_0]$ et $\mathrm{H}(k)[\pi_1]$ sont isomorphes à $\mathrm{H}(k-1)$ et $\mathrm{H}(k)_{/\Pi}$ est isomorphe à P(2), d'où $\mathrm{Cw}(\mathrm{H}(k)_{/\Pi}) = 1$, $\delta(\mathrm{H}(k)_{/\Pi}) = 1$, et

$$\begin{array}{lll} |\pi_0| = 2^{k-1}, & |\omega(\pi_0)| = 2^{k-1}, & \mathrm{Cw}(G[\pi_0]) = \mathrm{Cw}(\mathrm{H}(k-1)), \\ |\pi_1| = 2^{k-1}, & |\omega(\pi_1)| = 2^{k-1}, & \mathrm{Cw}(G[\pi_1]) = \mathrm{Cw}(\mathrm{H}(k-1)), \\ & \delta^-_{\Pi,\varphi_Q}(\pi_0) = 0, & \delta^+_{\Pi,\varphi_Q}(\pi_0) = 1, \\ & \delta^-_{\Pi,\varphi_Q}(\pi_1) = 1, & \delta^+_{\Pi,\varphi_Q}(\pi_1) = 0. \end{array}$$

Par le théorème 5.6, nous obtenons

$$\begin{aligned} \mathrm{Cw}(\mathrm{H}(k)) &\leq \max_{\pi \in \{\pi_0, \pi_1\}} \left(\mathrm{Cw}(\mathrm{H}(k)[\pi]) + |\omega(\pi)| - |\pi| \cdot \min\left(\delta^-_{\Pi,\varphi_Q}(\pi), \delta^+_{\Pi,\varphi_Q}(\pi) \right) \right) \\ &\leq \mathrm{Cw}(\mathrm{H}(k-1)) + 2^{k-1} \ . \end{aligned}$$

Comme H(2) est isomorphe au cycle $C(4)$, $\mathrm{Cw}(\mathrm{H}(2)) = 2$, et

$$\mathrm{Cw}(\mathrm{H}(k)) \leq 2^{k-1} + 2^{k-2} + \cdots + 4 + 2 = 2^k - 2 \ .$$

Il a été prouvé dans [21, page 11] que $\mathrm{Bis}_\varepsilon(\mathrm{H}(k)) \geq 2^{k-1}$. Puisqu'un couplage parfait donne trivialement $\mathrm{Bis}_\varepsilon(\mathrm{H}(k)) \leq 2^{k-1}$, alors $\mathrm{Bis}_\varepsilon(\mathrm{H}(k)) = 2^{k-1}$. Appliquer le théorème 5.7 permet de conclure. □

La valeur exacte de $\mathrm{Cw}(\mathrm{H}(k))$ a été trouvée indépendamment par Nakano et Bel Hala.

Théorème 5.12. [8, 70]

$$\mathrm{Cw}(\mathrm{H}(k)) = \left\lfloor \frac{1}{3} 2^{k+1} \right\rfloor \ .$$

5.3.3 Le graphe de De Bruijn binaire

Nous allons exclusivement étudier ici le graph de De Bruijn non orienté binaire, c'est-à-dire $\mathrm{UB}(2,k)$.

Afin d'obtenir un graphe quotient qui soit une chaîne, nous quotientons $\mathrm{UB}(2,k)$ selon le poids de Hamming de ses sommets, puisque la différence entre les poids de Hamming de deux sommets liés par une arête est d'au plus 1. Nous allons donc utiliser la configuration suivante, illustrée en figure 5.3 :

- $\Pi = \{\pi_0, \pi_1, \ldots, \pi_k\}$, où $\pi_q = \{v \in V(\mathrm{UB}(2,k)) \,/\, \mathcal{H}(v) = q\}$;
- $\varphi_Q(\pi_q) = q$;
- φ_G, définie comme :
  - $\varphi_G(0^k) = 0$,
  - $\forall v', v'' \in V(\mathrm{UB}(2,k)), (\varphi_G(v') > \varphi_G(v'')) \Longleftrightarrow ((\varphi_Q(\pi(v')) > \varphi_Q(\pi(v''))) \text{ ou } (\pi(v') = \pi(v'') \text{ et } [v'] < [v'']))$.

φ_G, qui est induite par φ_Q, revient, dans chaque partie, à numéroter les sommets v par ordre décroissant par rapport à $[v]$; elle est équivalente à la numérotation définie dans [47] pour l'hypercube.

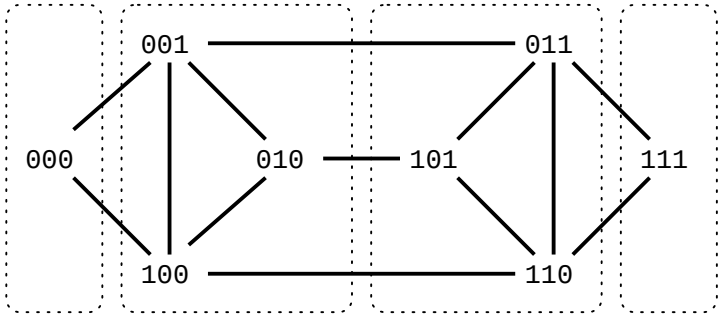


Figure 5.3: $\mathrm{UB}(2,3)$ et sa partition.

Les propriétés de cette configuration sont énoncées dans le lemme suivant.

Lemme 5.13.

$$
\begin{aligned}
&(1) \qquad \mathrm{UB}(2,k)_{/\Pi} \textit{ est la chaîne } \mathrm{P}(k+1) \ . \\
&(2) \qquad |\pi_q| = \binom{k}{q} \leq \binom{k}{\lceil \frac{k}{2} \rceil} \ . \\
&(3) \qquad (v \in \pi_q) \Longleftrightarrow (\overline{v} \in \pi_{k-q}) \ . \\
&(4) \qquad \varphi_G(\overline{v}) = 2^k - 1 - \varphi_G(v) \ .
\end{aligned} \tag{5.1}
$$

Preuve. (1) Pour toute $\{v', v''\} \in E(\mathrm{UB}(2,k))$, avec $v' \in \pi_{q'}$ et $v'' \in \pi_{q''}$, on a $\mathcal{H}(v') - 1 \leq \mathcal{H}(v'') \leq \mathcal{H}(v') + 1$, puisque v'' est obtenue à partir de v' par le remplacement d'une unique lettre. De fait, $q' - 1 \leq q'' \leq q' + 1$.
Pour tout q tel que $0 \leq q < k$, $\{1^q 0^{k-q}, 1^{q+1}0^{k-q-1}\}$ est une arête de $E(\mathrm{UB}(2,k))$ reliant π_q à π_{q+1}, et $\{\pi_q, \pi_{q+1}\}$ est une arête de $\mathrm{UB}(2,k)_{/\Pi}$.
Par ce qui précède, $\mathrm{UB}(2,k)_{/\Pi}$ est isomorphe à $\mathrm{P}(k+1)$.
(2) Le nombre de sommets de poids q est égal au nombre de manières de placer q lettres "1" parmi k positions, donc $|\pi_q| = \binom{k}{q}$, et est maximal pour $q = \lceil \frac{k}{2} \rceil$.
(3) Immédiatement, $(v \in \pi_q) \Longleftrightarrow (\mathcal{H}(v) = q) \Longleftrightarrow \mathcal{H}(\overline{v}) = k - q \Longleftrightarrow (\overline{v} \in \pi_{k-q})$.
(4) Pour tous $v', v'' \in V(\mathrm{UB}(2,k))$, on a, par définition de φ_G, $(\varphi_G(v') < \varphi_G(v'')) \Longleftrightarrow (\varphi_G(\overline{v'}) > \varphi_G(\overline{v''}))$, donc

$$\begin{aligned} \varphi_G(v) &= |\{v' \in V(\mathrm{UB}(2,k)) \,/\, \varphi_G(v') < \varphi_G(v)\}| \\ &= |\{v' \in V(\mathrm{UB}(2,k)) \,/\, \varphi_G(v') > \varphi_G(\overline{v})\}| \quad = 2^k - 1 - \varphi_G(\overline{v}) \ . \end{aligned}$$

□

Proposition 5.14.

$$\frac{2^k - 1}{k} \leq \mathrm{Bd}(\mathrm{UB}(2,k)) \leq \binom{k}{\lceil \frac{k}{2} \rceil} \ .$$

Preuve. Démontrons d'abord la borne supérieure. Puisque, d'après le lemme 5.13, $\mathrm{UB}(2,k)_{/\Pi}$ est isomorphe à $\mathrm{P}(k+1)$, $\mathrm{Bd}(\mathrm{UB}(2,k)_{/\Pi}) = 1$, et $\mathrm{Bd}(\mathrm{UB}(2,k)) = \max(\epsilon_i(\varphi_G), \epsilon_e(\varphi_G))$. Du fait que le graphe quotient est une chaîne, pour toute arête $\{v', v''\}$ de $E_E(\Pi)$ telle que $\varphi_G(v') < \varphi_G(v'')$, on a $P_{\varphi_Q(\pi(v'))} = p_{\varphi_Q(\pi(v''))} - 1$, et donc $\epsilon_e(\varphi_G) = \max\limits_{\{v',v''\} \in E_E(\Pi)} (|\varphi_G(v'') - \varphi_G(v')|)$.
Considérons toutes les arêtes $\{v', v''\}$ de $E(\mathrm{UB}(2,k))$, en supposant sans perte de généralité que $q' = \varphi_Q(\pi(v')) \leq \varphi_Q(\pi(v'')) = q''$.

- Si $q'' = q'$, alors $|\varphi_G(v') - \varphi_G(v'')| \leq |\pi_{q'}| - 1 \leq \binom{k}{\lceil \frac{k}{2} \rceil} - 1$, et donc $\epsilon_i(\varphi_G) = \binom{k}{\lceil \frac{k}{2} \rceil} - 1$.
- Si $q'' = q' + 1$, alors v'' est obtenu à partir de v' par suppression d'une lettre "0" et ajout d'une lettre "1".
  - Si $v' = 0m$ et $v'' = m1$, considérons le sommet $v'_\perp = 0^{k-q}1^q = 0m_\perp$ de $\pi_{q'}$ et son voisin $v''_\perp = m_\perp 1$ de $\pi_{q''}$. Par définition, pour tout m, $[m_\perp] \leq [m]$, et, arithmétiquement, $([0m] - [0m_\perp]) \leq ([m1] - [m_\perp 1])$. Comme $(\varphi_G(v'_\perp) - \varphi_G(v')) = ([0m] - [0m_\perp])$ et $(\varphi_G(v''_\perp) - \varphi_G(v'')) = ([m1] - [m_\perp 1])$, on a donc $(\varphi_G(v'') - \varphi_G(v')) \leq (\varphi_G(v''_\perp) - \varphi_G(v'_\perp)) = |\pi_{q''}| \leq \binom{k}{\lceil \frac{k}{2} \rceil}$.
  - Si $v' = m0$ et $v'' = 1m$, alors $\overline{v'} = \overline{m}1$ et $\overline{v''} = 0\overline{m}$, ce qui revient au cas précédent car, par (5.1), $|\varphi_G(v'') - \varphi_G(v')| = |\varphi_G(\overline{v''}) - \varphi_G(\overline{v'})|$.

  Par ce qui précède, $\epsilon_e(\varphi_G) = \binom{k}{\lceil \frac{k}{2} \rceil}$.

On obtient alors la majoration en calculant le maximum de $\epsilon_i(\varphi_G)$ et de $\epsilon_e(\varphi_G)$.

Pour démontrer la borne inférieure, on utilise le théorème 3.2, sachant que $|V(\mathrm{UB}(2,k))| = 2^k$ et $\mathrm{diam}(\mathrm{UB}(2,k)) = k$. Ceci nous permet de conclure. □

Proposition 5.15.

$$\frac{1}{2}.\frac{2^k}{k} \leq \mathrm{Cw}(\mathrm{UB}(2,k)) \leq 2\binom{k}{\lceil \frac{k}{2} \rceil} + 2 \ .$$

Preuve. Nous utilisons pour démontrer ce théorème une configuration basée sur les mêmes Π et φ_Q que précédemment, le φ_G induit correspondant étant défini plus bas. Puisque $\mathrm{UB}(2,k)_{/\Pi}$ est isomorphe à $\mathrm{P}(k+1)$, $\mathrm{Cw}(\mathrm{UB}(2,k)_{/\Pi}) = 1$ et $\delta(\mathrm{UB}(2,k)_{/\Pi}) = 1$. Comme de chaque sommet d'une partie π partent au plus deux arêtes vers d'autres parties, $|\omega(\pi)| \leq 2|\pi|$ pour toute π.
Les arêtes $\{v', v''\}$ internes à chaque partie π lient des sommets de même poids, donc si $v' = v_1 v_2 \ldots v_{k-1} v_k$, alors $v'' = v_2 \ldots v_{k-1} v_k v_1$ ou $v'' = v_k v_1 v_2 \ldots v_{k-1}$. De fait, les composantes connexes de $\mathrm{UB}(2,k)[\pi]$ sont des sommets isolés, des arêtes simples, ou des cycles (ces cycles éventuellement dégénérés sont appelés cycles d'Etzion [26]), et donc, pour toute π, $\mathrm{Cw}(\mathrm{UB}(2,k)[\pi]) \leq 2$.
Pour $k \geq 3$ et pour toute partie π_q, il est possible de trouver des sommets soit n'ayant pas de voisins dans π_{q-1} (de la forme $0v_2 \ldots v_{k-1}0$), soit n'ayant pas de voisins dans π_{q+1} (de la forme $1v_2 \ldots v_{k-1}1$). Il en découle que, pour toute π, $\min\left(\delta^-_{\Pi,\varphi_Q}(\pi), \delta^+_{\Pi,\varphi_Q}(\pi)\right) = 0$ et, par le théorème 5.6,

$$\begin{aligned}
\mathrm{Cw}(\mathrm{UB}(2,k)) &\leq \left(\mathrm{Cw}\left(\mathrm{UB}(2,k)_{/\Pi}\right) - \left\lceil \frac{\delta(\mathrm{UB}(2,k)_{/\Pi})}{2} \right\rceil\right) \cdot \max_{\pi\in\Pi}(|\omega(\pi)|) + \\
&\quad \max_{\pi\in\Pi}\left(\mathrm{Cw}(\mathrm{UB}(2,k)[\pi]) + |\omega(\pi)| - |\pi| \cdot \min\left(\delta^-_{\Pi,\varphi_Q}(\pi), \delta^+_{\Pi,\varphi_Q}(\pi)\right)\right) \\
&\leq \max_{\pi\in\Pi}(\mathrm{Cw}(\mathrm{UB}(2,k)[\pi]) + |\omega(\pi)|) \\
&\leq 2 + 2\binom{k}{\lceil \frac{k}{2} \rceil} \ .
\end{aligned}$$

Il est facile de voir que $\mathrm{Cw}(\mathrm{UB}(2,1)) = 1$ et $\mathrm{Cw}(\mathrm{UB}(2,2)) = 3$, ce qui valide la majoration pour tout k.

Pour ce qui est de la borne inférieure, on sait d'après [57, page 480] que $\mathrm{Bis}_e(\mathrm{SE}(k)) = \frac{2^k-1}{k}$. Comme $\mathrm{SE}(k)$ est un sous-graphe partiel de $\mathrm{UB}(2,k)$, $\mathrm{Bis}_e(\mathrm{SE}(k)) \leq \mathrm{Bis}_e(\mathrm{UB}(2,k))$, et donc par le théorème 5.7

$$\begin{aligned}
\mathrm{Cw}(\mathrm{UB}(2,k)) &\geq \mathrm{Bis}_e(\mathrm{UB}(2,k)) \\
&\geq \tfrac{1}{2} \cdot \tfrac{2^k}{k} \ .
\end{aligned}$$

□

Remarque. Il a été démontré [29] que $\binom{k}{\lceil \frac{k}{2} \rceil}$ appartient à $\Theta\left(2^k \sqrt{\frac{2}{k\pi}}\right)$; on peut également montrer que, pour tout k, $\binom{k}{\lceil \frac{k}{2} \rceil} \leq 2^k \sqrt{\frac{2}{k\pi}}$.
Par conséquent, les majorations des largeurs de bande et de coupe du graphe de De Bruijn non-orienté binaire sont toutes deux en $\Theta\left(2^k \sqrt{\frac{2}{k\pi}}\right)$.

5.3.4 Le graphe de De Bruijn d-aire

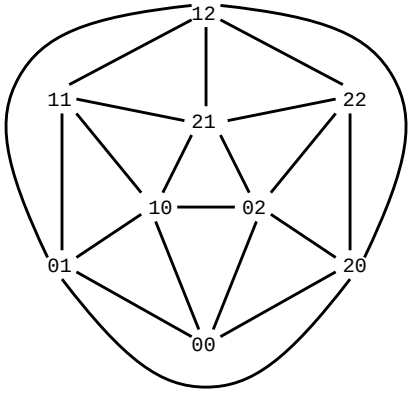
Nous étudions ici le graphe de De Bruijn non orienté d-aire, $\mathrm{UB}(d,k)$, pour $d > 2$.

Afin d'obtenir un graphe quotient qui soit une chaîne, nous quotientons $\mathrm{UB}(d,k)$ selon le poids de Hamming de ses sommets en regroupant dans chaque partie les sommets correspondant à $(d-1)$ poids de Hamming consécutifs, puisque la différence entre les poids de deux sommets liés par une

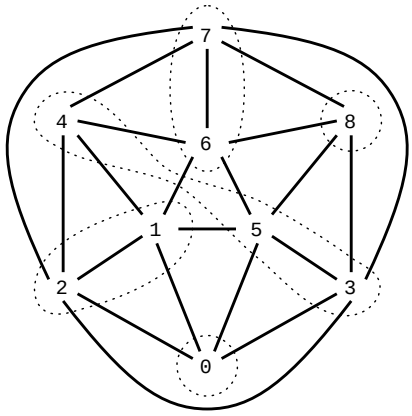
arête est d'au plus $(d-1)$. Le "cadrage" des parties par rapport à la bande de poids 0 se fait au moyen de la variable entière γ que nous définissons ci-après. Soit γ un entier tel que $0 \leq \gamma \leq (d-2)$, et soit la configuration :

- $\Pi = \{\pi_0, \pi_1, \ldots, \pi_k\}$, où $\pi_q = \{v \in V(\mathrm{UB}(d,k)) / q(d-1)+\gamma-(d-2) \leq \mathcal{H}(v) \leq q(d-1)+\gamma\}$;
- $\varphi_Q(\pi_q) = q$;
- φ_G, définie comme :
  - $\varphi_G(0^k) = 0$,
  - $\forall v', v'' \in V(\mathrm{UB}(d,k)), (\varphi_G(v') > \varphi_G(v'')) \Longleftrightarrow ((\varphi_Q(\pi(v')) > \varphi_Q(\pi(v''))) \text{ ou } (\pi(v') = \pi(v'') \text{ et } [v'] < [v'']))$.

φ_G, qui est induite par φ_Q, revient, dans chaque partie, à numéroter les sommets v par ordre décroissant par rapport à $[v]$. Un exemple de cette configuration est donné en figure 5.4. Par extension, on notera $\mathcal{H}(\pi)$ le plus grand poids de Hamming contenu dans une partie, c'est-à-dire $\mathcal{H}(\pi_q) = q(d-1)+\gamma$.



a. Le graphe UB(3, 2).



b. Numérotation φ_G de UB(3, 2). Les ensembles en pointillés regroupent les sommets de même poids.

Figure 5.4: Le graphe UB(3, 2).

Les propriétés de cette configuration sont énoncées dans le lemme suivant.

Lemme 5.16.

$$\begin{aligned} &(1) \qquad \mathrm{UB}(d,k)_{/\Pi} \textit{ est la chaîne } \mathrm{P}(k+1) \ . \\ &(2) \qquad |\pi| = \left| A_{d,k}^{d-1}\left(\mathcal{H}(\pi)\right) \right| \ . \\ &(3) \qquad \varphi_G(\overline{v}) = d^k - 1 - \varphi_G(v) \ . \end{aligned} \tag{5.2}$$

Preuve. (1) Cette démonstration est identique à celle réalisée pour le cas binaire dans le lemme 5.13.
(2) Ceci découle immédiatement de la définition de $A_{d,k}$.

(3) Pour tous $v', v'' \in V(\mathrm{UB}(d,k))$, on a, par définition de φ_G, $(\varphi_G(v') < \varphi_G(v'')) \Longleftrightarrow (\varphi_G(\overline{v'}) > \varphi_G(\overline{v''}))$, donc

$$\begin{aligned}\varphi_G(v) &= |\{v' \in V(\mathrm{UB}(d,k)) \,/\, \varphi_G(v') < \varphi_G(v)\}| \\ &= |\{v' \in V(\mathrm{UB}(d,k)) \,/\, \varphi_G(v') > \varphi_G(\overline{v})\}| \quad = d^k - 1 - \varphi_G(\overline{v}) \ .\end{aligned}$$

□

Proposition 5.17.

$$\frac{d^k-1}{d} \leq \mathrm{Bd}(\mathrm{UB}(d,k)) \leq \left| A_{d,k}^{d-1}\left(h_{d,k}^{d-1}\right)\right| \ .$$

Preuve. Démontrons d'abord la majoration. Puisque, d'après le lemme 5.16, $\mathrm{UB}(d,k)_{/\Pi}$ est isomorphe à $\mathrm{P}(k+1)$, $\mathrm{Bd}(\mathrm{UB}(d,k)_{/\Pi}) = 1$, et donc $\mathrm{Bd}(\mathrm{UB}(d,k)) = \max(\epsilon_i(\varphi_G), \epsilon_e(\varphi_G))$. Comme pour les graphes de De Bruijn binaires, on montre que $\epsilon_e(\varphi_G) = \max\limits_{\{v',v''\}\in E_E(\Pi)} (|\varphi_G(v'') - \varphi_G(v')|)$. Considérons donc toutes les arêtes $\{v', v''\}$ de $E(\mathrm{UB}(d,k))$, en supposant sans perte de généralité que $q' = \varphi_Q(\pi(v')) \leq \varphi_Q(\pi(v'')) = q''$.

- Si $q'' = q'$, alors $|\varphi_G(v') - \varphi_G(v'')| \leq |\pi_{q'}| - 1 \leq \left|A_{d,k}^{d-1}(\mathcal{H}(\pi_{q'}))\right| - 1$, et donc

$$\epsilon_i(\varphi_G) \leq \max_{0\leq j\leq k} \left|A_{d,k}^{d-1}(\mathcal{H}(\pi_j))\right| - 1 \ .$$

- Si $q'' = q' + 1$, alors nécessairement $\mathcal{H}(v'') > \mathcal{H}(v')$. Si $\mathcal{H}(v'') - \mathcal{H}(v') < (d-1)$, il existe toujours une arête $\{w', w''\}$ de dilatation supérieure à celle étudiée. En effet, v'' diffère de v' par l'enlèvement d'une lettre x' et l'ajout d'une lettre x'', avec $x'' > x'$.

  - Si $x' > 0$, alors en prenant l'arête telle que $x' = 0$ on augmente la dilatation.
  - Si $x'' < (d-1)$, alors en prenant l'arête telle que $x'' = (d-1)$ on augmente la dilatation.

  Puisque, pour toute arête telle que $\mathcal{H}(v'') - \mathcal{H}(v') < (d-1)$, il existe une arête $\{w', w''\}$ de plus grande dilatation telle que $\mathcal{H}(w'') - \mathcal{H}(w') = (d-1)$, nous ne considérons dans la suite que ces dernières, obtenues par suppression d'une lettre "0" et ajout d'une lettre "(d-1)".

  - Si $v' = 0m$ et $v'' = m(d-1)$, considérons le sommet $v'_\perp = 0^{k-q}(d-1)^q = 0m_\perp$ de $\pi_{q'}$ et son voisin $v''_\perp = m_\perp(d-1)$ de $\pi_{q''}$. Par définition, pour tout m, $[m_\perp] \leq [m]$, et, arithmétiquement, $([0m] - [0m_\perp]) \leq ([m(d-1)] - [m_\perp(d-1)])$. Comme $(\varphi_G(v'_\perp) - \varphi_G(v')) = ([0m] - [0m_\perp])$ et $(\varphi_G(v''_\perp) - \varphi_G(v'')) = ([m(d-1)] - [m_\perp(d-1)])$, on a $(\varphi_G(v'') - \varphi_G(v')) \leq (\varphi_G(v''_\perp) - \varphi_G(v'_\perp)) = |\pi_{q''}| \leq \left|A_{d,k}^{d-1}(\mathcal{H}(\pi_{q''}))\right|$.
  - Si $v' = m0$ et $v'' = (d-1)m$, alors $\overline{v'} = \overline{m}(d-1)$ et $\overline{v''} = 0\overline{m}$, ce qui revient au cas précédent car, par (5.2), $|\varphi_G(v'') - \varphi_G(v')| = |\varphi_G(\overline{v''}) - \varphi_G(\overline{v'})|$.

  Par ce qui précède, $\epsilon_e(\varphi_G) = \max\limits_{0\leq j\leq k} \left|A_{d,k}^{d-1}(\mathcal{H}(\pi_j))\right| \leq \left|A_{d,k}^{d-1}\left(h_{d,k}^{d-1}\right)\right|$.

La borne supérieure s'obtient alors en prenant le maximum des majorations de $\epsilon_i(\varphi_G)$ et de $\epsilon_e(\varphi_G)$.

Pour démontrer la minoration, on utilise le théorème 3.2, sachant que $|V(\mathrm{UB}(d,k))| = d^k$ et $\mathrm{diam}(\mathrm{UB}(d,k)) = k$. □

Proposition 5.18.

$$\mathrm{Cw}(\mathrm{UB}(d,k)) \leq (d-1)\left|A_{d,k}^{d-1}\left(h_{d,k}^{d-1}\right)\right| + \frac{(d-1)d(d+1)}{3}|A_{d,k-1}(h_{d,k-1})| + 2$$

Preuve. Nous utilisons pour démontrer ce théorème la même configuration que ci-dessus quant au partitionnement Π et à la numérotation φ_Q des parties ; nous expliciterons ci-dessous la numérotation induite φ_G retenue. Puisque $\mathrm{UB}(d,k)_{/\Pi}$ est isomorphe à $\mathrm{P}(k+1)$, $\mathrm{Cw}(\mathrm{UB}(d,k)_{/\Pi}) = 1$ et $\delta(\mathrm{UB}(d,k)_{/\Pi}) = 1$.
Pour $k \geq 3$ et pour toute partie π_q, il est possible de trouver des sommets soit n'ayant pas de voisins dans π_{q-1} (de la forme $0v_2 \ldots v_{k-1}0$), soit n'ayant pas de voisins dans π_{q+1} (de la forme $(d-1)v_2 \ldots v_{k-1}(d-1)$). Il en découle que, pour toute π, $\min\left(\delta^-_{\Pi,\varphi_Q}(\pi), \delta^+_{\Pi,\varphi_Q}(\pi)\right) = 0$. En appliquant le théorème 5.6, on trouve donc que

$$\mathrm{Cw}(\mathrm{UB}(d,k)) \leq \max_{\pi\in\Pi}(\mathrm{Cw}(\mathrm{UB}(d,k)[\pi]) + |\omega(\pi)|) \quad . \tag{5.3}$$

Posons :

$$\begin{aligned} E'_I(\pi) &= \{\{v',v''\} \in E_I(\pi) \,/\, \mathcal{H}(v') \neq \mathcal{H}(v'')\} \ , \\ f'_I(\pi) &= |\{\{v',v''\} \in E'_I(\pi) \,/\, v''_1 = v'_2 \text{ et } v''_k = v'_{k-1}\}| \ , \\ f_C(\pi) &= |\{\{v',v''\} \in E_C(\pi) \,/\, v''_1 = v'_2 \text{ et } v''_k = v'_{k-1}\}| \ . \end{aligned}$$

$E'_I(\pi)$ est l'ensemble des arêtes internes à π et n'appartenant pas aux cycles d'Etzion de $\mathrm{UB}(d,k)$, qui sont les cycles (éventuellement dégénérés) obtenus par décalage cyclique des lettres des mots de la partie. $f'_I(\pi)$ est le nombre d'arêtes $\{v',v''\}$ appartenant à $E'_I(\pi)$ telles que v'' puisse être obtenu à partir de v' indifféremment par décalage à gauche ou à droite ; il représente le nombre d'arêtes doubles que contiendrait la partie si le graphe n'était pas simple. $f_C(\pi)$ dénombre ces arêtes au sein du cocycle de la partie (rappelons que $E_C(\pi)$ est équivalent à $\omega(\pi)$).
Si l'on considère le nombre d'arêtes incidentes à tous les sommets d'une partie et n'appartenant pas aux cycles d'Etzion, on obtient par construction de $\mathrm{UB}(d,k)$

$$2(d-1)|\pi| = (|E_C(\pi)| + f_C(\pi)) + 2|E'_I(\pi)| + 2f'_I(\pi) \quad . \tag{5.4}$$

Si nous choisissons une numérotation φ_G des sommets de $\mathrm{UB}(d,k)$ telle que la congestion induite dans chaque partie par les cycles d'Etzion soit minimale (figure 5.5, la congestion interne à chaque partie sera bornée supérieurement par sommation de la largeur de coupe des cycles et du nombre d'arêtes internes n'appartenant pas aux cycles, c'est-à-dire

$$\mathrm{Cw}(\mathrm{UB}(d,k)[\pi]) \leq 2 + |E'_I(\pi)| \quad . \tag{5.5}$$

En combinant les équations (5.3), (5.4) et (5.5), il vient :

$$\mathrm{Cw}(\mathrm{UB}(d,k)) \leq \max_{\pi\in\Pi}\left((d-1)|\pi| + 2 + \frac{1}{2}|E_C(\pi)| - \frac{1}{2}f_C(\pi) - f'_I(\pi)\right) \quad . \tag{5.6}$$

Évaluons donc maintenant $|E_C(\pi)|$. Soit v un sommet d'une partie π de plus grand poids $\mathcal{H}(\pi)$, et $j = \mathcal{H}(\pi) - \mathcal{H}(v)$, avec $0 \leq j \leq (d-2)$. Le nombre d'arêtes du cocycle incidentes à v par décalage gauche est

$$g(j,v_1) = \max(0,(v_1+j)-(d-2)) + \max(0,(d-1)-(v_1+j)) = \frac{1}{2} + \left|(d-2)-(v_1+j)+\frac{1}{2}\right| \ ;$$

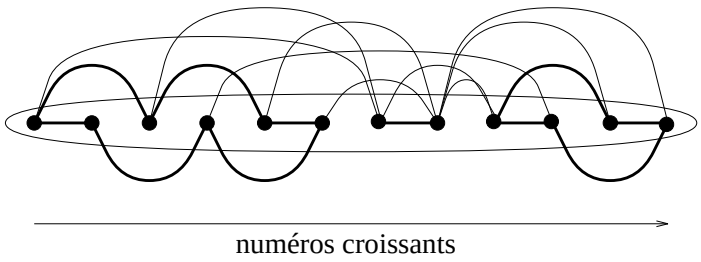


Figure 5.5: Numérotation φ_G dans une partie π de Π. Seules les arêtes internes sont représentées, avec en gras les arêtes appartenant aux cycles d'Etzion.

l'expression est identique pour les arêtes obtenues par décalage droit, avec v_k remplaçant v_1. En sommant les expressions des décalages à gauche et à droite sur tous les sommets de la partie, on obtient le cardinal du cocycle de la partie, majoré de $f_C(\pi)$:

$$|E_C(\pi)| = 2\sum_{j=0}^{d-2}\sum_{k=0}^{d-1}(g(j,k)\,|A_{d,k-1}\left(\mathcal{H}(\pi)-(j+k)\right)|) + f_C(\pi) \ .$$

Posons $x = j+k$. On a alors

$$\begin{aligned} \overline{g}(x) &\stackrel{\text{def}}{=} \tfrac{1}{2} + \left|(d-2) - x + \tfrac{1}{2}\right| \ , \\ |E_C(\pi)| &= 2\sum_{x=0}^{2d-3}(\overline{g}(x)\cdot(d-\overline{g}(x))\cdot|A_{d,k-1}\left(\mathcal{H}(\pi)-x\right)|) + f_C(\pi) \ . \end{aligned}$$

Soit $a(\pi) = \max\limits_{0\le x\le 2d-3}|A_{d,k-1}\left(\mathcal{H}(\pi)-x\right)|$. Par ce qui précède et sachant que $\overline{g}(x)$ est symétrique par rapport à $(d-2)+\frac{1}{2}$,

$$\begin{aligned} |E_C(\pi)| &\le 2a(\pi)\sum_{x=0}^{2d-3}\overline{g}(x)\cdot(d-\overline{g}(x)) + f_C(\pi) \\ &\le 4a(\pi)\sum_{x=0}^{d-2}\overline{g}(x)\cdot(d-\overline{g}(x)) + f_C(\pi) \\ &\le 4a(\pi)\sum_{x=0}^{d-2}\left((d-1)+(d-2)x-x^2\right) + f_C(\pi) \\ &\le 4a(\pi)\left((d-1)^2 + \frac{(d-2)^2(d-1)}{2} - \frac{(d-2)(d-1)(2d-3)}{6}\right) + f_C(\pi) \\ &\le 2\,\frac{(d-1)d(d+1)}{3}\,a(\pi) + f_C(\pi) \ . \end{aligned} \tag{5.7}$$

En injectant cette équation dans (5.6), on obtient

$$\begin{aligned} \mathrm{Cw}(\mathrm{UB}(d,k)) &\le \max_{\pi\in\Pi}\left((d-1)|\pi| + 2 + \frac{(d-1)d(d+1)}{3}\,a(\pi) - f'_I(\pi)\right) \\ &\le (d-1)\left|A_{d,k}^{d-1}\left(h_{d,k}^{d-1}\right)\right| + 2 + \frac{(d-1)d(d+1)}{3}\,|A_{d,k-1}\left(h_{d,k-1}\right)| \ , \end{aligned}$$

ce qui nous permet de conclure. □

5.3.5 Le graphe "*Shuffle-Exchange*"

De manière analogue à celle employée pour le graphe de De Bruijn, on quotiente SE(k) selon le poids de Hamming de ses sommets, comme représenté en figure 5.6. De plus, on définit au sein de chaque partie des sous-parties regroupant les sommets commençant et finissant par certaines lettres. Ainsi, $\pi_q(v_s, v_e) \stackrel{\text{def}}{=} \{v \in V(\text{SE}(k)) \,/\, v \in \pi_q \text{ et } v = v_s\, w\, v_e\}$, c'est-à-dire l'ensemble de mots de poids q commençant par le sous-mot v_s et finissant par le sous-mot v_e.

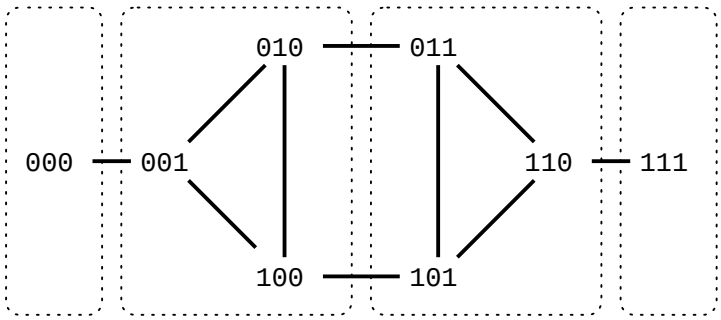


Figure 5.6: SE(3) et sa partition.

La configuration que nous utilisons est donc telle que :

- $\Pi = \{\pi_0, \pi_1, \ldots, \pi_k\}$, où $\pi_q = \{v \in V(\text{SE}(k)) \,/\, \mathcal{H}(v) = q\}$;
- $\varphi_Q(\pi_q) = q$;
- φ_G, définie à partir d'un ordre total $\succ$ sur les mots de deux lettres tel que $00 \succ 10 \succ 01 \succ 11$, est exprimée comme suit :
  - $\varphi_G(0^k) = 0$,
  - pour tous $v', v'' \in V(\text{SE}(k))$, avec $v' = v'_1 v'_2 \ldots v'_k \in \pi_{q'}$ et $v'' = v''_1 v''_2 \ldots v''_k \in \pi_{q''}$,

$$\begin{aligned}(\varphi_G(v') > \varphi_G(v'')) \Longleftrightarrow ((q' > q'') \quad & \text{ou} \\ ((q' = q'') \text{ et } & ((v'_{k-1}v'_k \succ v''_{k-1}v''_k) \quad \text{ou} \\ & ((v'_{k-1}v'_k = v''_{k-1}v''_k) \text{ et } ([v'_1 v'_2 \ldots v'_{k-2}] > \\ & \qquad [v''_1 v''_2 \ldots v''_{k-2}])))))\ .\end{aligned}$$

La numérotation φ_G revient, dans chaque partie π_q, à ordonner les ensembles $\pi_q(\epsilon, xy)$ comme indiqué dans la figure 5.7, les sommets étant numérotés par valeur de sommet décroissante dans chaque sous-ensemble.

Les propriétés de cette configuration sont regroupées dans le lemme suivant.

Lemme 5.19.

$$\begin{aligned}&(1) \qquad \text{SE}(k)_{/\Pi} \textit{ est la chaîne } \text{P}(k+1)\ . \\ &(2) \qquad |\pi_q| = \binom{k}{q} \leq \binom{k}{\lceil \frac{k}{2} \rceil}\ .\end{aligned}$$

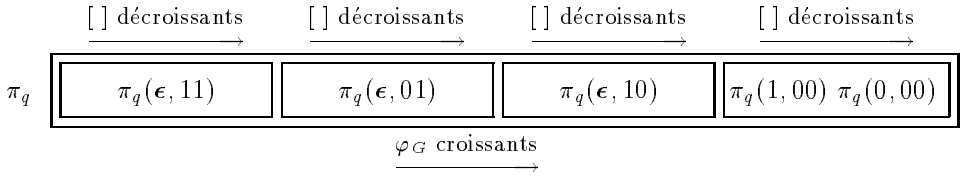


Figure 5.7: Numérotation des sommets de même poids de Hamming dans SE(k)

$$\begin{aligned} &(3) \qquad (v \in \pi_q) \Longleftrightarrow (\overline{v} \in \pi_{k-q}) \ . \\ &(4) \qquad \varphi_G(\overline{v}) = 2^k - 1 - \varphi_G(v) \ . \end{aligned} \tag{5.8}$$

Preuve. Les preuves des points de ce lemme sont analogues à celles du lemme 5.13. □

Proposition 5.20.

$$\frac{2^k - 1}{2k - 1} \leq \mathrm{Bd}(\mathrm{SE}(k)) \leq 2 \cdot \binom{k-2}{\lceil \frac{k-2}{2} \rceil}$$

Preuve. Pour $k \geq 3$ et $0 \leq q \leq k$, on a

$$\begin{aligned} |\pi_q(\epsilon, 00)| &= \begin{cases} \binom{k-2}{q} & \text{si } 0 \leq q \leq k-2 \\ 0 & \text{sinon ,} \end{cases} \\ |\pi_q(1, 00)| &= \begin{cases} \binom{k-3}{q-1} & \text{si } 1 \leq q \leq k-2 \\ 0 & \text{sinon ,} \end{cases} \\ |\pi_q(\epsilon, 01)| &= |\pi_q(\epsilon, 10)| = \begin{cases} \binom{k-2}{q-1} & \text{si } 1 \leq q \leq k-1 \\ 0 & \text{sinon ,} \end{cases} \\ |\pi_q(\epsilon, 11)| &= \begin{cases} \binom{k-2}{q-2} & \text{si } 2 \leq q \leq k \\ 0 & \text{sinon .} \end{cases} \end{aligned}$$

Puisque, d'après le lemme 5.19, $\mathrm{SE}(k)_{/\Pi}$ est isomorphe à $\mathrm{P}(k+1)$, $\mathrm{Bd}(\mathrm{SE}(k)_{/\Pi}) = 1$, et donc $\mathrm{Bd}(\mathrm{SE}(k)) = \max(\epsilon_i(\varphi_G), \epsilon_e(\varphi_G))$, avec $\epsilon_e(\varphi_G) = \max_{\{v', v''\} \in E_E(\Pi)} (|\varphi_G(v'') - \varphi_G(v')|)$.
Considérons donc toutes les arêtes $\{v', v''\}$ de $E(\mathrm{SE}(k))$, en supposant sans perte de généralité que $q' = \varphi_Q(\pi(v')) \leq \varphi_Q(\pi(v'')) = q''$.

- Si $q' = q''$, alors l'arête est de type "*shuffle*". Étudions ces arêtes par rapport aux ensembles $\pi_{q'}(\epsilon, xy)$ utilisés pour définir la numérotation φ_G à l'intérieur des parties.
  - Si $v' = m'xy$ et $v'' = m''xy$, la dilatation de $\{v', v''\}$ est bornée par $|\pi_{q'}(\epsilon, xy)| - 1$, et donc $|\varphi_G(v') - \varphi_G(v'')| \leq \binom{k-2}{\lceil \frac{k-2}{2} \rceil} - 1$.
  - Si $v' = m'00$ et $v'' = m''01$, alors nécessairement $v' = 1m00 \in \pi_{q'}(1, 00)$ et $v'' = m001 \in \pi_{q'}(\epsilon, 001)$. Puisque $|\pi_{q'}(1, 00)| = |\pi_{q'}(0, 01)|$ et $\pi_{q'}(0, 01) \subseteq \pi_{q'}(\epsilon, 01)$, alors $|\pi_{q'}(1, 00)| \leq |\pi_{q'}(\epsilon, 01)|$. Comme tout sommet de $\pi_{q'}(1, 00)$ est lié à un sommet de $\pi_{q'}(\epsilon, 001)$ par définition, la dilatation de $\{v', v''\}$ ne peut donc être supérieure à la

dilatation de l'arête reliant le sommet $v'_\top = 1^{q'}0^{k-q'}$ de $\pi_{q'}(1,00)$ au sommet $v''_\top = 1^{q'-1}0^{k-q'}1$ de $\pi_{q'}(\epsilon,01)$. De fait, par construction (voir figure 5.7), $|\varphi_G(v') - \varphi_G(v'')| \leq |\pi_{q'}(\epsilon,10)| + |\pi_{q'}(\epsilon,01)| = 2|\pi_{q'}(\epsilon,10)| \leq 2\binom{k-2}{\lceil \frac{k-2}{2} \rceil}$.

- Si $v' = m'00$ et $v'' = m''10$, alors par construction $|\varphi_G(v') - \varphi_G(v'')| \leq |\pi_{q'}(\epsilon,00)| + |\pi_{q'}(\epsilon,10)| - 1 = |\pi_{q'}(\epsilon,0)| - 1 \leq \binom{k-1}{\lceil \frac{k-1}{2} \rceil} - 1$.
- Si $v' = m'01$ et $v'' = m''10$, alors par construction $|\varphi_G(v') - \varphi_G(v'')| \leq |\pi_{q'}(\epsilon,01)| + |\pi_{q'}(\epsilon,10)| - 1 = 2|\pi_{q'}(\epsilon,10)| - 1 \leq 2\binom{k-2}{\lceil \frac{k-2}{2} \rceil} - 1$.

Toutes les autres arêtes internes sont les complémentaires d'arêtes de $\pi_{k-q'}$ de types étudiés ci-dessus, et mènent aux mêmes résultats d'après l'équation (5.8). Il en découle que $\epsilon_i(\varphi_G) \leq \max\left(\binom{k-1}{\lceil \frac{k-1}{2} \rceil} - 1, 2\binom{k-2}{\lceil \frac{k-2}{2} \rceil}\right) \leq 2\binom{k-2}{\lceil \frac{k-2}{2} \rceil}$.

- Si $q'' = q' + 1$, alors l'arête est de type "*exchange*", et donc $v' \in \pi_{q'}(\epsilon,0)$ et $v'' \in \pi_{q''}(\epsilon,1)$.
  - Si $v' = m00$, alors nécessairement $v'' = m01$, $0 \leq q' \leq k-2$, et $|\pi_{q'}(\epsilon,00)| = |\pi_{q'+1}(\epsilon,01)|$. Il en découle que toutes les arêtes de ce type ont même dilatation, et donc
    $$\begin{aligned} |\varphi_G(v') - \varphi_G(v'')| &= |\pi_{q'}(\epsilon,00)| + |\pi_{q'+1}(\epsilon,11)| \\ &= \begin{cases} \binom{k-1}{q'} & \text{si } 1 \leq q' \leq k-2 \\ 1 & \text{si } q' = 0 \ . \end{cases} \end{aligned}$$
  - Si $v' = m10$, alors nécessairement $v'' = m11$, $1 \leq q' \leq k-1$, et $|\pi_{q'}(\epsilon,10)| = |\pi_{q'+1}(\epsilon,11)|$. Toutes les arêtes de ce type ont donc même dilatation, et donc, en se basant sur le sommet v' de plus petit φ_G,
    $$|\varphi_G(v') - \varphi_G(v'')| = |\pi_{q'}(\epsilon,10)| + |\pi_{q'}(\epsilon,00)| = \begin{cases} \binom{k-1}{q'} & \text{si } 1 \leq q' \leq k-2 \\ \binom{k-2}{q'-1} & \text{si } q' = k-1 \ . \end{cases}$$

  Dans les deux cas ci-dessus, la plus grande dilatation est égale à $\binom{k-1}{q'}$ pour $q' = \lceil \frac{k-1}{2} \rceil$, et donc $\epsilon_e(\varphi_G) \leq \binom{k-1}{\lceil \frac{k-1}{2} \rceil}$.

D'après ce qui précède, on trouve en appliquant le théorème 5.3 que $\mathrm{Bd}(\mathrm{SE}(k)) \leq 2\binom{k-2}{\lceil \frac{k-2}{2} \rceil}$. On vérifie manuellement que ce résultat est également valide pour $k = 2$.

Puisque $|V(\mathrm{SE}(k))| = 2^k$ et $\mathrm{diam}(\mathrm{SE}(k)) = 2k-1$ par [22], le théorème 3.2 donne directement la minoration, ce qui conclut la preuve. □

Proposition 5.21.

$$\frac{1}{2}.\frac{2^k}{k} \leq \mathrm{Cw}(\mathrm{SE}(k)) \leq \binom{k}{\lceil \frac{k}{2} \rceil} + 2 \ .$$

Preuve. Nous utilisons pour démontrer la majoration la même configuration que pour la largeur de bande. Comme $\mathrm{SE}(k)_{/\Pi}$ est isomorphe à $\mathrm{P}(k+1)$, $\mathrm{Cw}(\mathrm{SE}(k)_{/\Pi}) = 1$ et $\delta(\mathrm{SE}(k)_{/\Pi}) = 1$. Chaque sommet étant lié par une unique arête de type "*exchange*" à un sommet de poids de Hamming

différent, $|\omega(\pi)| = |\pi|$ pour toute π.
Les arêtes $\{v', v''\}$ internes à chaque partie π lient des sommets de même poids ; ce sont les arêtes de type "*shuffle*", telles que si $v' = v_1 v_2 \dots v_{k-1} v_k$, alors $v'' = v_2 \dots v_{k-1} v_k v_1$ ou $v'' = v_k v_1 v_2 \dots v_{k-1}$.
De fait, les composantes connexes de $\mathrm{SE}(k)[\pi]$ sont des sommets isolés, des arêtes simples, ou des cycles, et donc, pour toute π, $\mathrm{Cw}(\mathrm{SE}(k)[\pi]) \leq 2$.
Pour tout k, on peut trouver dans toute partie π_q des sommets soit n'ayant pas de voisin dans π_{q-1} (de la forme $v_1 \dots v_{k-1}0$), soit n'ayant pas de voisin dans π_{q+1} (de la forme $v_1 \dots v_{k-1}1$). Il en découle que, pour toute π, $\min\left(\delta^-_{\Pi,\varphi_Q}(\pi), \delta^+_{\Pi,\varphi_Q}(\pi)\right) = 0$ et, par le théorème 5.6,

$$\begin{aligned}
\mathrm{Cw}(\mathrm{SE}(k)) &\leq \left(\mathrm{Cw}(\mathrm{SE}(k)_{/\Pi}) - \left\lceil \tfrac{\delta(\mathrm{SE}(k)_{/\Pi})}{2} \right\rceil\right) \cdot \max_{\pi\in\Pi}(|\omega(\pi)|) + \\
&\quad \max_{\pi\in\Pi}\left(\mathrm{Cw}(\mathrm{SE}(k)[\pi]) + |\omega(\pi)| - |\pi| \cdot \min\left(\delta^-_{\Pi,\varphi_Q}(\pi), \delta^+_{\Pi,\varphi_Q}(\pi)\right)\right) \\
&\leq \max_{\pi\in\Pi}(\mathrm{Cw}(\mathrm{SE}(k)[\pi]) + |\omega(\pi)|) \\
&\leq 2 + \binom{k}{\lceil \frac{k}{2} \rceil} \ .
\end{aligned}$$

La minoration découle directement du théorème 5.7 :

$$\begin{aligned}
\mathrm{Cw}(\mathrm{SE}(k)) &\geq \mathrm{Bis}_\epsilon(\mathrm{SE}(k)) \\
&\geq \tfrac{1}{2} \cdot \tfrac{2^k}{k} \qquad \text{d'après [57, page 480].}
\end{aligned}$$

□

Remarque. On peut prouver sans difficulté que $2\binom{k-2}{\lceil \frac{k-2}{2} \rceil} \underset{k\gg 1}{\approx} \frac{1}{2}\binom{k}{\lceil \frac{k}{2} \rceil}$. De ce fait, les majorations des largeurs de bande et de coupe du graphe "*Shuffle-Exchange*" appartiennent à $O\left(\frac{2^k}{\sqrt{k}}\right)$.

5.3.6 Le graphe FFT

Proposition 5.22.

$$2^{k-1} \leq \mathrm{Bd}(\mathrm{FFT}(k)) \leq 3 \cdot 2^{k-1} \ .$$

Preuve. Une manière naturelle de partitionner les graphes FFT est de les quotienter par niveaux, comme illustré en figure 5.8. Nous définissons donc la configuration suivante :

- $\Pi = \{\pi_0, \pi_1, \dots, \pi_{k-1}\}$, où $\pi_q = \{(l;m) \in V(\mathrm{FFT}(k)) \,/\, l = q\}$;
- $\varphi_Q(\pi_q) = q$;
- $\varphi_G(l;m) = \varphi_Q(\pi_l) \cdot 2^k + m$.

Par définition, chaque partie contient les sommets d'un niveau, et donc $|\pi| = 2^k$ pour toute π. Comme les sommets de chaque niveau ne sont connectés qu'à des sommets des niveaux immédiatement inférieur et supérieur, $\mathrm{FFT}(k)_{/\Pi}$ est isomorphe à $\mathrm{P}(k+1)$. On en déduit que $\mathrm{Bd}(\mathrm{FFT}(k)_{/\Pi}) = 1$, et donc $\mathrm{Bd}(\mathrm{FFT}(k)) = \max(\epsilon_i(\varphi_G), \epsilon_e(\varphi_G))$, avec $\epsilon_e(\varphi_G) = \max_{\{v',v''\}\in E_E(\Pi)}(|\varphi_G(v'') - \varphi_G(v')|)$.
Puisque aucune arête de $\mathrm{FFT}(k)$ ne lie de sommets appartenant au même niveau, $\epsilon_i(\varphi_G) = 0$.

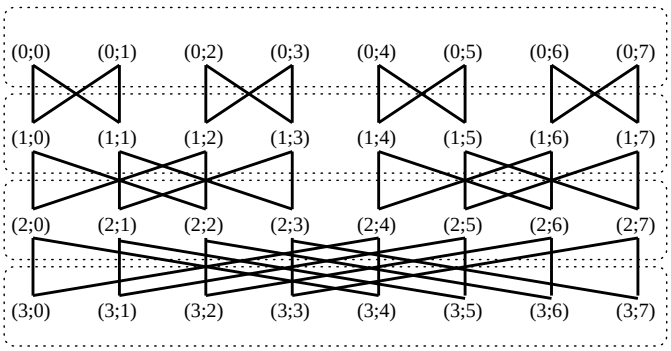


Figure 5.8: FFT(3) et son partitionnement par niveaux.

Tout sommet $v = (l; m)$ de $V(\mathrm{FFT}(k))$ tel que $0 \leq l < k$ a pour voisins les sommets $v' = (l+1; m)$ et $v'' = (l+1; m \oplus 2^l)$. De fait,

$$\begin{aligned}
|\varphi_G(v) - \varphi_G(v')| &= |(\varphi_Q(\pi_l) \cdot 2^k + m) - (\varphi_Q(\pi_{l+1}) \cdot 2^k + m)| \\
&= |\varphi_Q(\pi_l) - \varphi_Q(\pi_{l+1})| \cdot 2^k \\
&\leq \mathrm{Bd}(\mathrm{FFT}(k)_{/\Pi}) \cdot 2^k \\
&\leq 2^k \ , \\
|\varphi_G(v) - \varphi_G(v'')| &= |(\varphi_Q(\pi_l) \cdot 2^k + m) - (\varphi_Q(\pi_{l+1}) \cdot 2^k + m \oplus 2^l)| \\
&\leq \mathrm{Bd}(\mathrm{FFT}(k)_{/\Pi}) \cdot 2^k + 2^l \\
&\leq 2^k + 2^{k-1} \ ,
\end{aligned}$$

et donc $\epsilon_e(\varphi_G) \leq 3 \cdot 2^{k-1}$, ce qui donne la majoration.

Sachant que $\mathrm{diam}(\mathrm{FFT}(k)) = 2k$, le théorème 3.2 nous donne directement la minoration, c'est-à-dire

$$\mathrm{Bd}(\mathrm{FFT}(k)) \geq \left\lceil \frac{k \cdot 2^k - 1}{2k} \right\rceil = \left\lceil 2^{k-1} - \frac{1}{2k} \right\rceil = 2^{k-1} \ ,$$

puisque le contenu de la partie entière n'est jamais entier. □

Proposition 5.23.

$$\mathrm{Cw}(\mathrm{FFT}(k)) \leq 2^{k+1} - 2 \ .$$

Preuve. Une propriété intéressante des graphes FFT est que leur construction est récursive ; en effet, $\mathrm{FFT}(k)$ peut s'obtenir en plaçant deux copies de $\mathrm{FFT}(k-1)$ au dessus d'un niveau de 2^k sommets et en les reliant de manière à ce que le niveau ajouté devienne le niveau k, comme représenté en figure 5.9. La preuve du théorème repose sur cette construction, à l'image de laquelle nous définissons la configuration :

- $\Pi = \{\pi_0, \pi_1, \pi_2\}$, où $\begin{cases} \pi_0 = \{(l; m) \in V(\mathrm{FFT}(k)) \,/\, l < k, m < 2^{k-1}\} \ , \\ \pi_1 = \{(l; m) \in V(\mathrm{FFT}(k)) \,/\, l = k\} \ , \\ \pi_2 = \{(l; m) \in V(\mathrm{FFT}(k)) \,/\, l < k, m \geq 2^{k-1}\} \ ; \end{cases}$

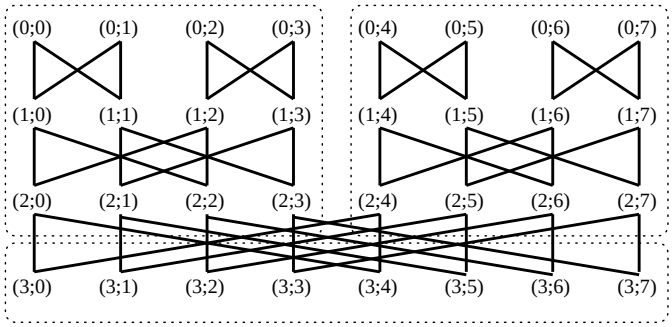


Figure 5.9: FFT(3) et son partitionnement : deux FFT(2) et un niveau de sommets.

- $\varphi_Q(\pi_q) = q$;
- $\varphi_G(l;m) = \varphi_Q(\pi_l) \cdot 2^k + m$.

Par construction, $\mathrm{FFT}(k)[\pi_0]$ et $\mathrm{FFT}(k)[\pi_2]$ sont tous deux isomorphes à $\mathrm{FFT}(k-1)$, $\mathrm{FFT}(k)[\pi_1]$ est un ensemble de sommets isolés, et le graphe quotient $\mathrm{FFT}(k)_{/\Pi}$ est une chaîne. On en déduit que $\mathrm{Cw}(\mathrm{FFT}(k)_{/\Pi}) = 1$, $\delta(\mathrm{FFT}(k)_{/\Pi}) = 1$, et que

$$\begin{array}{lll}
|\pi_0| = k \cdot 2^{k-1}, & |\omega(\pi_0)| = 2 \cdot 2^{k-1}, & \mathrm{Cw}(G[\pi_0]) = \mathrm{Cw}(\mathrm{FFT}(k-1)), \\
|\pi_1| = 2^k, & |\omega(\pi_1)| = 2 \cdot 2^k, & \mathrm{Cw}(G[\pi_1]) = 0, \\
|\pi_2| = k \cdot 2^{k-1}, & |\omega(\pi_2)| = 2 \cdot 2^{k-1}, & \mathrm{Cw}(G[\pi_2]) = \mathrm{Cw}(\mathrm{FFT}(k-1)), \\
 & \delta^-_{\Pi,\varphi_Q}(\pi_0) = 0, & \delta^+_{\Pi,\varphi_Q}(\pi_0) = 0, \\
 & \delta^-_{\Pi,\varphi_Q}(\pi_1) = 1, & \delta^+_{\Pi,\varphi_Q}(\pi_1) = 1, \\
 & \delta^-_{\Pi,\varphi_Q}(\pi_2) = 0, & \delta^+_{\Pi,\varphi_Q}(\pi_2) = 0.
\end{array}$$

En appliquant le théorème 5.6, on obtient

$$\begin{aligned}
\mathrm{Cw}(\mathrm{FFT}(k)) &\leq \max_{\pi \in \{\pi_0,\pi_1,\pi_2\}} \left(\mathrm{Cw}(\mathrm{FFT}(k)[\pi]) + |\omega(\pi)| - |\pi| \cdot \min\left(\delta^-_{\Pi,\varphi_Q}(\pi), \delta^+_{\Pi,\varphi_Q}(\pi) \right) \right) \\
&\leq \mathrm{Cw}(\mathrm{FFT}(k-1)) + 2^k \quad .
\end{aligned}$$

Comme $\mathrm{FFT}(1)$ est isomorphe à $C(4)$, $\mathrm{Cw}(\mathrm{FFT}(1)) = 2$, et donc

$$\mathrm{Cw}(\mathrm{FFT}(k)) \leq 2^k + 2^{k-1} + \cdots + 2 = 2^{k+1} - 2 \quad .$$

□

Du fait que nous ne connaissons pas de minoration précise de la bissection arête de $\mathrm{FFT}(k)$, nous ne pouvons pas utiliser le théorème 5.7 pour obtenir une minoration de sa largeur de coupe, mais il est néanmoins possible d'en déterminer un équivalent asymptotique.

Proposition 5.24. $\mathrm{Cw}(\mathrm{FFT}(k))$ *appartient à* $\Theta(2^k)$.

Preuve. Il a été montré dans [57, page 442] que $\mathrm{Bis}_\varepsilon(\mathrm{FFT}(k))$ appartient à $\Theta\left(\frac{(k+1)2^k}{\log_2((k+1)2^k)}\right)$. Il existe donc un nombre $\alpha > 0$ et un entier $k_0 > 0$ tels que pour tout $k > k_0$, $\mathrm{Bis}_\varepsilon(\mathrm{FFT}(k)) >$

$\alpha\left(\frac{(k+1)2^k}{\log_2((k+1)2^k)}\right)$. En appliquant le théorème 5.7 sous ces conditions, on trouve que

$$\begin{aligned} \mathrm{Cw}(\mathrm{FFT}(k)) &\geq \mathrm{Bis}_\varepsilon(\mathrm{FFT}(k)) \\ &\geq \alpha\left(\frac{(k+1)2^k}{\log_2((k+1)2^k)}\right) = \alpha\left(\frac{2^k}{\frac{k}{k+1}+\frac{\log_2(k+1)}{k+1}}\right) \\ &\geq \alpha 2^{k-1} \ , \end{aligned}$$

et $\mathrm{Cw}(\mathrm{FFT}(k))$ appartient donc à $\Omega(2^k)$. Puisque la proposition 5.23 implique que $\mathrm{Cw}(\mathrm{FFT}(k))$ appartient également à $O(2^k)$, alors $\mathrm{Cw}(\mathrm{FFT}(k))$ appartient à $\Theta(2^k)$. □

5.3.7 Le graphe "*Butterfly*"

Proposition 5.25.

$$\left\lfloor \frac{8}{3} \cdot 2^{k-2} \right\rfloor \leq \mathrm{Bd}(\mathrm{BF}(k)) \leq 9 \cdot 2^{k-2} \ .$$

Preuve. Le graphe "*Butterfly*" se prête au même type de quotientement que le graphe FFT auquel il est apparenté. Nous définissons donc la configuration suivante, illustrée en figure 5.10 :

- $\Pi = \{\pi_0, \pi_1, \ldots, \pi_{k-1}\}$, où $\pi_q = \{(l;m) \in V(\mathrm{BF}(k)) \,/\, l = q\}$;
- $\varphi_Q(\pi_q) = \begin{cases} 2(q+1) & \text{si } 0 \leq q < \left\lfloor \frac{k-1}{2} \right\rfloor \ , \\ 2(k-(q+1))-1 & \text{si } \left\lfloor \frac{k-1}{2} \right\rfloor \leq q < k-1 \ , \\ 0 & \text{si } q = k-1 \ ; \end{cases}$
- $\varphi_G(l;m) = \varphi_Q(\pi_l) \cdot 2^k + m$.

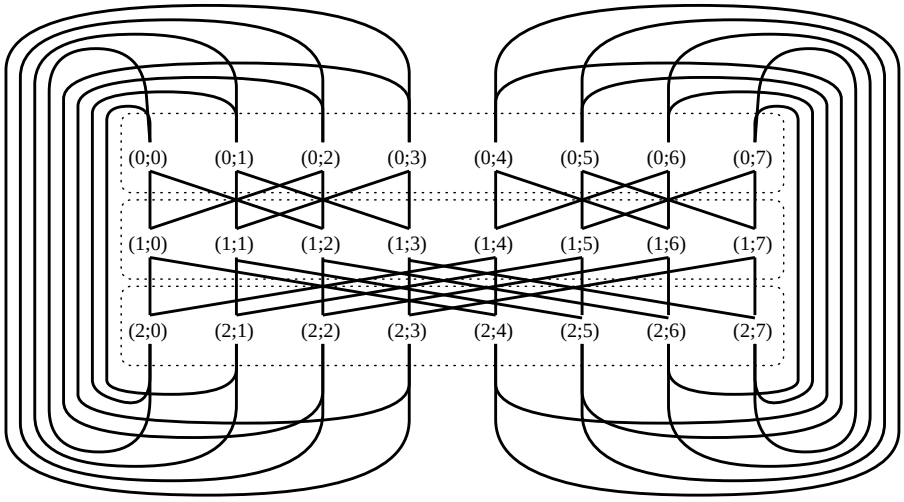


Figure 5.10: BF(3) et son partitionnement par niveaux.

Par définition, chaque partie contient les sommets d'un niveau, et donc $|\pi| = 2^k$ pour toute π. Les sommets d'un niveau l n'étant connectés qu'aux sommets des niveaux $(l-1)$ mod k et $(l+1)$ mod k,

$\mathrm{BF}(k)_{/\Pi}$ est isomorphe à $C(k)$, et donc $\mathrm{Bd}(\mathrm{BF}(k)_{/\Pi}) = 2$. Il est à noter que la numérotation φ_Q ci-dessus donne une dilatation égale à cette borne, et est telle que $|\varphi_Q(\pi_{k-2}) - \varphi_Q(\pi_{k-1})| = 1$.
Puisque aucune arête de $\mathrm{BF}(k)$ ne lie de sommets appartenant au même niveau, $\epsilon_i(\varphi_G) = 0$.
Tout sommet $v = (l; m)$ appartenant à $V(\mathrm{BF}(k))$ est lié aux sommets $v' = ((l+1) \bmod k; m)$ et $v'' = ((l+1) \bmod k; m \oplus 2^{(l+1) \bmod k})$. De fait,

$$\begin{aligned}
|\varphi_G(v) - \varphi_G(v')| &= |(\varphi_Q(\pi_l) \cdot 2^k + m) - (\varphi_Q(\pi_{(l+1) \bmod k}) \cdot 2^k + m)| \\
&= |\varphi_Q(\pi_l) - \varphi_Q(\pi_{(l+1) \bmod k})| \cdot 2^k \\
&\leq \mathrm{Bd}(\mathrm{BF}(k)_{/\Pi}) \cdot 2^k \\
&\leq 2 \cdot 2^k \ , \\
|\varphi_G(v) - \varphi_G(v'')| &= |(\varphi_Q(\pi_l) \cdot 2^k + m) - (\varphi_Q(\pi_{(l+1) \bmod k}) \cdot 2^k + m \oplus 2^{(l+1) \bmod k})| \\
&\leq \mathrm{Bd}(\mathrm{BF}(k)_{/\Pi}) \cdot 2^k + 2^{(l+1) \bmod k} \\
&\leq 2 \cdot 2^k + 2^{k-1} \ ,
\end{aligned}$$

et donc

$$\epsilon_e(\varphi_G) = 2^k + 2^{k-1} \ .$$

Cependant, la numérotation φ_Q choisie est telle que toute les arêtes qui maximisent $\epsilon_e(\varphi_G)$ lient la partie π_{k-2} à la partie π_{k-1}, et $(\varphi_Q(\pi_{k-2}) - \varphi_Q(\pi_{k-1}))$ est par construction toujours égale à 1 (et non à 2, qui est la largeur de bande de Q). Par conséquent, la dilatation maximale des arêtes externes est en fait moindre que celle obtenue par le théorème 5.3. En effet,

$$\begin{aligned}
\mathrm{Bd}(\mathrm{BF}(k)) &\leq \max(2^k \cdot |\varphi_Q(\pi_{k-1}) - \varphi_Q(\pi_{k-2})| + 2^{k-1}, \\
&\qquad 2^k \cdot |\varphi_Q(\pi_{k-2}) - \varphi_Q(\pi_{k-3})| + 2^{k-2}) \\
&\leq \max(1 \cdot 2^k + 2^{k-1}, 2 \cdot 2^k + 2^{k-2}) \\
&\leq 9 \cdot 2^{k-2} \ .
\end{aligned}$$

Nous savons par [57] que $\mathrm{diam}(\mathrm{BF}(k)) = \lfloor \frac{3}{2}k \rfloor$. Par le théorème 3.2,

$$\mathrm{Bd}(\mathrm{BF}(k)) \geq \left\lceil \frac{k \cdot 2^k - 1}{\lfloor \frac{3}{2}k \rfloor} \right\rceil \geq \left\lceil \frac{k \cdot 2^k - 1}{\frac{3}{2}k} \right\rceil = \left\lceil \frac{8}{3} \cdot 2^{k-2} - \frac{2}{3k} \right\rceil \geq \left\lfloor \frac{8}{3} \cdot 2^{k-2} \right\rfloor \ ,$$

puisque le contenu de la partie entière n'est jamais entier. □

Proposition 5.26.

$$\mathrm{Cw}(\mathrm{BF}(k)) \leq 3 \cdot 2^k \ .$$

Preuve. De manière analogue à celle employée pour le graphe FFT, on utilise les propriétés de décomposition récursive du graphe "*Butterfly*" en se basant sur une configuration judicieuse.
Soit $\overline{\mathrm{BF}}(k)$ le graphe obtenu à partir de $\mathrm{BF}(k)$ par la suppression des arêtes liant les sommets du niveau 0 à ceux du niveau $k-1$. $\mathrm{BF}(k)$ peut être construit en plaçant deux copies de $\overline{\mathrm{BF}}(k-1)$ au dessus d'un niveau de 2^k sommets et en les liant de manière à ce que le niveau ajouté devienne le niveau $k-1$; cette construction s'applique également, de manière directe, à $\overline{\mathrm{BF}}(k)$. Ces deux décompositions sont représentées respectivement en figures 5.11 et 5.12. La première étape de la preuve consiste à exprimer $\mathrm{Cw}(\mathrm{BF}(k))$ en fonction de $\mathrm{Cw}(\overline{\mathrm{BF}}(k-1))$, au moyen de la configuration suivante :

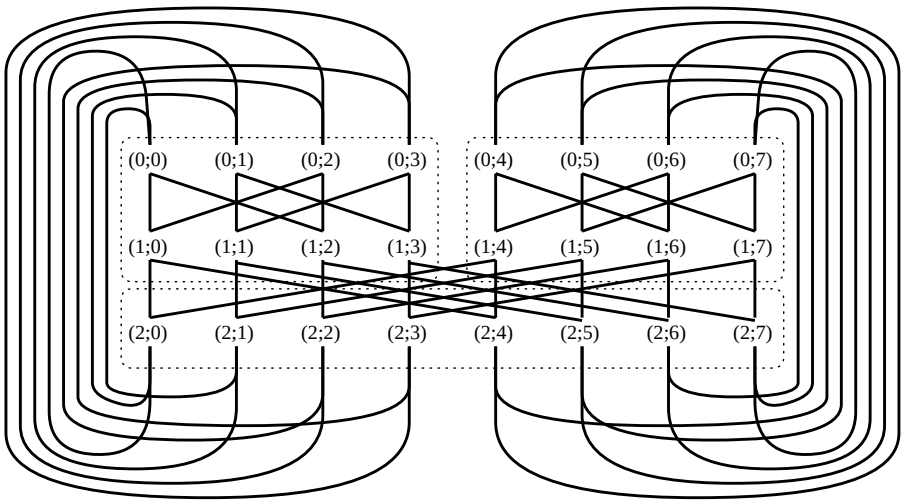
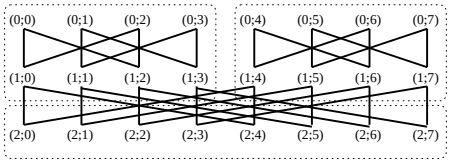
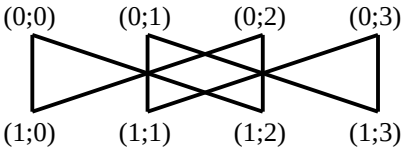


Figure 5.11: BF(3) et son partitionnement : deux $\overline{\mathrm{BF}}(2)$ et un niveau de sommets.



a. $\overline{\mathrm{BF}}(3)$ et son partitionnement : deux $\overline{\mathrm{BF}}(2)$ et un niveau de sommets.



b. $\overline{\mathrm{BF}}(2)$ est isomorphe à deux $C(4)$.

Figure 5.12: Structure récursive du graphe $\overline{\mathrm{BF}}$.

- $\Pi = \{\pi_0, \pi_1, \pi_2\}$, où $\begin{cases} \pi_0 = \{(l;m) \in V(\mathrm{BF}(k)) \,/\, l < k-1, m < 2^{k-1}\} \ , \\ \pi_1 = \{(l;m) \in V(\mathrm{BF}(k)) \,/\, l = k-1\} \ , \\ \pi_2 = \{(l;m) \in V(\mathrm{BF}(k)) \,/\, l < k-1, m \geq 2^{k-1}\} \ ; \end{cases}$
- $\varphi_Q(\pi_q) = q$;
- $\varphi_G(l;m) = \varphi_Q(\pi_l) \cdot 2^k + m$.

D'après ce qui précède, $\mathrm{BF}(k)[\pi_0]$ et $\mathrm{BF}(k)[\pi_2]$ sont tous deux isomorphes à $\overline{\mathrm{BF}}(k-1)$, $\mathrm{BF}(k)[\pi_1]$ est un ensemble de sommets isolés, et $\mathrm{BF}(k)_{/\Pi}$ est isomorphe à P(3). On a donc $\mathrm{Cw}(\mathrm{BF}(k)_{/\Pi}) = 1$, $\delta(\mathrm{BF}(k)_{/\Pi}) = 1$, et

$$
\begin{array}{lll}
|\pi_0| = (k-1)2^{k-1}, & |\omega(\pi_0)| = 4 \cdot 2^{k-1}, & \mathrm{Cw}(\mathrm{BF}(k)[\pi_0]) = \mathrm{Cw}(\overline{\mathrm{BF}}(k-1)), \\
|\pi_1| = 2^k, & |\omega(\pi_1)| = 4 \cdot 2^k, & \mathrm{Cw}(\mathrm{BF}(k)[\pi_1]) = 0, \\
|\pi_2| = (k-1)2^{k-1}, & |\omega(\pi_2)| = 4 \cdot 2^{k-1}, & \mathrm{Cw}(\mathrm{BF}(k)[\pi_2]) = \mathrm{Cw}(\overline{\mathrm{BF}}(k-1)),
\end{array}
$$

$$
\begin{array}{ll}
\delta^-_{\Pi,\varphi_Q}(\pi_0) = 0, & \delta^+_{\Pi,\varphi_Q}(\pi_0) = 0, \\
\delta^-_{\Pi,\varphi_Q}(\pi_1) = 1, & \delta^+_{\Pi,\varphi_Q}(\pi_1) = 1, \\
\delta^-_{\Pi,\varphi_Q}(\pi_2) = 0, & \delta^+_{\Pi,\varphi_Q}(\pi_2) = 0.
\end{array}
$$

Le théorème 5.6 donne donc

$$\begin{aligned} \mathrm{Cw}(\mathrm{BF}(k)) &\leq \max_{\pi \in \{\pi_0, \pi_1, \pi_2\}} \left(\mathrm{Cw}(\mathrm{BF}(k)[\pi]) + |\omega(\pi)| - |\pi| \cdot \min\left(\delta^-_{\Pi,\varphi_Q}(\pi), \delta^+_{\Pi,\varphi_Q}(\pi) \right) \right) \\ &\leq \max\left(\mathrm{Cw}(\overline{\mathrm{BF}}(k-1)) + 2^{k+1}, 3 \cdot 2^k \right) \quad . \end{aligned} \tag{5.9}$$

La seconde étape utilise la décomposition récursive de $\overline{\mathrm{BF}}(k)$, en se basant sur une configuration identique à celle définie précédemment. On obtient alors, par le théorème 5.6,

$$\begin{aligned} \mathrm{Cw}(\overline{\mathrm{BF}}(k)) &\leq \max\left(\mathrm{Cw}(\overline{\mathrm{BF}}(k-1)) + 2^k, 2^k \right) \\ &\leq \mathrm{Cw}(\overline{\mathrm{BF}}(k-1)) + 2^k \quad . \end{aligned} \tag{5.10}$$

Comme $\overline{\mathrm{BF}}(2)$ est isomorphe à deux $C(4)$, $\mathrm{Cw}(\overline{\mathrm{BF}}(2)) = 2$, ce qui permet de résoudre l'équation récurrente (5.10) en $\mathrm{Cw}(\overline{\mathrm{BF}}(k)) \leq 2^{k+1} - 6$ qui, injectée dans l'équation (5.9), amène au résultat annoncé. □

Comme nous ne disposons pas de minoration de la bissection arête de $\mathrm{BF}(k)$, nous ne pouvons en déduire de minoration de sa largeur de coupe, mais nous pouvons cependant en donner un équivalent asymptotique.

Proposition 5.27. $\mathrm{Cw}(\mathrm{BF}(k))$ *appartient à* $\Theta(2^k)$.

Preuve. Il a été montré dans [57, page 451] que $\mathrm{Bis}_\varepsilon(\mathrm{CCC}(k))$ appartient à $\Theta\left(\frac{k2^k}{\log_2(k2^k)}\right)$. Il existe donc un nombre $\alpha > 0$ et un entier $k_0 > 0$ tels que pour tout $k > k_0$, $\mathrm{Bis}_\varepsilon(\mathrm{CCC}(k)) > \alpha\left(\frac{k2^k}{\log_2(k2^k)}\right)$. En outre, puisque $CCC(k)$ est un graphe partiel de $\mathrm{BF}(k)$ [27], $\mathrm{Bis}_\varepsilon(CCC(k)) \leq \mathrm{Bis}_\varepsilon(\mathrm{BF}(k))$. En combinant ces résultats avec le théorème 5.7, il vient :

$$\begin{aligned} \mathrm{Cw}(\mathrm{BF}(k)) &\geq \mathrm{Bis}_\varepsilon(\mathrm{BF}(k)) \geq \mathrm{Bis}_\varepsilon(\mathrm{CCC}(k)) \\ &\geq \alpha\left(\frac{k2^k}{\log_2(k2^k)}\right) = \alpha\left(\frac{2^k}{1+\frac{\log_2(k)}{k}}\right) \\ &\geq \alpha 2^{k-1} \quad , \end{aligned}$$

et donc $\mathrm{Cw}(\mathrm{BF}(k))$ appartient à $\Omega(2^k)$. Puisque la proposition 5.26 implique que $\mathrm{Cw}(\mathrm{BF}(k))$ appartient également à $O(2^k)$, alors $\mathrm{Cw}(\mathrm{BF}(k))$ appartient à $\Theta(2^k)$. □

5.3.8 Le graphe CCC

Proposition 5.28.

$$\left\lfloor \frac{8}{5} \cdot 2^{k-2} \right\rfloor \leq \mathrm{Bd}(\mathrm{CCC}(k)) \leq 9 \cdot 2^{k-2} \quad .$$

Preuve. Comme il a été montré dans [27] que $\mathrm{CCC}(k)$ est un graphe partiel de $\mathrm{BF}(k)$, $\mathrm{Bd}(\mathrm{CCC}(k)) \leq \mathrm{Bd}(\mathrm{BF}(k)) \leq 9 \cdot 2^{k-2}$.

On sait par [22] que $\mathrm{diam}(\mathrm{CCC}(k)) = 2k - 1 + \max\left(1, \lfloor \frac{k-2}{2} \rfloor\right)$. En appliquant le théorème 3.2, on obtient pour $k \geq 4$

$$\mathrm{Bd}(CCC(k)) \geq \left\lceil \frac{k \cdot 2^k - 1}{\lfloor \frac{5}{2}k \rfloor - 2} \right\rceil \geq \left\lceil \frac{k \cdot 2^k - 1}{\frac{5}{2}k} \right\rceil \geq \left\lceil \frac{8}{5}2^{k-2} - \frac{2}{5k} \right\rceil \geq \left\lfloor \frac{8}{5}2^{k-2} \right\rfloor \quad ,$$

puisque le contenu de la partie entière n'est jamais entier.
Par calcul direct, on vérifie que $\left\lceil \frac{k\cdot 2^{k}-1}{\mathrm{diam}(\mathrm{CCC}(k))} \right\rceil \geq \left\lfloor \frac{8}{5}2^{k-2} \right\rfloor$ lorsque $k = 1$, 2, et 3, ce qui conclut la preuve. □

Proposition 5.29.

$$\left\lfloor \frac{1}{5}\cdot 2^{k+1} \right\rfloor \leq \mathrm{Cw}(\mathrm{CCC}(k)) \leq 2^{k+1}+1 \ .$$

Preuve. Shahrokhi et Szekely [83] ont donné récemment une minoration de la bissection arête d'un graphe G quelconque par une expression dépendant de la congestion du plongement dans G du graphe complet double (c'est-à-dire le graphe complet tel que toute paire de sommets soit liée par deux arêtes), noté $2K$, de même ordre. Si l'on note $\mathrm{cg}(2K(|V(G)|), G)$ la congestion de ce plongement,

$$\mathrm{Bis}_e(G) \geq \frac{\mathrm{Bis}_e(2K(|V(G)|))}{\mathrm{cg}(2K(|V(G)|), G)} = \frac{2\left\lceil \frac{|V(G)|}{2} \right\rceil \left\lfloor \frac{|V(G)|}{2} \right\rfloor}{\mathrm{cg}(2K(|V(G)|), G)} \ .$$

En utilisant la valeur de $\mathrm{cg}(2K(|V(\mathrm{CCC}(k))|), \mathrm{CCC}(k))$ fournie dans [83, tableau 1], on trouve alors

$$\mathrm{Bis}_e(\mathrm{CCC}(k)) \geq \frac{2\left\lceil \frac{k\cdot 2^k}{2} \right\rceil \left\lfloor \frac{k\cdot 2^k}{2} \right\rfloor}{\frac{5}{4}\cdot k^2\cdot 2^k(1-o(1))} \geq \left\lfloor \frac{1}{5}\cdot 2^{k+1} \right\rfloor \ ,$$

et le théorème 5.7 nous amène au résultat annoncé.

Afin de prouver la majoration, nous proposons, à l'image de ce qui a été fait pour le graphe "*Butterfly*", une décomposition récursive en deux étapes du graphe CCC. Soit $\overline{\mathrm{CCC}}(k)$ le graphe obtenu à partir de $\mathrm{CCC}(k)$ par la suppression des arêtes liant les sommets du niveau 0 à ceux du niveau $k-1$. $\mathrm{CCC}(k)$ peut être construit en plaçant deux copies de $\overline{\mathrm{CCC}}(k-1)$ au dessus d'un niveau de 2^k sommets et en les liant de manière à ce que le niveau ajouté devienne le niveau $k-1$, comme le montre la figure 5.13. La construction équivalente pour le $\overline{\mathrm{CCC}}$ est illustrée en figure 5.14. On

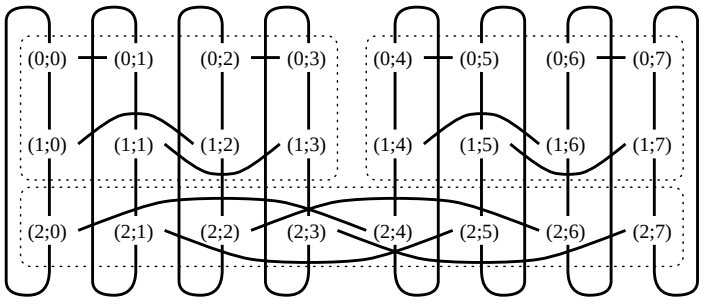
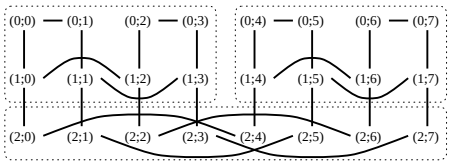


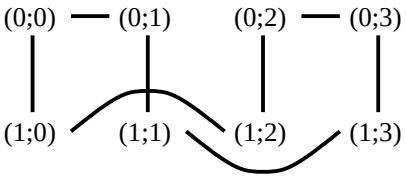
Figure 5.13: CCC(3) et son partitionnement : deux $\overline{\mathrm{CCC}}(2)$ et un niveau de sommets.

commence alors par exprimer $\mathrm{Cw}(\mathrm{CCC}(k))$ en fonction de $\mathrm{Cw}(\overline{\mathrm{CCC}}(k-1))$, grâce à la configuration ci-dessous :

- $\Pi = \{\pi_0, \pi_1, \pi_2\}$, où $\begin{cases} \pi_0 = \{(l;m) \in V(\mathrm{CCC}(k)) \,/\, l < k-1, m < 2^{k-1}\} \ , \\ \pi_1 = \{(l;m) \in V(\mathrm{CCC}(k)) \,/\, l = k-1\} \ , \\ \pi_2 = \{(l;m) \in V(\mathrm{CCC}(k)) \,/\, l < k-1, m \geq 2^{k-1}\} \ ; \end{cases}$



a. $\overline{\mathrm{CCC}}(3)$ et son partitionnement : deux $\overline{\mathrm{CCC}}(2)$ et un niveau de sommets.



b. $\overline{\mathrm{CCC}}(2)$ est isomorphe à $C(8)$.

Figure 5.14: Structure récursive du graphe $\overline{\mathrm{CCC}}$.

- $\varphi_Q(\pi_q) = q$;
- $\varphi_G(l;m) = \varphi_Q(\pi_l) \cdot 2^k + m$.

D'après ces définitions, $\mathrm{CCC}(k)[\pi_0]$ et $\mathrm{CCC}(k)[\pi_2]$ sont tous deux isomorphes à $\overline{\mathrm{CCC}}(k-1)$, $\mathrm{CCC}(k)[\pi_1]$ est un ensemble d'arêtes simples, et $\mathrm{CCC}(k)_{/\Pi}$ est isomorphe à P(3). On en déduit que $\mathrm{Cw}(\mathrm{CCC}(k)_{/\Pi}) = 1$, $\delta(\mathrm{CCC}(k)_{/\Pi}) = 1$, et

$$
\begin{array}{lll}
|\pi_0| = (k-1)2^{k-1}, & |\omega(\pi_0)| = 2 \cdot 2^{k-1}, & \mathrm{Cw}(\mathrm{CCC}(k)[\pi_0]) = \mathrm{Cw}(\overline{\mathrm{CCC}}(k-1)), \\
|\pi_1| = 2^k, & |\omega(\pi_1)| = 2 \cdot 2^k, & \mathrm{Cw}(\mathrm{CCC}(k)[\pi_1]) = 1, \\
|\pi_2| = (k-1)2^{k-1}, & |\omega(\pi_2)| = 2 \cdot 2^{k-1}, & \mathrm{Cw}(\mathrm{CCC}(k)[\pi_2]) = \mathrm{Cw}(\overline{\mathrm{CCC}}(k-1)), \\
 & \delta^-_{\Pi,\varphi_Q}(\pi_0) = 0, & \delta^+_{\Pi,\varphi_Q}(\pi_0) = 0, \\
 & \delta^-_{\Pi,\varphi_Q}(\pi_1) = 0, & \delta^+_{\Pi,\varphi_Q}(\pi_1) = 0, \\
 & \delta^-_{\Pi,\varphi_Q}(\pi_2) = 0, & \delta^+_{\Pi,\varphi_Q}(\pi_2) = 0.
\end{array}
$$

Par le théorème 5.6, on trouve alors

$$
\begin{aligned}
\mathrm{Cw}(\mathrm{CCC}(k)) &\leq \max_{\pi \in \{\pi_0,\pi_1,\pi_2\}} \left(\mathrm{Cw}(\mathrm{CCC}(k)[\pi]) + |\omega(\pi)| - |\pi| \cdot \min\left(\delta^-_{\Pi,\varphi_Q}(\pi), \delta^+_{\Pi,\varphi_Q}(\pi)\right)\right) \\
&\leq \max\left(\mathrm{Cw}(\overline{\mathrm{CCC}}(k-1)) + 2^k, 2^{k+1}+1\right) \quad . \qquad (5.11)
\end{aligned}
$$

On utilise alors la décomposition récursive de $\overline{\mathrm{CCC}}(k)$ en s'appuyant sur la même configuration que ci-dessus, ce qui donne

$$
\mathrm{Cw}(\overline{\mathrm{CCC}}(k)) \leq \max\left(\mathrm{Cw}(\overline{\mathrm{CCC}}(k-1)) + 2^{k-1}, 2^k + 1\right) \quad . \qquad (5.12)
$$

Comme $\overline{\mathrm{CCC}}(2)$ est isomorphe à $C(8)$, $\mathrm{Cw}(\overline{\mathrm{CCC}}(2)) = 2$. En injectant ce résultat dans l'équation (5.12), et en étudiant la récurrence, on obtient

$$
\mathrm{Cw}(\overline{\mathrm{CCC}}(k)) \leq 2^k + 1 \quad .
$$

Il sufft alors de répercuter ce résultat dans (5.11) pour conclure. □

Chapitre 6

Évaluations asymptotiques et application à $\mathrm{UB}(d,k)$

Dans le chapitre précédent, nous avons appliqué les théorèmes généraux énoncés précédemment à plusieurs types de graphes. Dans le cas du graphe de De Bruijn d-aire, nous n'avons pu qu'établir des majorations symboliques, exprimées en fonction de $|A_{d,k}(h)|$ et $\left|A_{d,k}^{d-1}(h)\right|$.
C'est dans ce cadre que nous sommes amenés à nous intéresser aux valeurs de $|A_{d,k}(h)|$ et $\left|A_{d,k}^{d-1}(h)\right|$, et en donnons un équivalent asymptotique.

6.1 Évaluation de $|A_{d,k}(h)|$

Nous donnons dans cette section trois expressions distinctes de $|A_{d,k}(h)|$, obtenues grâce à des approches différentes.

Proposition 6.30.

$$|A_{d,k}(h)| = \sum_{j=0}^{d-1} |A_{d,k-1}(h-j)| \ .$$

Preuve. Par définition de $A_{d,k}(h)$,

$$\begin{aligned} |A_{d,k}(h)| &= \left| \left\{ m \in \mathcal{A}_d^k \ / \ \sum_{p=1}^{k} m_p = h \right\} \right| \\ &= \sum_{j=0}^{d-1} \left| \left\{ m \in \mathcal{A}_d^{k-1} \ / \ \sum_{p=1}^{k-1} m_p = h-j \right\} \right| \\ &= \sum_{j=0}^{d-1} |A_{d,k-1}(h-j)| \ . \end{aligned}$$

□

Proposition 6.31.

$$|A_{d,k}(h)| = \sum_{0\le p\le \lfloor \frac{h}{d} \rfloor} (-1)^p \binom{k}{p} \binom{(h-dp)+(k-1)}{(k-1)} \ .$$

Preuve. La preuve de cette proposition est basée sur les séries formelles. L'idée directrice est de représenter l'expression à évaluer au moyen de séries formelles, afin d'utiliser les opérations sur les séries pour la simplifier.
Puisque multiplier deux puissances d'un nombre équivaut à additionner leurs exposants, nous pouvons dénombrer les termes de poids (c'est-à-dire d'exposant) donné par la série

$$\sum_{p=0}^{\infty} a_p z^p \stackrel{\text{def}}{=} \left(1+z+z^2+\cdots+z^{d-1}\right)^k \ , \tag{6.1}$$

dans laquelle la somme $1+z+\cdots+z^{d-1}$ énumère les poids possibles d'une lettre, et la puissance k le fait qu'on considère k lettres indépendantes. Compter le nombre de mots ayant un certain poids h revient donc à compter le nombre d'occurrences du terme z^h, c'est-à-dire la valeur de a_h. De fait,

$$\begin{aligned} \sum_{p=0}^{\infty} a_p z^p &= \left(1+z+z^2+\cdots+z^{d-1}\right)^k \\ &= \left(1-z^d\right)^k (1-z)^{-k} \\ &= \left(\sum_{p=0}^{k} \binom{k}{p} (-1)^p z^{dp}\right) \left(\sum_{q=0}^{\infty} \binom{-k}{q} (-1)^q z^q\right) \ , \end{aligned}$$

où

$$\binom{-\alpha}{\beta} \stackrel{\text{def}}{=} \frac{(-\alpha)(-\alpha-1)\ldots(-\alpha-\beta+1)}{\beta!} = (-1)^\beta \binom{\alpha+\beta-1}{\alpha-1} \ .$$

$|A_{d,k}(h)|$ s'obtient alors en prenant le $h^{\text{ième}}$ coefficient de la série.

$$\begin{aligned} |A_{d,k}(h)| &= a_h \\ &= \sum_{0\le p\le \lfloor \frac{h}{d} \rfloor} (-1)^{p+h-dp} \binom{k}{p} \binom{-k}{h-dp} \\ &= \sum_{0\le p\le \lfloor \frac{h}{d} \rfloor} (-1)^p \binom{k}{p} \binom{(h-dp)+(k-1)}{(k-1)} \ . \end{aligned}$$

□

La présence de sommes discrètes dans l'expression ci-dessus la rend inexploitable en pratique, mis à part pour des évaluations numériques.
Pour contourner cette difficulté, une solution consiste à trouver une expression propice au passage dans le domaine continu ; c'est ce que nous faisons dans la proposition suivante.

Proposition 6.32.

$$|A_{d,k}(h)| = \frac{2}{\pi} \int_0^{\frac{\pi}{2}} \cos(x(2h-k(d-1))) \left(\frac{\sin(dx)}{\sin(x)}\right)^k \mathbf{d}x \ .$$

Preuve. Le nombre de mots de $\mathcal{A}_d^k$ de poids de Hamming étendu h peut être calculé en parcourant toutes les combinaisons possibles et en incrémentant un compteur chaque fois qu'un tel mot est

trouvé. Ainsi, en utilisant le symbole de Kronecker

$$\delta_{p,0} = \begin{cases} 1 & \text{si } p = 0 \\ 0 & \text{si } p \neq 0 \end{cases} ,$$

on peut écrire

$$|A_{d,k}(h)| = \sum_{m_1=0}^{(d-1)} \sum_{m_2=0}^{(d-1)} \cdots \sum_{m_k=0}^{(d-1)} \delta_{(h-\sum_{j=1}^{k} m_j),0} .$$

Pour tout entier p, le symbole de Kronecker peut s'écrire

$$\delta_{p,0} = \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{ipt} \mathbf{d}t ,$$

donc

$$\begin{aligned} |A_{d,k}(h)| &= \sum_{m_1=0}^{(d-1)} \sum_{m_2=0}^{(d-1)} \cdots \sum_{m_k=0}^{(d-1)} \left(\frac{1}{2\pi} \int_{-\pi}^{\pi} e^{i\left(h-\sum_{j=1}^{k} m_j\right)t} \mathbf{d}t \right) \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} \left(\sum_{m_1=0}^{(d-1)} \sum_{m_2=0}^{(d-1)} \cdots \sum_{m_k=0}^{(d-1)} e^{iht} e^{-it\sum_{j=1}^{k} m_j} \right) \mathbf{d}t . \end{aligned}$$

Comme, d'après l'équation (6.1), il advient que, pour tout $z \neq 1$,

$$\sum_{m_1=0}^{(d-1)} \sum_{m_2=0}^{(d-1)} \cdots \sum_{m_k=0}^{(d-1)} z^{\left(\sum_{j=1}^{k} m_j\right)} = \left(1 + z + \cdots + z^{d-1}\right)^k = \left(\frac{1-z^d}{1-z} \right)^k ,$$

l'équation précédente peut être réécrite en

$$\begin{aligned} |A_{d,k}(h)| &= \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{iht} \left(\frac{1-e^{-idt}}{1-e^{-it}} \right)^k \mathbf{d}t \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{iht} \left(\frac{e^{-\frac{it}{2}}}{e^{-\frac{idt}{2}}} \cdot \frac{e^{\frac{idt}{2}} - e^{-\frac{idt}{2}}}{e^{\frac{it}{2}} - e^{-\frac{it}{2}}} \right)^k \mathbf{d}t \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{i(2h-k(d-1))\frac{t}{2}} \left(\frac{\sin\left(\frac{dt}{2}\right)}{\sin\left(\frac{t}{2}\right)} \right)^k \mathbf{d}t . \end{aligned}$$

Une étude de symétrie de la fonction contenue dans l'intégrale par rapport à la variable d'intégration montre que sa partie imaginaire est impaire et sa partie réelle paire. Il en découle que la partie imaginaire peut être supprimée, ce à quoi on pouvait s'attendre puisqu'on dénombre des quantités réelles. Si l'on effectue en outre le changement de variable $x = \frac{t}{2}$, on obtient

$$|A_{d,k}(h)| = \frac{2}{\pi} \int_0^{\frac{\pi}{2}} \cos(x(2h - k(d-1))) \left(\frac{\sin(dx)}{\sin(x)} \right)^k \mathbf{d}x ,$$

qui est le résultat annoncé. □

Lemme 6.33.

(1) $|A_{d,k}(h)|$ *est symétrique par rapport à* $\frac{k(d-1)}{2}$.

(2) $|A_{d,k}(h)|$ *croît pour* h *compris entre* 0 *et* $\left\lfloor \frac{k(d-1)}{2} \right\rfloor$.

Preuve. (1) La symétrie du problème étudié implique que $|A_{d,k}(h)|$ est égal à $|A_{d,k}(k(d-1)-h)|$, puisqu'à tout mot de poids h correspond un complémentaire de poids $k(d-1)-h$. De fait, l'expression de $|A_{d,k}(h)|$ de la proposition 6.32 est symétrique par rapport à la valeur $\frac{k(d-1)}{2}$.
(2) Dans le cadre de cette preuve, on considère en fait le comportement de $|A_{d,k}(h)|$ pour h compris entre $-(d-1)$ et $\left\lfloor \frac{k(d-1)}{2} \right\rfloor$. En effet, si $|A_{d,k}(h)|$ croît pour $0 \leq h \leq \left\lfloor \frac{k(d-1)}{2} \right\rfloor$, alors $|A_{d,k}(h)|$ croît pour $-(d-1) \leq h \leq \left\lfloor \frac{k(d-1)}{2} \right\rfloor$, puisque $|A_{d,k}(h)| = 0$ quand $-(d-1) \leq h < 0$ et $|A_{d,k}(h)| > 0$ quand $0 \leq h \leq \left\lfloor \frac{k(d-1)}{2} \right\rfloor$.
Pour tout $d > 0$, on peut mener la récurrence suivante.

- Pour $k = 1$, la proposition est vérifiée car pour tout h compris entre 0 et $(d-1)$, $|A_{d,1}(h)| = 1$;
- Supposons que le lemme soit vrai jusqu'à $k-1$. En utilisant la proposition 6.30, on montre facilement que $|A_{d,k}(h)| - |A_{d,k}(h-1)| = |A_{d,k-1}(h)| - |A_{d,k-1}(h-d)|$.
  - Pour $h \leq \left\lfloor \frac{(k-1)(d-1)}{2} \right\rfloor$, la différence est positive par récurrence.
  - Pour $\left\lfloor \frac{(k-1)(d-1)}{2} \right\rfloor < h \leq \left\lfloor \frac{k(d-1)}{2} \right\rfloor$, $|A_{d,k-1}(h)| = |A_{d,k-1}((k-1)(d-1)-h)|$ par symétrie, et la différence est donc également positive par récurrence.

Par ci-dessus, pour tout d et tout k, $|A_{d,k}(h)|$ est croissante pour $0 \leq h \leq \left\lfloor \frac{k(d-1)}{2} \right\rfloor$. □

6.2 Évaluation de $\left|A_{d,k}^{d-1}(h)\right|$

La proposition suivante nous permet d'exprimer simplement $\left|A_{d,k}^{d-1}(h)\right|$ en fonction de $|A_{d,k}(h)|$, et donc d'appliquer au premier les résultats calculés pour ce dernier.

Proposition 6.34.

$$\left|A_{d,k}^{d-1}(h)\right| = |A_{d,k+1}(h)| - |A_{d,k}(h-(d-1))| \ .$$

Preuve. En combinant la définition de $\left|A_{d,k}^{d-1}(h)\right|$ et la proposition 6.30, on trouve directement

$$\begin{aligned} \left|A_{d,k}^{d-1}(h)\right| &= \sum_{j=0}^{(d-2)} |A_{d,k}(h-j)| \\ &= \sum_{j=0}^{(d-1)} |A_{d,k}(h-j)| - |A_{d,k}(h-(d-1))| \\ &= |A_{d,k+1}(h)| - |A_{d,k}(h-(d-1))| \ . \end{aligned}$$

□

6.3 Comportement asymptotique de $|A_{d,k}(h)|$

D'après le lemme 6.33, $|A_{d,k}(h)|$ est maximal pour $h = h_{d,k} \stackrel{\text{def}}{=} \left\lfloor \frac{k(d-1)}{2} \right\rfloor$. Soit $\tilde{h} = k(d-1) \bmod 2$. Par la proposition 6.32,

$$\frac{|A_{d,k}(h_{d,k})|}{d^k} = \frac{2}{\pi} \int_0^{\frac{\pi}{2}} \cos(\tilde{h}x) \left(\frac{\sin(dx)}{d \sin(x)} \right)^k \mathbf{d}x \ . \tag{6.2}$$

L'évaluation asymptotique de ce rapport est effectuée en utilisant la méthode de Laplace, dont on peut trouver une description détaillée dans [9].

Proposition 6.35.

$$|A_{d,k}(h_{d,k})| \underset{k\gg 1}{\approx} d^k \sqrt{\frac{6}{\pi k(d-1)(d+1)}} \ .$$

Preuve. Le théorème de Laplace dit que si deux fonctions réelles f et g continues sur $[a;b]$ sont telles que $g(t)$ atteint son maximum pour $t=a$ et que $f(a)\neq 0$, alors on peut approximer l'intégrale $\int_a^b f(t)e^{x\,g(t)}\mathbf{d}t$ par $\int_a^{a+\epsilon} f(t)e^{x\,g(t)}\mathbf{d}t$ pour $0<\epsilon\leq(b-a)$ et $x\gg 1$, car les contributions importantes à l'intégrale sont celles du voisinage de a.
Une conséquence pratique de cela est que nous pouvons modifier la borne supérieure des intégrales sans altérer leur comportement asymptotique, puisque cela n'induit que des erreurs exponentiellement faibles.
Définissons alors $f_d(t)\overset{\text{def}}{=}\cos(\tilde{h}t)$ et $G_d(t)\overset{\text{def}}{=}\frac{\sin(dt)}{d\sin(t)}$. En toute rigueur, la définition de f_d ne nous permet pas d'utiliser le théorème de Laplace, car $\tilde{h}$ dépend également de k. Cependant, $\tilde{h}$ ne prenant que les valeurs 0 ou 1, il suffit d'effectuer deux calculs distincts, selon la parité de $k(d-1)$. Comme ceux-ci mènent au même résultat, nous pouvons les écrire en une fois, en conservant $\tilde{h}$ dans les formules pour distinguer les deux cas. On peut montrer simplement que :

- $f_d(0)=1$;
- $G_d(0)=1$;
- $\forall t\in\,]0,\frac{\pi}{d}[,\ 0<G_d(t)<1$;
- $G_d(\frac{\pi}{d})=0$;
- $G_d(t)$ atteint son maximum pour $t=0$ dans $[0,\frac{\pi}{d}]$.

Soit η_d un nombre pris dans $]0,\frac{\pi}{d}[$. Par le théorème de Laplace, pour tout ϵ tel que $0<\epsilon<\eta_d$, on obtient :

$$\int_0^{\eta_d} f_d(t)\,e^{k\log(G_d(t))}\mathbf{d}t \underset{k\gg 1}{\approx} \int_0^{\epsilon} f_d(t)\,e^{k\log(G_d(t))}\mathbf{d}t \ .$$

Supposons que $\epsilon\ll 1$. Dans $[0,\epsilon]$,

$$\begin{aligned} f_d(t) &= 1+\tilde{h}\left(-\frac{1}{2}t^2+o(t^3)\right) \ , \\ G_d(t) &= 1-\frac{1}{6}(d-1)(d+1)t^2+o(t^3) \ , \\ \log(G_d(t)) &= -\frac{1}{6}(d-1)(d+1)t^2+o(t^3) \ , \end{aligned}$$

donc en ne prenant en compte que les premiers ordres des approximations polynomiales,

$$\int_0^{\eta_d} f_d(t)\,e^{k\log(G_d(t))}\mathbf{d}t \underset{k\gg 1}{\approx} \int_0^{\epsilon} e^{-\frac{k(d-1)(d+1)}{6}t^2}\mathbf{d}t \ .$$

Puisque le contenu de l'intégrale ne contribue à la valeur asymptotique de celle-ci que dans le voisinage de 0, le domaine de l'intégrale peut être étendu à l'infini en ne générant que des erreurs exponentiellement faibles, d'où

$$\int_0^{\eta_d} f_d(t)\, e^{k\log(G_d(t))}\mathbf{d}t \underset{k\gg 1}{\approx} \int_0^{+\infty} e^{-\frac{k(d-1)(d+1)}{6}t^2}\mathbf{d}t \ .$$

Pour tout réel $z>0$,

$$\int_0^{+\infty} e^{-zt^2}\mathbf{d}t = \frac{1}{2}\sqrt{\frac{\pi}{z}} \ ,$$

et donc

$$\int_0^{\eta_d} f_d(t)\, e^{k\log(G_d(t))}\mathbf{d}t \underset{k\gg 1}{\approx} \frac{1}{2}\sqrt{\frac{6\pi}{k(d-1)(d+1)}} \ .$$

Puisque, de nouveau, le contenu de l'intégrale ne contribue à la valeur asymptotique de celle-ci que dans le voisinage de 0,

$$\int_0^{\eta_d} f_d(t)\, e^{k\log(G_d(t))}\mathbf{d}t \ = \ \int_0^{\eta_d} f_d(t)\, G_d(t)^k\mathbf{d}t \underset{k\gg 1}{\approx} \int_0^{\frac{\pi}{2}} f_d(t)\, G_d(t)^k\mathbf{d}t \ ,$$

et donc, par l'équation (6.2),

$$\frac{|A_{d,k}\,(h_{d,k})|}{d^k} \underset{k\gg 1}{\approx} \sqrt{\frac{6}{\pi k(d-1)(d+1)}} \ .$$

□

6.4 Comportement asymptotique de $\left|A_{d,k}^{d-1}\,(h)\right|$

L'étude de $\left|A_{d,k}^{d-1}\,(h)\right|$ est directement inspirée de celle de $|A_{d,k}\,(h)|$. D'après le lemme 6.33, la plus grande bande de largeur $(d-1)$ est celle "centrée" autour de $\frac{k(d-1)}{2}$, c'est-à-dire telle que le plus grand poids qu'elle contient soit égal à $\left\lfloor \frac{k(d-1)}{2}+\frac{(d-1)}{2}\right\rfloor$. Nous noterons $h_{d,k}^{d-1}$ la valeur de poids qui maximise $\left|A_{d,k}^{d-1}\,(h)\right|$, c'est-à-dire $h_{d,k}^{d-1} \stackrel{\text{def}}{=} \left\lfloor \frac{(k+1)(d-1)}{2}\right\rfloor$, et $\tilde{h}=(k+1)(d-1) \bmod 2$. En appliquant successivement les propositions 6.34 et 6.32, on obtient

$$\begin{aligned}\frac{\left|A_{d,k}^{d-1}\left(h_{d,k}^{d-1}\right)\right|}{d^k} &= \frac{\left|A_{d,k+1}\left(h_{d,k}^{d-1}\right)\right| - \left|A_{d,k}\left(h_{d,k}^{d-1}-(d-1)\right)\right|}{d^k} \\ &= \frac{2}{\pi}\int_0^{\frac{\pi}{2}}\left(\cos(\tilde{h}x)\frac{\sin(dx)}{\sin(x)} - \cos(x(d-1-\tilde{h}))\right)\left(\frac{\sin(dx)}{d\sin(x)}\right)^k\mathbf{d}x \ . \qquad (6.3)\end{aligned}$$

Comme précédemment, on utilise le théorème de Laplace pour déterminer l'équivalent asymptotique de cette expression.

Proposition 6.36.

$$\frac{\left|A_{d,k}^{d-1}\left(h_{d,k}^{d-1}\right)\right|}{d^k} \underset{k\gg 1}{\approx} \sqrt{\frac{6(d-1)}{\pi k(d+1)}} \ .$$

Preuve. La démonstration est analogue à celle du théorème 6.35 ; seule change la définition de la fonction f_d. Soit

$$f_d(t) \stackrel{\text{def}}{=} \left(\cos(\tilde{h}t)\frac{\sin(dt)}{\sin(t)} - \cos(t(d-1-\tilde{h}))\right) \quad .$$

On a $f_d(0) = d-1$ et, pour ϵ tel que $0 \leq \epsilon \ll 1$,

$$\begin{aligned} f_d(t) &= \left(1-\tilde{h}\left(\frac{1}{2}t^2+o(t^3)\right)\right)\left(\frac{dt-\frac{1}{6}d^3t^3+o(t^4)}{t-\frac{1}{6}t^3+o(t^4)}\right) - \left(1-\tilde{h}\left(\frac{1}{2}(d-1)^2t^2+o(t^3)\right)\right) \\ &= (d-1) - \tilde{h}\frac{(d-1)^2-d}{2}t^2 + o(t^2) \quad . \end{aligned}$$

En ne prenant ici encore en compte que les premiers ordres des approximations polynomiales, on obtient

$$\int_0^{\eta_d} f_d(t)\, e^{k\log(G_d(t))}\mathbf{d}t \underset{k\gg 1}{\approx} \int_0^{\epsilon} (d-1)\, e^{-\frac{k(d-1)(d+1)}{6}t^2}\mathbf{d}t \quad .$$

Par le même processus que précédemment, et en utilisant l'équation (6.3), on trouve le résultat attendu. □

Remarque. Pour $d=2$, on a $\left|A_{2,k}^{2-1}(h)\right| = |A_{2,k}(h)| = \binom{k}{h}$, et les deux formules donnent effectivement l'approximation asymptotique de Stirling de $\binom{k}{\lceil \frac{k}{2} \rceil}$ par $2^k\sqrt{\frac{2}{\pi k}}$ lorsque $k \gg 1$.

6.5 Applications

Les équivalents asymptotiques déterminés ci-dessus nous permettent d'obtenir ceux de bornes que nous n'avions jusqu'à présent pu exprimer que symboliquement.

Proposition 6.37. *Notre majoration de* $\mathrm{Bd}(\mathrm{UB}(d,k))$ *est asymptotiquement équivalente à* $d^k\sqrt{\frac{6(d-1)}{\pi k(d+1)}}$ *lorsque* $k \gg 1$.

Preuve. Ceci s'obtient directement au moyen des propositions 5.17 et 6.36. □

Proposition 6.38. *Notre majoration de* $\mathrm{Cw}(\mathrm{UB}(d,k))$ *est asymptotiquement équivalente à* $(2d-1)d^k\sqrt{\frac{8(d-1)}{3\pi k(d+1)}}$ *lorsque* $k \gg 1$.

Preuve. Par les propositions 5.18, 6.35, et 6.36,

$$\begin{aligned} (d-1)\left|A_{d,k}^{d-1}\left(h_{d,k}^{d-1}\right)\right| + \tfrac{(d-1)d(d+1)}{3}|A_{d,k-1}(h_{d,k-1})| + 2 & \\ \underset{k\gg 1}{\approx} \quad (d-1)d^k\sqrt{\tfrac{6(d-1)}{\pi k(d+1)}} + \tfrac{(d-1)d(d+1)}{3}d^{k-1}\sqrt{\tfrac{6}{(d-1)(d+1)\pi(k-1)}} + 2 & \\ \underset{k\gg 1}{\approx} \quad d^k\left(\sqrt{\tfrac{6(d-1)^3}{\pi k(d+1)}} + \sqrt{\tfrac{2(d-1)(d+1)}{3\pi k}}\right) & \\ \underset{k\gg 1}{\approx} \quad d^k(2d-1)\sqrt{\tfrac{8(d-1)}{3(d+1)\pi k}} \quad . & \end{aligned}$$

□

Chapitre 7

Conclusion

7.1 Récapitulatif des résultats obtenus

Le tableau suivant reprend les résultats des sections 5.3 et 6.5.

Graphe	Largeur de bande		Largeur de coupe	
	minoration	majoration	minoration	majoration
H(k)	$\sum_{j=0}^{k-1} \binom{j}{\lceil \frac{j}{2} \rceil}$ [47]$^{\star\sharp}$		$\lfloor \frac{1}{3} 2^{k+1} \rfloor$ [70]	
$M_2(d_0, d_1)$	d_0 [20]		$d_0 + 1 - (d_0 \bmod 2)$	$d_0 + 1$
UB$(2,k)$	$\frac{2^k-1}{k}$ *	$\binom{k}{\lceil \frac{k}{2} \rceil}$ $^\sharp$	$\frac{1}{2} \cdot \frac{2^k}{k}$	$2\binom{k}{\lceil \frac{k}{2} \rceil} + 2$ $^\sharp$
UB(d,k)	$\frac{d^k-1}{k}$ *	$\left\lvert A_{d,k}^{d-1}\left(h_{d,k}^{d-1}\right)\right\rvert$ $^\diamond$	–	$(d-1)\left\lvert A_{d,k}^{d-1}\left(h_{d,k}^{d-1}\right)\right\rvert + 2 +$ $\frac{(d-1)d(d+1)}{3}\lvert A_{d,k-1}\left(h_{d,k-1}\right)\rvert$ $^\triangle$
SE(k)	$\frac{2^k-1}{2k-1}$ *	$2\binom{k-2}{\lceil \frac{k-2}{2} \rceil}$ $^\sharp$	$\frac{1}{2} \cdot \frac{2^k}{k}$	$\binom{k}{\lceil \frac{k}{2} \rceil} + 2$ $^\sharp$
FFT(k)	2^{k-1} *	$3 \cdot 2^{k-1}$	– †	$2^{k+1} - 2$ †
BF(k)	$\lfloor \frac{8}{3} \cdot 2^{k-2} \rfloor$ *	$9 \cdot 2^{k-2}$	– ‡	$3 \cdot 2^k$ ‡
CCC(k)	$\lfloor \frac{8}{5} \cdot 2^{k-2} \rfloor$ *	$9 \cdot 2^{k-2}$	$\lfloor \frac{1}{5} \cdot 2^{k+1} \rfloor$	$2^{k+1} + 1$

* Bd(H(k)) appartient à $\Theta\left(\left(\binom{k}{\lceil \frac{k}{2} \rceil}\right)\right)$ [55].

$^\sharp$ $\binom{k}{\lceil \frac{k}{2} \rceil}$ appartient à $\Theta\left(2^k \sqrt{\frac{2}{k\pi}}\right)$.

* en appliquant le théorème 3.2.

$^\diamond$ asymptotiquement équivalent à $d^k \sqrt{\frac{6(d-1)}{\pi k(d+1)}}$.

$^\triangle$ asymptotiquement équivalent à $(2d-1)d^k \sqrt{\frac{8(d-1)}{3\pi k(d+1)}}$.

† Cw(FFT(k)) appartient à $\Theta(2^k)$.

‡ Cw(BF(k)) appartient à $\Theta(2^k)$.

7.2 Conclusion

Nous avons présenté dans cette partie une expression de la largeur de bande d'un graphe à partir de celle de ses graphes quotients et de ses sous-graphes induits, et avons fait de même pour la largeur de coupe. Nous avons également donné une minoration des largeurs de bande et de coupe d'un graphe, par ses bissections sommet et arête respectivement.

La méthode de quotientement est bien adaptée aux graphes qui peuvent être quotientés, éventuellement récursivement, en graphes dont les largeurs de bande et de coupe sont connues ou majorées finement. Cependant, le choix des graphes quotients n'est pas simple, puisque c'est sur lui que se reporte la complexité du problème. Les quotientements doivent être simples pour que leurs paramètres puissent être calculés, mais aussi choisis judicieusement afin de capturer dans le graphe quotient les propriétés topologiques des graphes étudiés.
Quand la structure des graphes est bien connue, il est parfois profitable d'utiliser la numérotation induite par le partitionnement afin de calculer directement et efficacement les majorations.
En ce sens, nos résultats doivent plus être vus comme un cadre formel et méthodologique de travail, que comme des théorèmes directement applicables.

Conjointement aux résultats présentés ici, Barth [5] a obtenu par cette méthode des majorations et minorations des largeurs de bande et de coupe des graphes FFT d-aire, "*Butterfly*" d-aire, et grille d'arbres binaire, ainsi que du graphe grille d'arbres d-aire qu'il a défini.

Les équivalents asymptotiques calculés dans la section précédente peuvent être appliqués à d'autres graphes sur alphabets, tel le graphe "*Shuffle-Exchange*" d-aire.
Plus généralement, on peut envisager l'utilisation du théorème de Laplace pour déterminer les équivalents asymptotiques de quantités discrètes, au moins lorsqu'elles sont de structure polynomiale.

Partie II

Placement statique

Chapitre 8

Présentation du problème

Parmi les moyens de délivrer aux utilisateurs une puissance informatique croissante, le parallélisme est une voie de recherche prometteuse. Une des premières classifications des architectures parallèles, due à Flynn [33], distingue deux grandes classes de machines :

- les machines SIMD ("*Single Instruction, Multiple Data*"), dans lesquelles les instructions d'un programme sont exécutées simultanément et de manière synchrone sur tous les nœuds qui les composent. Elles possèdent un frontal qui décode les instructions parallèles et les diffuse sur les unités d'exécution des nœuds, et exécute localement les instructions purement scalaires ;

- les machines MIMD ("*Multiple Instruction, Multiple Data*"), constituées de processeurs complets indépendants possédant chacun leurs unités de décodage et d'exécution propres.

La machine MIMD à mémoire distribuée et à communication par passage de messages est le type de machine parallèle le plus courant, en ce qu'elle permet simplement d'implanter le modèle de programmation par processus communicants. Celui-ci conduit à exprimer les algorithmes parallèles sous forme de processus séquentiels à mémoire privée communicant par échange de messages à travers des canaux de communication [52]. Cependant, la versatilité de ces machines a pour contrepartie une assez grande complexité de mise en œuvre si l'on souhaite en exploiter au mieux les capacités.
En particulier, l'exécution efficace d'un programme parallèle sur une telle machine nécessite que les processus communicants constituant le programme soient chargés sur les processeurs de la machine de manière à minimiser le temps total d'exécution. Pour cela, il faut minimiser le temps passé à réaliser les communications induites par la parallélisation, tout en équilibrant dans le temps la charge de calcul attribuée à chaque processeur afin d'exploiter pleinement le parallélisme.
Cette opération est appelée *ordonnancement* si l'on considère les dépendances logiques et temporelles entre processus, et *placement* si l'on suppose que tous les processus coexistent simultanément sur la machine parallèle pendant toute la durée d'exécution du programme. Un placement est dit *statique* s'il a lieu avant l'exécution du programme et n'est jamais remis en cause, et *dynamique* si les processus peuvent être déplacés en cours d'exécution.

Le mode d'expression du parallélisme par processus communicants n'est pas le seul qui existe. Le parallélisme orienté par les données, qui est mis en application dans plusieurs langages [1, 51], exprime un algorithme parallèle sous la forme d'un algorithme séquentiel auquel sont ajoutées des directives de partition des données. Les compilateurs de ces langages utilisent ces informations pour générer un ensemble de processus communicants, qui se partagent les données en fonction de ces directives. Le code séquentiel propre à chaque processus est construit à partir de l'algorithme séquentiel initial par ajout des instructions de communication nécessaires à l'échange des données entre processus.
Bien que le découpage des données soit à l'heure actuelle souvent très régulier, le graphe de communication généré peut être complexe, ce qui nécessite toujours de recourir à un logiciel de placement pour optimiser les communications et équilibrer la charge des processeurs de l'architecture cible.

Dans la suite de cette partie, nous nous intéresserons uniquement au placement statique de processus.

8.1 Placement statique et fonctions de coût

Le calcul d'un placement statique efficace nécessite une connaissance à priori du comportement dynamique de la machine cible vis-à-vis du programme qui y est exécuté. On doit savoir pour cela :

- modéliser les interactions entre le programme et la machine, en isolant les critères déterminants de la qualité d'un placement ;
- extraire du programme parallèle l'information nécessaire à son placement en fonction des critères définis ;
- trouver le placement du programme qui optimise ces critères.

La modélisation de la qualité d'un placement s'effectue habituellement au moyen de fonctions de coût, qui déterminent par leur contenu les caractéristiques du placement optimal que l'on désire obtenir. Les principaux critères cités dans la littérature [3, 67, 79, 89, et leurs références] et qui peuvent intervenir dans ces fonctions sont :

- le temps total d'utilisation de la machine, c'est-à-dire la durée d'exécution du processus le plus long, qui conditionne la terminaison du programme parallèle ;
- l'équilibrage de la charge de calcul entre les différents processeurs. Le temps d'exécution du programme y est directement lié, puisque, si l'on ne tient pas compte des communications, une charge équirépartie assure un temps d'exécution minimal ;
- l'équilibrage de la charge de communication entrant et sortant des processeurs ;
- l'équilibrage de la charge de communication interne aux processeurs, c'est-à-dire de la communication entre processus placés sur les mêmes processeurs ;

- l'équilibrage de la charge de communication transitant sur les liens physiques de la machine ;
- la minimisation du coût total de communication, pris comme la somme des coûts des canaux de communication du programme parallèle pondérés par une fonction dépendant du placement. Celle-ci peut tenir compte de la distance de plus courts chemins entre les processeurs sur lesquels sont placés les processus extrémités du canal.

Dans la majorité des cas, les fonctions de coût combinent plusieurs de ces critères par sommation, en les pondérant par des paramètres dépendant éventuellement des caractéristiques de la machine cible. Ainsi, si une machine calcule rapidement mais communique lentement, on pénalisera plus la communication que le coût de calcul, afin de privilégier les placements minimisant la communication par rapport aux autres.

L'inconvénient majeur des fonctions de coût unifiées est l'existence de ces coefficients de pondération. Outre le fait qu'ils doivent être mesurés pour chaque machine cible, l'amalgame de quantités de nature différente, même si on les manipule comme des nombres sans dimension (écarts relatifs à la moyenne, par exemple), n'est en pratique pas facile à réaliser.
Certains auteurs préfèrent donc dissocier la partie calcul et la partie communication. Il s'agit alors de minimiser une fonction de coût de communication, tout en conservant l'équilibrage de charge dans une tolérance prédéfinie [25]. Ceci pourrait se traduire dans une fonction de coût unique par le fait de rajouter à la fonction de communication un terme de valeur nulle si l'équilibrage est réalisé, et infinie dans le cas contraire. Cependant, certains algorithmes de placement tirent implicitement parti de la continuité de la fonction de coût pour en rechercher les minima locaux (les algorithmes de recuit simulé et génétiques en particulier), et employer une fonction discontinue remettrait en cause leur efficacité. La hiérarchisation des fonctions de calcul et de communication n'apparaît donc principalement que dans les algorithmes de graphes [30, 53].

8.2 Les algorithmes de placement statique

La recherche du placement optimal d'un programme à s processus sur une machine à t processeurs nécessite en théorie l'évaluation des t^s placements possibles. Bien que les dimensions du problème puissent être réduites du fait des contraintes s'appliquant aux placements effectivement réalisables, cette recherche reste NP-complète dans le cas général [36]. L'alternative est donc soit d'effectuer une recherche exhaustive en risquant l'explosion combinatoire, soit de calculer au moyen d'heuristiques un placement acceptable en un temps polynomial.

8.2.1 Les algorithmes exacts

Les algorithmes exacts permettent toujours de trouver la ou les solutions de coût minimal. Du fait de sa NP-complétude, la recherche d'une solution optimale dans le cas général nécessite l'exploration systématique de l'espace des solutions. Pour ce faire, on se base habituellement sur un arbre de recherche dont les nœuds correspondent à des sous-ensembles de solutions, le parcours

des branches de la racine vers les feuilles représentant alors le fait de lier l'une après l'autre les variables du problème jusqu'à l'obtension de solutions complètes.

Les méthodes de type "séparation et évaluation" ("*branch and bound*") permettent d'éviter de parcourir toutes les branches, en "élaguant" les sous-arbres menant à des solutions toutes moins bonnes qu'une solution donnée. Pour cela, on associe à chaque sommet de l'arbre une minoration, aussi précise que possible, de la fonction de coût calculée sur les variables du sous-problème correspondant à ce sommet[†] ; par construction, cette minoration est telle que la valeur d'un sommet est toujours supérieure à celle de son père. Il en découle que si la valeur d'un sommet est supérieure au coût d'une solution déjà trouvée, il est inutile de parcourir le sous-arbre correspondant, puisque ses feuilles ne seront pas de meilleures solutions. L'algorithme consiste donc, partant d'une solution initiale et de la racine de l'arbre, à parcourir celui-ci, en mettant à jour la solution courante dès qu'une solution de plus faible coût est trouvée, et en évitant les branches menant à des sommets de valeur supérieure au coût de la solution courante.
Plusieurs variantes de l'algorithme existent, selon que l'on explore l'arbre en profondeur, par niveaux, ou par ordre croissant des valeurs des fils non encore traités (le parcours de l'arbre est alors analogue à celui effectué par un algorithme de type A* [84, 86]).
Bien que la complexité de cet algorithme soit dans le pire des cas identique à celle d'une énumération explicite, il est en moyenne plus efficace.

Une autre approche, que nous ne détaillerons pas ici, est celle de la programmation dynamique. On représente alors le problème comme la minimisation avec contraintes d'une fonction de coût quadratique en variables $\{0, 1\}$ [45, 82].

Il est à noter que, dans certains cas, le problème de placement peut être réduit à un problème polynomial. Stone [88] a ainsi montré comment un placement optimal peut être obtenu en temps polynomial pour une machine bi-processeurs. Ce résultat a été partiellement étendu à un nombre quelconque de processeurs par Lo [63], qui donne un algorithme polynomial pouvant conduire à une solution optimale prouvée, mais pouvant aussi échouer. Dans le même article, elle fournit également un algorithme polynomial optimal dans le cas de processeurs identiques ne supportant qu'au plus deux processus chacun.

Stone modélise le placement à réaliser par un graphe unique, dont l'ensemble des sommets contient à la fois les processus et les deux processeurs. Une arête entre deux processus est pondérée par le coût (volume) des données qu'ils échangent, et une arête entre un processus et un processeur est valuée par le coût du placement du processus sur l'autre processeur. En prenant les deux processeurs comme source et puits de capacité infinie, il est alors possible, au moyen d'un algorithme de flot maximal, de calculer en temps polynomial une coupe de coût minimal dont chaque partie contient un unique processeur. Cette coupe correspond au placement optimal recherché, que l'on

[†] Prendre en compte dans la minoration l'information associée aux variables libres d'un problème demande souvent une connaissance approfondie de celui-ci. Lorsque la minoration ne prend en compte que les variables liées, on parle plutôt de "chaînage arrière" ("*backtracking*").

construit en affectant à chaque processeur les processus situés dans la même partie que lui.

Pour étendre cette méthode, Lo utilise un graphe analogue à celui décrit ci-dessus, mais contenant plus de deux sommets représentant des processeurs. Pour déterminer les processus placés sur chaque processeur, on applique l'algorithme de flot maximal sur le graphe, en prenant comme source le processeur considéré, le puits de capacité infinie étant construit par fusion de tous les autres processeurs du graphe en un sommet unique. Les processus n'ayant été sélectionnés que par un unique processeur lui sont attribués, et sont éliminés du graphe. On applique alors à nouveau la méthode sur le sous-graphe obtenu. Lorsque le sous-graphe ne contient plus de processus, le placement calculé est optimal ; si une itération termine sans qu'un seul processus ait été placé, l'algorithme échoue.

Le deuxième algorithme proposé par Lo est totalement différent du précédent, car basé sur l'utilisation d'un couplage parfait maximal. Il n'est valide qui si le nombre de processus est inférieur ou égal au double du nombre de processeurs, et si chaque processeur n'accepte qu'au plus deux processus. Le graphe considéré est ici le graphe de communication entre processus, auquel on applique un algorithme polynomial de calcul de couplage parfait maximal. Les extrémités des arêtes obtenues correspondent aux processus communiquant le plus entre eux, qui sont donc placés sur le même processeur. Les sommets isolés restants sont alors placés par paires sur les processeurs non encore utilisés.

8.2.2 Les heuristiques

De nombreuses heuristiques ont été proposées afin de trouver des solutions sous-optimales en un temps raisonnable [12, 13, 25, 65, 68]. Elles sont issues de différents domaines, tels la théorie des graphes (recherche d'homomorphismes faibles, partitions de coupe minimale, groupement de processus), les méthodes d'optimisation combinatoire liées à la simulation de phénomènes physiques (recuit simulé et algorithmes génétiques), l'étude spectrale des matrices, l'utilisation de l'information géométrique des graphes (méthodes inertielles et rectilinéaires), etc...

Les classifications des heuristiques [3, 67, 79] distinguent entre les algorithmes gloutons qui, initialisés avec une solution partielle, construisent une solution complète sans jamais remettre en cause un choix effectué, et les algorithmes itératifs, qui procèdent par raffinement d'une solution réalisable courante. Notons cependant que la limite n'est pas toujours nette, principalement avec les heuristiques de graphes : certains algorithmes peuvent être considérés comme itératifs au sens où ils remettent en cause des choix effectués, et gloutons parce que leur comportement est totalement déterministe.
L'objet de notre étude nous suggère un critère de classification supplémentaire, selon que la résolution du problème de placement est envisagée globalement ou comme un ensemble de sous-problèmes plus ou moins dépendants les uns des autres.

8.2.2.1 L'approche globale

Elle consiste à considérer le placement dans sa globalité, en manipulant simultanément l'ensemble des variables du problème. C'est le cas des méthodes itératives de type recuit simulé, des algorithmes génétiques, et de certaines méthodes de graphes [12] ou basées sur la géométrie du problème [71]. L'inconvénient de ces approches est que la taille de l'espace des solutions augmente exponentiellement par rapport à la taille du problème, ce qui rend son exploration très coûteuse.

8.2.2.2 L'approche "diviser pour résoudre"

Afin de placer des graphes de plus en plus gros sur des machines parallèles de taille croissante, de nombreux auteurs se sont tournés vers l'approche "diviser pour résoudre", conceptuellement plus simple que de multipartitionner les graphes en une seule fois pour obtenir immédiatement le nombre de parties voulu. Elle consiste à séparer le graphe à placer en plusieurs parties (habituellement deux, plus rarement quatre ou huit), qui sont affectées à un nombre équivalent de parties de la machine cible. Ce processus est récursivement appliqué aux sous-graphes et sous-machines ainsi obtenus, jusqu'à ce que les sous-machines soient réduites à un unique processeur, sur lequel sont placés les processus du sous-graphe correspondant.
Remarquons que c'est une approche gloutonne, puisque les choix effectués ne pourront être remis en cause. C'est cependant une "bonne" approche gloutonne, parce que les choix initiaux ne sont pas très informants, et que leurs effets peuvent éventuellement être atténués par des choix ultérieurs. Par exemple, deux processus séparés sur deux sous-machines différentes au début de la récursion peuvent cependant être maintenus à petite distance l'un de l'autre si les séparations ultérieures les affectent à des sous-machines toujours proches les unes des autres.

L'obtension de bons placements nécessite que les bipartitionnements successifs concourent à la minimisation de la fonction de coût choisie. Il s'agit donc de bipartitionnements avec contraintes, sur lesquels se reporte la complexité du problème de placement.
Par exemple, l'un des objectifs communs à toutes les méthodes de bipartitionnement est de favoriser la minimisation de la coupe entre les parties qu'elles génèrent, car cela revient à privilégier la localité des communications sur la machine parallèle. En effet, plus la communication entre deux processus est intensive, plus ceux-ci auront tendance à se trouver placés à proximité l'un de l'autre, car les séparer reviendrait à augmenter de beaucoup la coupe correspondante. À cet objectif de minimisation de la communication s'ajoute toujours une contrainte d'équilibrage des cardinaux des sous-graphes, afin que le coût de calcul soit équitablement réparti entre toutes les sous-machines, et donc tous les processeurs.
Ces seules contraintes de minimisation de la communication et d'équilibrage de la charge de calcul suffisent à rendre le problème de bipartitionnement NP-complet. En effet, si l'on ne s'attache qu'à minimiser strictement la coupe entre les parties tout en assurant l'égalité à un près de leurs cardinaux, on se ramène au problème du calcul de la bissection arête d'un graphe, qui est NP-complet dans le cas général [37].

Dans le cadre de l'approche "diviser pour résoudre", de nombreux auteurs se sont donc consacrés à l'étude d'heuristiques de bipartitionnement efficaces [11, 14, 25, 44, 48, 49, 54, 92, 93, 95].

L'une des méthodes les plus couramment utilisées est la bissection spectrale. Celle-ci, issue des travaux de Donath et Hoffman [23, 24], se base sur les propriétés de la matrice de Laplace du graphe à bissecter. La matrice de Laplace d'un graphe non-valué sans cycles S est la matrice carrée $Q = D - A$, où A est la matrice d'adjacence du graphe, et D la matrice diagonale telle que $D_{i,i} = \delta(v_i)$, pour $v_i \in V(S)$.
Pothen, Simon, et Liou [80], reprenant les travaux de Fiedler [31, 32], ont montré que lorsque S est connexe, les composantes x_i du vecteur propre associé à la deuxième plus petite valeur propre λ_2 de Q (dit vecteur de Fiedler) permettent de définir une bissection de S. Pour cela, ils séparent dans les deux sous-graphes d'une part les sommets de plus petits x_i, et d'autre part les sommets de plus grands x_i, les résultats de Fiedler assurant que les deux sous-graphes obtenus seront eux aussi connexes.
Cette méthode est fortement liée à la recherche des modes propres de vibration de l'objet maillé. Ainsi, dans le cas des graphes planaires, la coupe donnée par le vecteur de Fiedler correspond au premier ventre de vibration de la membrane imaginaire dont le maillage homogène donnerait le graphe considéré.
L'inconvénient des méthodes de bissection spectrale utilisées actuellement dans les algorithmes de placement est qu'elles ne tiennent pas compte de la topologie de la machine cible. L'objectif d'une bissection est en effet uniquement de minimiser la coupe du sous-graphe qu'elle bipartitionne (c'est-à-dire le nombre d'arêtes du sous-graphe ayant leurs extrémités dans les deux parties). De ce point de vue, la bissection spectrale récursive revient à placer sur le graphe complet (équivalent à un réseau d'interconnexion de type bus). Les auteurs qui utilisent la bissection spectrale traitent le problème de deux manières :

- soit ils considèrent que, grâce au routage de type "*cut-through*", la communication est identique entre toute paire de processeurs indépendamment de la topologie considérée [4], et n'en tiennent donc pas compte ;
- soit ils appliquent à l'issue de la bissection spectrale un post-traitement heuristique itératif destiné à optimiser la bissection en fonction de la topologie de la machine cible [44, 48].

Bien qu'appliquée au placement, la méthode de bissection spectrale récursive est plus orientée vers la décomposition de domaines ou la renumérotation de graphes utilisée dans la résolution par dissections emboîtées de grands systèmes linéaires creux [41, 61, 81].

L'autre classe de méthodes de bipartitionnement la plus utilisée est constituée des heuristiques de graphes de type Kernighan-Lin et Fiduccia-Mattheyses [53, 30], que nous étudions de façon détaillée en section 11.4.4. Ces algorithmes partent d'une solution réalisable de coût quelconque, qu'ils améliorent progressivement en construisant des séquences d'échange ou de déplacement de sommets conduisant à une nouvelle solution réalisable de plus faible coût. Afin de ne pas être piégés

dans des minima locaux de la fonction de coût, ils tolèrent dans ces séquences des mouvements de sommets conduisant à des solutions intermédiaires de coût plus élevé que celui de la solution de départ, si cela permet des gains ultérieurs plus importants.

D'autres heuristiques de graphes ont également été proposées, telles les méthodes de type GRASP (pour "*Greedy Randomized Adaptative Search Procedure*") [54]. Celles-ci consistent à itérer un grand nombre de fois une routine gloutonne de calcul d'une solution réalisable, en conservant à chaque fois la meilleure solution trouvée. Le calcul de la solution courante s'effectue en construisant une solution initiale au moyen d'un algorithme glouton, puis en lui appliquant un second algorithme glouton d'optimisation locale. Afin d'explorer au fil des itérations le plus de configuration initiales et d'états finaux possibles, les algorithmes gloutons de construction et d'optimisation utilisent explicitement des variables aléatoires pour influencer leur comportement.
Ces méthodes se démarquent des méthodes de recherche purement probabilistes en ce que les algorithmes gloutons cherchent toujours à minimiser la fonction de coût, et diffèrent des heuristiques de type Kernighan-Lin en ce que les algorithmes d'optimisation sont beaucoup plus simples, les fonctionnalités de sortie des minima locaux étant ici remplacées par la multiplicité des essais effectués.

Les algorithmes de calcul de flots maximaux ont également été employés pour déterminer la bissection arête de graphes. L'algorithme de Bui, Chaudhuri, Leighton, et Sipser [14] construit, pour toute paire de sommets du graphe considéré, un graphe contracté sur lequel est appliqué l'algorithme de flot maximal. Les fréquences d'apparition des arêtes du graphe initial dans les coupes minimales ainsi obtenues peuvent permettre de construire une bissection optimale prouvée du graphe, mais l'algorithme peut également échouer.
Cet algorithme donne d'excellents résultats pour des graphes réguliers de petits degrés et de petite bissection, mais est largement dépassé par les heuristiques de type Kernighan-Lin dès que le degré des graphes dépasse 4.

Indépendamment des méthodes de bipartitionnement utilisées, l'approche "diviser pour résoudre" à permis d'obtenir de très bons résultats. Ce n'est que récemment que Simon et Teng [85] ont justifié théoriquement la capacité des algorithmes de bipartitionnement récursif à fournir de bons placements pour les graphes les plus couramment utilisés. Le résultat majeur de leur article est que, lorsqu'on place un graphe de maillage d-dimensionnel[‡] de type éléments finis (à degré borné) S sur une architecture cible T par bipartitionnements récursifs, le rapport entre le nombre d'arêtes liant des processus placés sur des processeurs différents (c'est-à-dire le cardinal de l'union des coupes successives) et $|E(S)|$ appartient à $O\left(\left(\frac{|V(T)|}{|V(S)|}\right)^{\frac{1}{d}}\right)$. Ils montrent ainsi que les algorithmes parallèles associant un traitement élémentaire à chaque sommet et dont le schéma de communication suit la topologie de ces graphes sont adaptés au parallélisme massif.
En effet, pour un graphe donné, l'augmentation linéaire de la taille de la machine cible n'induit

[‡]La preuve de Simon et Teng est fondée sur l'existence de théorèmes de séparation pour les familles de graphes étudiés. Leur résultat s'applique donc également aux graphes planaires, à genre borné, à mineur borné, et en général à tous les graphes possédant de "bons" théorèmes de séparation.

qu'une augmentation en racine $d^{ième}$ de la taille des coupes (c'est-à-dire du volume de communication à échanger entre processeurs), alors que la capacité de communication des machines évolue habituellement linéairement avec leur taille. Ces programmes parallèles peuvent donc utiliser des machines plus grosses sans risquer de saturer leur réseau de communication. L'efficacité du programme vis-à-vis du nombre de processeurs ne dépend que de la taille des parties, qui détermine le rapport entre temps de calcul et temps de communication.

8.3 Notre travail

Beaucoup de travaux réalisés sur le placement portant sur des méthodes et des topologies spécifiques, nous avons essayé d'aborder le problème le plus globalement possible.
En utilisant l'approche "diviser pour résoudre", nous avons défini la structure d'un programme de placement fonctionnant par bipartitionnement récursif conjoint des graphes valués modélisant le programme à placer et la machine cible. Ce programme peut accepter n'importe quelles méthodes de bipartitionnement et, plus originalement, peut placer sur tout type de topologie cible.

Pour que notre algorithme donne de bons résultats, il faut que les bipartitionnements successifs du graphe modélisant l'architecture cible respectent certaines propriétés, telles la minimisation de la coupe et la connexité des sous-parties.
Pour les topologies classiques (grilles, hypercubes, arbres, bus), les décompositions se font de manière algorithmique, au moyen de routines spécifiques à chaque famille. Dans le cas de graphes cibles quelconques (pouvant correspondre à des sous-parties irrégulières de machines parallèles), nous montrons comment calculer la décomposition au moyen de notre programme. Ceci se fait plaçant le graphe représentant la machine cible sur le graphe complet.

La première heuristique de bipartitionnement que nous avons étudiée est celle de Fiduccia et Mattheyses. Pour pouvoir appliquer celle-ci à des graphes valués, nous modifions les structures de données qu'elle manipule, en construisant des tableaux à indexation logarithmique plutôt que linéaire. Ce choix est validé expérimentalement.

La structure de notre programme nous permet d'appeler successivement plusieurs méthodes différentes pour effectuer un bipartitionnement. Nous nous servons de cette fonctionnalité pour expérimenter plusieurs algorithmes différents en pré- et post-traitement de l'heuristique de Fiduccia et Mattheyses, dont la sensibilité aux conditions initiales est bien connue.
Nous montrons que l'action de ces pré- et post-traitements diffère selon les types de graphes de processus. Ainsi, dans le cas des graphes maillés d'éléments finis, la qualité des solutions dépend peu du bipartitionnement initial, mais le calcul d'une bipartition initiale judicieuse par un algorithme adapté peut faire diminuer de plus de 30 pourcents le temps de calcul du placement. La méthode que nous utilisons pour cela est une modification de l'algorithme de Gibbs, Poole, et Stockmeyer.

Nous comparons les résultats de notre méthode à ceux obtenus par différents programmes. Les résultats obtenus avec notre programme sont équivalents en moyenne à ceux obtenus avec les meilleurs algorithmes de bissection spectrale, en un temps comparable à celui des méthodes les plus rapides. Le rapport entre la qualité de nos résultats et le temps de calcul est donc très intéressant.

La description de notre méthode et des améliorations apportées à l'heuristique de Fiduccia et Mattheyses a fait l'objet d'une publication [75]. Un deuxième article est en préparation, concernant la décomposition par placement des architectures cibles.

Chapitre 9

Définitions et notations

9.1 Graphe source et graphe cible

On modélise le programme parallèle à placer par un graphe simple non-orienté valué, appelé *graphe source* et noté usuellement S, dont les sommets représentent les processus du programme parallèle, et les arêtes les canaux de communication entre processus communicants. Les valuations sommet et arête associent à tout sommet v_S et toute arête e_S de S des nombres entiers $w(v_S)$ et $w(e_S)$, qui représentent des estimations du coût de calcul du processus correspondant et de la quantité de communication transmise sur le canal, respectivement.

La machine cible est également modélisée par un graphe simple non-orienté valué, appelé *graphe cible* et noté T, qui décrit la topologie de la machine. Ses sommets représentent les processeurs, et ses arêtes les liens de communication directs existant entre les processeurs (on transforme donc un bus en une clique). On associe aux sommets v_T et arêtes e_T de T des valuations entières $w(v_T)$ et $w(e_T)$ estimant respectivement la puissance de traitement des processeurs et le coût de traversée des liens (assimilable à l'inverse de la bande passante).
Ces unités de mesure ont été choisies de manière à pouvoir sommer la puissance de traitement disponible sur un ensemble de processeurs et évaluer par sommation le coût de transfert d'un message le long d'un chemin.

9.2 Placement statique

On définit un placement de S sur T par un couple $\xi_{S,T} = (\tau_{S,T}, \rho_{S,T})$ de S dans T, où $\tau_{S,T}$ est une application de $V(S)$ dans $V(T)$, et $\rho_{S,T}$ est une application associant à toute arête $\{v'_S, v''_S\}$ de $E(S)$ une chaîne reliant $\tau_{S,T}(v'_S)$ à $\tau_{S,T}(v''_S)$. Cette définition diffère de celle du plongement en ce que les applications $\tau_{S,T}$ et $\rho_{S,T}$ ne sont pas nécessairement injectives.
$\tau_{S,T}(v_S) = v_T$ si le processus v_S est placé sur le processeur v_T, et $\rho_{S,T}(e_S) = \{e_T^1, e_T^2, \ldots, e_T^n\}$ si le

canal de communication e_S est routé à travers les liens e_T^1, e_T^2, ..., e_T^n de la machine cible. La dilatation $|\rho_{S,T}(e_S)|$ de l'arête e_S par $\xi_{S,T}$ est le nombre de liens utilisés dans T pour router le canal e_S.

Un placement $\xi_{S,T} = (\tau_{S,T}, \rho_{S,T})$ de S dans T est dit *de plus courts chemins* si, pour toute arête $\{v'_S, v''_S\}$ de $E(S)$, $|\rho_{S,T}(\{v'_S, v''_S\})|$ est égal à la distance dans T entre $\tau_{S,T}(v'_S)$ et $\tau_{S,T}(v''_S)$.

Soit $\xi_{S,T} = (\tau_{S,T}, \rho_{S,T})$ un placement de S dans T. On définit la *charge* des sommets et arêtes de T relativement à $\xi_{S,T}$ par

$$c(v_T) = \sum_{\substack{v_S \in V(S) \\ \tau_{S,T}(v_S) = v_T}} w(v_S) \ , \qquad \text{et} \qquad c(e_T) = \sum_{\substack{e_S \in E(S) \\ e_T \in \rho_{S,T}(e_S)}} w(e_S) \ .$$

$c(v_T)$ et $c(e_T)$ représentent respectivement les charges de calcul et de communication supportées par le processeur v_T et l'arête e_T.

9.3 Complexité et comportement expérimental

Soit $ALGO$ un algorithme. On note $\mathcal{C}_E$(ALGO) la complexité maximale en espace de cet algorithme. Elle représente la plus grande taille mémoire dont peut avoir besoin l'algorithme au cours de son exécution, cet espace mémoire étant libéré à la fin de celle-ci.
De manière similaire, on note $\mathcal{C}_T$(ALGO) sa complexité maximale en temps.

L'intérêt de certaines heuristiques est que leur comportement effectif peut se situer plusieurs ordres de grandeur en dessous de leur complexité maximale, sans que cela puisse cependant être mathématiquement démontré.
On note donc $\mathcal{C}'_E$(ALGO) le comportement en espace de l'algorithme $ALGO$, et $\mathcal{C}'_T$(ALGO) son comportement en temps.
À la différence de la complexité, qui est un résultat théorique prouvé, le comportement d'un algorithme est un résultat empirique, uniquement fondé sur l'expérimentation. Il est cependant fiable, car observé de manière quasiment systématique sur un grand nombre de cas réels, et justifié par une argumentation qualitative.

Chapitre 10

L'algorithme de placement par bipartitionnement récursif conjoint

10.1 Schéma général de l'algorithme

La méthode que nous proposons, de type *diviser pour résoudre*, procède par allocation récursive de sous-ensembles de processus à des sous-ensembles de processeurs, jusqu'à ce que les sous-ensembles de processeurs soient réduits à un seul élément, ou que les sous-ensembles de processus soient vides. A chaque étape, l'algorithme effectue le bipartitionnement du sous-ensemble de processeurs, aussi appelé *domaine*, en deux sous-domaines disjoints, et appelle un algorithme de *bipartitionnement de graphe* pour placer le sous-ensemble de processus sur les deux sous-domaines (voir figure 10.1). On a ainsi le pseudo-programme suivant :

```
placement (D, P)
Ensemble_De_Processeurs  D;
Ensemble_De_Processus    P;
{
  Ensemble_De_Processeurs  D0, D1;
  Ensemble_De_Processus    P0, P1;

  si (|P| == 0)                                /* Si rien a faire */
     retour;
  si (|D| == 1) {                              /* Si D ne contient qu'un processeur     */
    resultat (D, P);                           /* Tous les processus de P y sont places */
    retour;
  }

  (D0, D1) = bipartition_processeur (D);       /* Effectue la bipartition des processeurs */
  (P0, P1) = bipartition_processus  (P, D0, D1); /* Effectue la bipartition des processus   */
  placement (D0, P0);                          /* Effectue la recursion                   */
  placement (D1, P1);
}
```

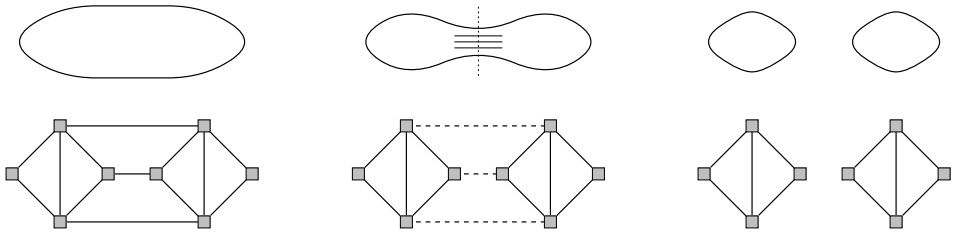


Figure 10.1: Bipartitionnement conjoint des graphes de processus et d'architecture, qui divise un problème de placement en deux sous-problèmes de plus petite taille.

L'association d'un sous-domaine à chaque processus définit un *placement partiel* du graphe des processus. Au fur et à mesure des bipartitionnements, la taille des sous-domaines sur lesquels sont placés les processus diminue, jusqu'à donner un placement complet lorsque tous les sous-domaines sont de taille 1.

L'algorithme ci-dessus repose sur la capacité à définir quatre objets principaux :

- une structure de *domaine*, qui représente un ensemble de processeurs ;
- une *fonction de bipartitionnement de domaine* qui, étant donné un domaine, le bipartitionne en deux sous-domaines disjoints ;
- une *fonction de bipartitionnement de processus* qui, étant donnés un domaine, ses deux sous-domaines, et un ensemble de processus, bipartitionne celui-ci en deux sous-ensembles disjoints à placer sur les sous-domaines. Les méthodes de bipartitionnement de graphes employées dans ce but sont étudiées au chapitre 11 ;
- une *fonction de distance* entre domaines, qui donne, dans le graphe cible, une mesure estimée de la distance entre deux domaines. Elle doit respecter certaines propriétés d'homogénéité, comme donner des résultats plus précis à mesure que les tailles des domaines diminuent.
  Cette fonction est utilisée lors des bipartitionnements de graphes de processus pour calculer dans la fonction de communication à minimiser la dilatation des arêtes du graphe source. Ceci revient à considérer, dans nos calculs, que les routages dans l'architecture cible sont de plus courts chemins. Cette supposition n'est pas déraisonnable, puisque les nouvelles générations de machines parallèles gèrent dynamiquement le routage par des algorithmes de plus courts chemins. Ainsi, la Paragon d'Intel utilise un routage nord-est/sud-ouest sur sa grille bidimensionnelle, et la Connection Machine 5 ne remonte-t-elle un message que jusqu'au premier père commun de l'émetteur et du récepteur.
  Nous avons donc choisi de ne pas implanter le calcul des routages dans notre modèle, laissant leur gestion au système de communication de la machine cible.

Toutes ces routines sont vues comme des boîtes noires par le programme de placement, qui peut donc accepter n'importe quel type d'architecture cible et de fonction de bipartitionnement de processus.

10.2 Fonction de coût

Afin de ne pas devoir paramétrer notre programme en fonction des machines cibles, nous avons choisi dans notre fonction de coût de dissocier la partie calcul de la partie communication. Nous cherchons à minimiser une fonction de coût de communication, tout en maintenant l'équilibrage de charge dans une tolérance fixée par l'utilisateur.
La fonction de coût que nous avons choisie est la somme, pour chaque arête du graphe source, du poids de l'arête multiplié par la dilatation de celle-ci :

$$f_C \stackrel{\text{def}}{=} \sum_{e_S \in E(S)} w(e_S)\,|\rho_{S,T}(e_S)| \quad .$$

Cette fonction, qui a déjà été retenue par plusieurs auteurs [25, 44, 48], possède plusieurs caractéristiques intéressantes :

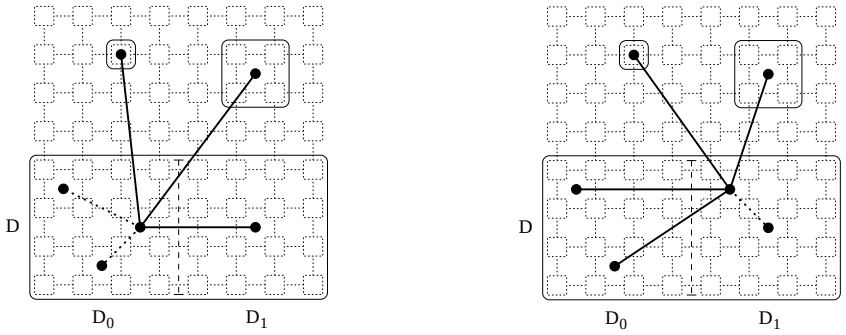
- elle est simple, rapide à calculer, et autorise les modifications incrémentales effectuées par les algorithmes itératifs ;
- sa minimisation favorise le regroupement des processus communiquant intensivement entre eux sur des processeurs aussi voisins que possible (voire les mêmes) ;
- indépendamment du type de routage implanté sur la machine cible ("*store-and-forward*", "*cut-through*"), elle modélise le trafic du réseau d'interconnexion et donc le risque de congestion de celui-ci.

La forte corrélation entre les valeurs de cette fonction et les temps effectifs d'exécution a été vérifiée expérimentalement par Hammond [44, page 48] sur la Connection Machine 2 (SIMD hypercubique), et par Hendrickson et Leland [50, page 683] sur un nCUBE 2 (machine MIMD hypercubique).

Pour que le placement final obtenu réalise nos objectifs, il faut que chaque placement partiel effectué lors des bipartitionnements successifs des sous-ensembles de processus sur les sous-domaines respecte également ces critères. Le bipartitionnement d'un sous-graphe S' de S maintiendra donc l'équilibrage de charge entre les deux sous-graphes dans la tolérance spécifiée par l'utilisateur, et la fonction de coût de communication à minimiser est

$$f'_C \stackrel{\text{def}}{=} \sum_{\substack{v \in V(S') \\ \{v,v'\} \in E(S)}} w(\{v,v'\})\,|\rho_{S,T}(\{v,v'\})| \quad ,$$

qui tient compte non seulement de la dilatation des arêtes internes au sous-graphe, mais aussi de celle des arêtes du cocycle de $V(S')$ dans S, comme illustré en figure 10.2.
La prise en compte, au moyen de la fonction de distance, de l'information de placement issue des autres tâches de bipartitionnement permet ainsi d'éviter de faire localement des choix qui pourraient se révéler globalement pénalisants. Ceci est plus amplement développé dans la section suivante.



a. Position initiale.

b. Après le déplacement d'un sommet.

Figure 10.2: Arêtes prises en compte dans la fonction de coût de communication lors du bipartitionnement d'un sous-graphe entre les deux sous-domaines D_0 et D_1 d'un domaine D. Les arêtes en pointillés correspondent à une dilatation nulle, leurs deux extrémités étant placées sur le même sous-domaine.

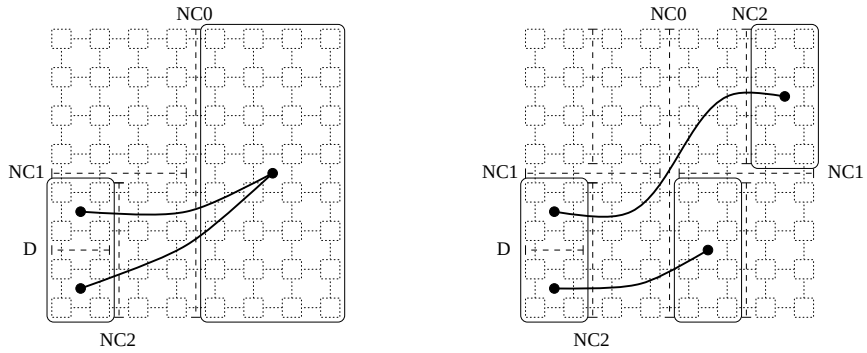
10.3 Schéma d'exécution

Du point de vue algorithmique, notre méthode de placement est de nature gloutonne, puisque les résultats des bipartitionnements successifs ne sont jamais remis en cause, mais autorise l'exécution d'algorithmes itératifs pour calculer ceux-ci.

Du double appel récursif de l'algorithme découle un schéma de récursion en arbre binaire, dont chaque nœud correspond à une tâche de bipartitionnement, c'est-à-dire aux bipartitionnements des sous-graphes de processeurs et de processus. À chaque manière de parcourir cet arbre correspond un séquencement des tâches de bipartitionnement, qui influence directement les placements obtenus puisque les résultats d'un bipartitionnement sont pris en compte par les tâches de bipartitionnement suivantes. Les deux types de séquencement considérés ici sont le séquencement en profondeur et le séquencement en largeur (par niveaux).

- Dans le cas d'un séquencement en profondeur, comme programmé dans l'ébauche ci-dessus, les tâches de bipartitionnement appelées dans les branches gauches de l'arbre n'ont pas d'informations précises sur les distances aux sommets qui seront traités par les branches droites (figure 10.3.a). En revanche, les tâches des branches droites disposent du placement exact des sommets traités par les branches gauches, et peuvent ainsi évaluer plus finement la fonction de coût à minimiser.

- Séquencer les tâches en effectuant un parcours par niveaux de l'arbre permet qu'à chaque niveau, une tâche de bipartitionnement puisse savoir sur quels sous-domaines ont été alloués les processus au niveau précédent. De ce fait, pour décider dans quel sous-domaine placer un processus donné, les tâches de bipartitionnement peuvent mieux prendre en compte les coûts de communication liés aux processus voisins, pour lesquels une distance globalement plus précise peut être calculée. Ceci provoque un intéressant effet de rétroaction : une fois qu'une



a. Séquencement en profondeur. **b.** Séquencement par niveaux.

Figure 10.3: Conséquences des séquencements en profondeur et par niveaux sur le bipartitionnement d'un domaine D appartenant à la branche la plus à gauche de l'arbre de bipartitionnement. Avec le séquencement par niveaux, l'information de placement concernant les sommets situés dans les branches droites de l'arbre de bipartitionnement est plus précise (N.C. indique le *Niveau de Coupe*).

arête a été prise dans une coupe entre deux sous-domaines, la distance entre ses extrémités sera prise en compte dans la fonction de coût de communication à minimiser, et les tâches suivantes auront alors plus tendance à garder ces processus proches l'un de l'autre, comme illustré en figure 10.3.b. De plus, comme tous les domaines sont bipartitionnés à chaque niveau, ils ont tous des tailles équivalentes, ce qui respecte l'homogénéité de la fonction de distance et donne plus de cohérence à l'algorithme.

Notre programme peut utiliser indifféremment les deux séquencements. Les tâches de bipartitionnement sont stockées en file, l'exécution d'une tâche donnant lieu à la création d'au plus deux nouvelles tâches. Celles-ci sont alors ajoutées à la file, en tête dans le cas d'un parcours en profondeur (on a alors une pile plutôt qu'une file), ou en queue dans le cas d'un parcours par niveaux. L'efficacité des deux séquencements est étudiée en section 12.3.2.

Remarquons que, en prenant l'hypercube comme topologie cible, et un séquencement en profondeur, notre programme de placement est équivalent à celui de Ercal, Ramanujam, et Sadayappan [25]. En ce sens, notre travail, en formalisant les concepts de domaine, distance, et schéma d'exécution, peut être vu comme une généralisation du leur, permettant de gérer de nombreuses topologies cibles et méthodes de bipartitionnement.

10.4 Complexité

L'objectif de cette section est d'évaluer la complexité de notre algorithme de bipartitionnement récursif en fonction de celle des méthodes de bipartitionnement que nous utilisons au cours de chaque tâche.
Appelons DRB (pour "*Dual Recursive Bipartitioning*") notre algorithme, et soient $BipaT$ et $BipaS$ les algorithmes de bipartitionnement de domaine et de graphe qu'il utilise.

Proposition 10.39. *Soient S un graphe source et T un graphe cible connexes, avec $|E(S)| \geq |E(T)|$. Si, pour tous $T' \subseteq T$ et $S' \subseteq S$, $\mathcal{C}_E(BipaT(T'))$ est en $O(|E(T')|+|V(T')|)$ et $\mathcal{C}_E(BipaS(S'))$ est en $O(|E(S')|+|V(S')|)$, et si les résultats des bipartitionnements de domaine sont équilibrés, alors $\mathcal{C}_E(DRB(S,T))$ est en $O(|E(S)|)$.*

Preuve. Le stockage du graphe source S se fait en $O(|V(S)|+|E(S)|)$, c'est-à-dire en $O(|E(S)|)$ puisque $|E(S)| \geq |V(S)|-1$.
Comme, par hypothèse, les résultats de l'algorithme de bipartitionnement de domaines sont équilibrés, l'arbre de bipartitionnement des domaines est complet au moins jusqu'à son avant-dernier niveau, et sa profondeur est $\lceil \log_2(|V(T)|) \rceil$. Soit $\mathcal{F}_i(S,T)$ l'ensemble des paires (T',S') traitées par les tâches de bipartitionnement du niveau i de l'arbre, où T' est un sous-domaine de T et S' le sous-graphe de S placé sur T'. En particulier, $\mathcal{F}_0(S,T) = \{(S,T)\}$.
Comme les tâches de bipartitionnement libèrent, au moment de leur terminaison, la mémoire qu'elles utilisent, la complexité en espace de notre algorithme DRB est le maximum sur toutes les tâches des complexités en espace :

$$\begin{aligned}\mathcal{C}_E(\mathrm{DRB}(S,T)) &= O(|E(S)|) + \\ &\quad O\left(\max_{i=0}^{\lceil \log_2(|V(T)|)\rceil} \max_{\substack{(S',T')\in \\ \mathcal{F}_i(S,T)}} \max(\mathcal{C}_E(\mathrm{BipaT}(T')), \mathcal{C}_E(\mathrm{BipaS}(S')))\right) .\end{aligned}$$

À chaque niveau de l'arbre de bipartitionnement, tous les sous-domaines et sous-graphes considérés sont des sous-graphes disjoints des graphes cible et source, dont les tailles sont majorées par les tailles de ces derniers. On a donc :

$$\begin{aligned}\mathcal{C}_E(\mathrm{DRB}(S,T)) &= O(|E(S)|) + O\left(\max_{i=0}^{\lceil \log_2(|V(T)|)\rceil} O(|E(T)|+|V(T)|+|E(S)|+|V(S)|)\right) \\ &= O(|E(S)|) .\end{aligned}$$

□

De manière analogue, nous pouvons établir la complexité en temps de l'algorithme sous les mêmes hypothèses.

Proposition 10.40. *Soient S un graphe source et T un graphe cible connexes, avec $|E(S)| \geq |E(T)|$. Si, pour tous $T' \subseteq T$ et $S' \subseteq S$, $\mathcal{C}_T(BipaT(T'))$ est en $O(|E(T')|+|V(T')|)$ et $\mathcal{C}_T(BipaS(S'))$ est en $O(|E(S')|+|V(S')|)$, et si les résultats des bipartitionnements de domaine sont équilibrés, alors $\mathcal{C}_T(DRB(S,T))$ est en $O(|E(S)|\log_2(|V(T)|))$.*

Preuve. En reprenant les définitions de la proposition précédente, on a

$$\mathcal{C}_T(\mathrm{DRB}(S,T)) = O\left(\sum_{i=0}^{\lceil \log_2(|V(T)|)\rceil} \sum_{(S',T')\in\mathcal{F}_i(S,T)} (\mathcal{C}_T(\mathrm{BipaT}(T')) + \mathcal{C}_T(\mathrm{BipaS}(S')))\right) .$$

À chaque niveau de l'arbre de bipartitionnement, tous les sous-domaines et sous-graphes considérés sont des sous-graphes disjoints des graphes cible et source. De fait, la somme pour chaque niveau

de complexités linéaires par rapport au nombre d'arêtes et de sommets de ces sous-graphes est linéaire par rapport au nombre d'arêtes et de sommets des graphes initiaux. On a donc

$$\begin{aligned} \mathcal{C}_T\left(\mathrm{DRB}(S,T)\right) &= O\left(\sum_{i=0}^{\lceil \log_2(|V(T)|)\rceil} O(|E(T)|+|V(T)|+|E(S)|+|V(S)|)\right) \\ &= O\left(|E(S)|\,\lceil \log_2(|V(T)|)\rceil\right) \quad . \end{aligned}$$

□

Chapitre 11

Bipartitionnement de domaines et de graphes

11.1 Fonctionnalités du module de bipartitionnement de domaines

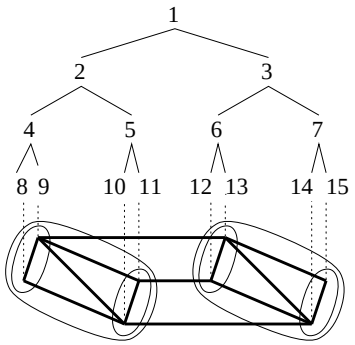
11.1.1 Fonctions de domaines

Suivant les topologies d'architecture cible, les résultats des fonctions de domaines sont soit calculés algorithmiquement (c'est le cas pour les grilles et tores bidimensionnels et tridimensionnels, l'hypercube, le graphe complet, et le graphe "feuilles d'arbre"), soit extraits de tables précalculées (des programmes spécifiques permettent de générer ces tables pour les graphes de De Bruijn, FFT, et CCC).
La gestion algorithmique de certaines architectures, qui peut sembler redondante vis-à-vis de l'utilisation de tables précalculées d'usage général, permet, en tirant parti de la régularité des topologies concernées, de gérer de grandes architectures cibles sans avoir besoin de précalculer et stocker sur disque des tables dont la taille évolue comme le carré du nombre de processeurs.

11.1.2 Description d'architectures par fichier

La description d'une topologie quelconque s'effectue au moyen de deux tableaux.

- Le premier associe à chaque processeur son numéro de *domaine terminal*. Les numéros de domaines terminaux des processeurs s'obtiennent en donnant au domaine initial contenant tous les processeurs le numéro 1, les deux sous-domaines d'un domaine non-terminal de numéro n étant alors respectivement numérotés $2n$ et $2n + 1$. La donnée des numéros de domaines terminaux des processeurs suffit ainsi à définir de manière univoque l'arbre de bipartitionnement du graphe d'architecture.



a. Bipartitionnements successifs du graphe d'architecture.

8	9	11	13	10	12	14	15
1							
2	1						
2	1	2					
1	1	1	2				
3	2	1	1	2			
2	2	2	1	1	1		
3	2	3	1	2	2	1	

b. Les deux tableaux décrivant la topologie du graphe.

Figure 11.1: Tableaux correspondant à une architecture homogène de graphe de De Bruijn UB(2, 3). Ces deux tableaux se basent sur une numérotation implicite des processeurs. Par exemple, les numéros de domaines terminaux 8 et 11 sont respectivement attribués aux premier et troisième processeurs, ceux-ci étant à distance 2 l'un de l'autre (deuxième ligne, première colonne de la matrice, qui ne possède pas de diagonale).

- Le deuxième tableau est une matrice triangulaire inférieure donnant la distance entre toute paire de processeurs. Cette matrice de distances, utilisée conjointement avec l'arbre de bipartitionnement, permet d'évaluer par moyennes successives la distance entre toute paire de domaines de l'architecture.

Un exemple de description de topologie est donné en figure 11.1.

Dans le cas d'architectures hétérogènes, la valeur de puissance de traitement associée aux processeurs permet aux algorithmes de bipartitionnement d'équilibrer en conséquence la charge sur les sous-domaines, et les distances entre processeurs sont calculées en fonction des chemins critiques obtenus par sommation des valeurs des arêtes du graphe cible. Les chemins de plus fort débit ont les plus petits coûts, et y router des canaux supportant un grand volume de données coûte autant (par produit) que de placer des canaux de faible volume sur des liens de grande valeur, c'est-à-dire de faible débit.

11.1.3 Décomposition d'architectures par placement

L'arbre de bipartitionnement d'une architecture peut facilement être calculé en utilisant notre programme de placement. Pour cela, on effectue le placement du graphe valué correspondant à l'architecture étudiée sur le graphe complet homogène de même ordre. Choisir le graphe complet comme graphe cible permet de ne minimiser que la coupe du bipartitionnement, sans tenir compte d'une topologie cible qui n'a ici aucun sens. Comme tous les autres domaines sont à distance 1 de celui qu'on bipartitionne, les contributions des arêtes du cocycle n'ont aucune influence sur le choix des sous-domaines où placer les sommets. En ne minimisant que la coupe, les bipartitionnements privilégient alors uniquement la localité des communications dans le sous-graphe qu'ils traitent.
Dans le cas d'architectures hétérogènes, la minimisation de la fonction de communication privilégie la coupe des arêtes de plus faible coût, c'est-à-dire de plus fort débit. Du point de vue de

la communication, on obtient une décomposition hiérarchisée, dans laquelle les liens de plus fort débit servent d'épine dorsale ("*backbone*") entre des domaines disposant de liens de plus faible débit.

Le placement d'un graphe de processus quelconque sur un graphe cible quelconque peut donc se faire automatiquement en deux étapes.

- On appelle tout d'abord le placeur pour placer le graphe d'architecture sur le graphe complet, ce qui permet de construire l'arbre de bipartitionnement de cette architecture.
  Le résultat du placement est concaténé à la matrice des distances, directement calculée à partir du graphe, pour former le fichier de description d'architecture.
- On appelle une seconde fois le placeur pour placer le graphe de processus sur le graphe d'architecture dont la description vient d'être construite.

11.2 Quelques méthodes de bipartitionnement de domaines

Nous présentons ici la manière dont nous bipartitionnons certaines des architectures que nous utilisons.

11.2.1 Le graphe complet

Un domaine du graphe complet est défini comme un ensemble de sommets numérotés consécutivement, et identifié par les valeurs extrêmes de ceux-ci. Le bipartitionnement d'un domaine de taille k s'effectue directement en affectant à un sous-domaine les $\lfloor \frac{k}{2} \rfloor$ sommets de plus petits numéros, et à l'autre les $\lceil \frac{k}{2} \rceil$ sommets restants. La distance entre deux domaines est égale à 0 s'il s'agit exactement du même domaine, et 1 sinon.

11.2.2 Les grilles et tores multidimensionnels

Les domaines des grilles et tores multidimensionnels sont des zones rectangulaires décrites par leurs extrêma. On bipartitionne un tel domaine en découpant en deux (à une colonne près) sa plus grande dimension, et la distance entre deux domaines est prise comme la distance en norme 1 entre les centres des deux domaines. Ce bipartitionnement est celui obtenu par la méthode des dissections emboîtées.

11.2.3 L'hypercube

Le découpage de l'hypercube tire parti de la structure très régulière de celui-ci. Les domaines hypercubiques sont des sous-hypercubes, et la fonction de bipartitionnement de domaine sépare un hypercube en deux sous-hypercubes selon les dimensions décroissantes.
Tout domaine est représenté par un mot dont les bits peuvent être positionnés ou non. Lorsqu'un domaine est bipartitionné selon une dimension, les représentations de ses deux sous-domaines sont calculées en positionnant le bit correspondant à cette dimension, à 0 et 1 respectivement. Par

exemple, le domaine 10*** de H(5) est bipartitionné selon la troisième dimension en 100** et 101**.
La distance entre deux sous-domaines est la distance de Hamming prise sur les bits positionnés des deux sous-domaines. Ainsi, sur H(5), les domaines 101** et 1101* sont à distance 2 l'un de l'autre.

11.2.4 Le graphe de De Bruijn binaire

La décomposition des graphes de De Bruijn binaire est moins naturelle que les précédentes, du fait que le bipartitionnement d'un graphe de De Bruijn ne donne qu'exceptionnellement des sous-graphes qui sont des graphes de De Bruijn.

Une première méthode, qui nous a été suggérée par Ioan Bond, repose sur la décomposition de $\mathrm{UB}(2,k)$ en deux copies d'un sous-graphe partiel de $\mathrm{UB}(2,k-1)$. Le bit de poids le plus fort d'un sommet de $\mathrm{UB}(2,k)$ indique à quelle copie il appartient, et son étiquette dans celle-ci s'obtient en effectuant le ou-exclusif deux-à-deux des bits constituant l'étiquette du sommet dans $\mathrm{UB}(2,k)$. Par exemple, le sommet 01100 de $UB(2,5)$ appartient à la première copie du sous-graphe partiel de $\mathrm{UB}(2,4)$, dans laquelle il a pour étiquette 1010. Il est facile de montrer que toutes les arêtes de chacun des deux sous-graphes ainsi obtenus sont des arêtes de $\mathrm{UB}(2,k-1)$ et que la réciproque est fausse. La décomposition d'une architecture de graphe de De Bruijn s'effectue en appliquant récursivement la méthode ci-dessus.

Ce découpage, qui a l'avantage d'être systématique, ne fournit cependant pas des décompositions satisfaisantes. Du fait que les sous-graphes obtenus possèdent à chaque fois moins d'arêtes que le sous-graphe de De Bruijn correspondant et que la coupe entre les deux sous-domaines n'est pas minimale, on ne privilégie pas la localité des communications dans chaque sous-domaine, ce qui va à l'encontre du principe de notre algorithme.
Les décompositions que nous utilisons sont donc celles fournies par la méthode de placement que nous avons présentée en section 11.1.3, qui permet en particulier d'obtenir l'exemple de la figure 11.1.a. Ce choix est discuté en section 12.3.3.

11.2.5 Le graphe "feuilles d'arbre"

Le graphe "feuilles d'arbre" correspond à une machine dont la topologie est un arbre binaire complet, les processeurs de calcul se trouvant aux feuilles et les autres nœuds étant des routeurs de communication (figure 11.2). Le placement ne s'effectue que sur les feuilles, mais les distances entre ces dernières sont prises sur l'arbre binaire entier.

Nous utilisons ce graphe pour modéliser les machines multi-étages à bande passante constante, telle la Connection Machine CM-5, dont le réseau de communication [59], représenté en figure 11.3, s'inspire très fortement du graphe "*Fat Tree*" proposés par Leiserson [58]. Le nombre de liens de communication de cette machine augmente à mesure que l'on s'approche de la racine afin d'éviter

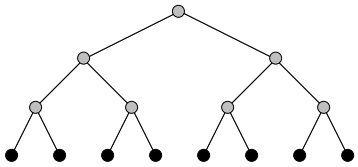


Figure 11.2: Le graphe "feuilles d'arbre". Les processeurs sont représentés en noir, et les routeurs en gris.

la congestion du sommet de l'arbre, ce qui fait qu'en pratique la bande passante est identique entre toute paire de processeurs et varie peu avec la charge du réseau [60].
Citons encore la CS-2 de PCI [77], bâtie sur un réseau multi-étages de type Oméga (voir figure 11.4), et la SP-1 d'IBM, qui utilise une topologie analogue à la précédente.

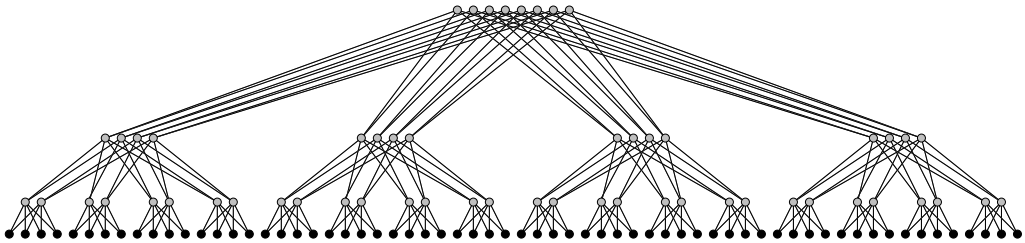


Figure 11.3: Réseau de communication de données de la Connection Machine 5 (représentée ici en configuration avec 64 processeurs). Les processeurs sont représentés en noir, et les routeurs en gris.

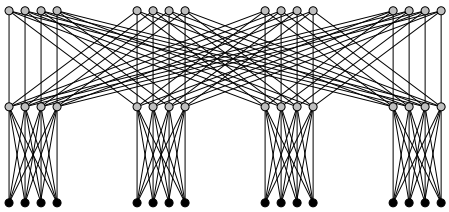


Figure 11.4: Réseau de communication de données de la Computing Surface 2. Les processeurs sont représentés en noir, et les routeurs en gris.

Le bipartitionnement du graphe "feuilles d'arbre" s'effectue simplement en ôtant la racine de l'arbre de routage, afin d'obtenir deux sous-arbres identiques ayant chacun la moitié des processeurs de l'arbre initial. Puisque la bande passante est supposée constante, les distances entre deux processeurs sont celles obtenues sur l'arbre de routage.

11.3 Fonctionnalités du module de bipartitionnement des graphes sources

11.3.1 Travaux et tâches

L'interfaçage du corps du programme avec les tâches de bipartitionnement se fait par l'intermédiaire d'objets appelés *travaux*. Un travail est constitué :

- du graphe des processus, qui possède l'information de voisinage ;
- d'un placement partiel servant au calcul des distances (c'est la liste des sous-domaines, éventuellement non réduits à un processeur, sur lesquels sont actuellement placés tous les processus du graphe initial) ;
- de la liste, non-vide, des processus sur lesquels porte le travail ;
- du domaine, couvrant plus d'un processeur, sur lequel sont actuellement placés les processus de la liste.

L'appel à une tâche se fait en passant en paramètre le travail à effectuer. En retour, on obtient deux sous-travaux au plus, tels que les deux sous-listes de processus soient disjointes et forment par concaténation la liste initiale, et que les deux sous-domaines soient le résultat du bipartitionnement du domaine passé en paramètre.

Les méthodes de bipartitionnement de processus sont utilisées dans l'algorithme comme des boîtes noires, ce qui permet d'utiliser toutes méthodes pouvant accepter notre fonction de coût, telles celles de type Kernighan-Lin [53, 30], recuit simulé [13], affectation quadratique [82], etc..

Les tâches de bipartitionnement maintiennent de manière interne une image de la bipartition en cours, indiquant pour chaque sommet du travail si celui-ci est actuellement placé sur le premier ou deuxième sous-domaine considéré. On peut, par ce moyen, appeler successivement plusieurs méthodes différentes, chacune utilisant le résultat de la précédente comme bipartition initiale. De fait, la première méthode appelée est toujours une méthode gloutonne directe, qui construit une bipartition uniquement à partir des données du travail.
Il est en outre possible de conditionner l'appel aux différentes méthodes en fonction des propriétés du travail à effectuer (taille du domaine, cardinal de la liste de processus, distribution de leur poids, ...), ce qui permet de définir des *stratégies de placement* spécifiques à certains types de problèmes.

11.3.2 Gestion des graphes déséquilibrés

La gestion des graphes de processus déséquilibrés, c'est-à-dire dont certains processus ont des poids bien plus grands que la moyenne, soulève plusieurs questions. Considérons par exemple un graphe de processus complet dont tous les sommets sont de poids unité, à l'exception d'un seul de poids égal à l'ordre de ce graphe moins un. Si un algorithme de bipartitionnement exact était appliqué à

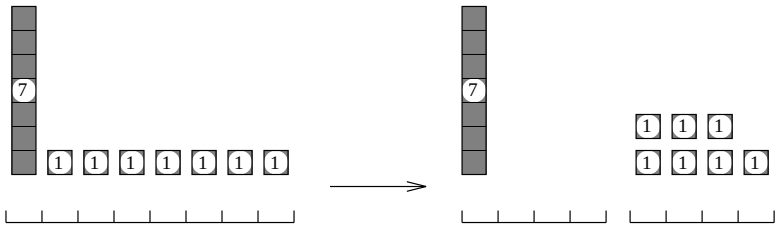


Figure 11.5: Bipartitionnement exact d'un graphe fortement déséquilibré.

ce graphe, le sommet le plus lourd serait placé seul dans un sous-domaine et tous les autres dans le second, laissant alors tous les processeurs du premier domaine inutilisés sauf un (voir figure 11.5).

La première question est de savoir si l'existence d'un tel cas est justifiée. On peut en effet penser qu'un graphe de calcul déséquilibré résulte d'une mauvaise parallélisation des algorithmes du programme sous-jacent, et qu'en tout état de cause il est très difficile au placeur d'équilibrer ce qui ne l'est pas, renvoyant ainsi la balle dans le camp du programmeur. Plusieurs facteurs rendent à notre sens nécessaire la gestion de tels graphes.

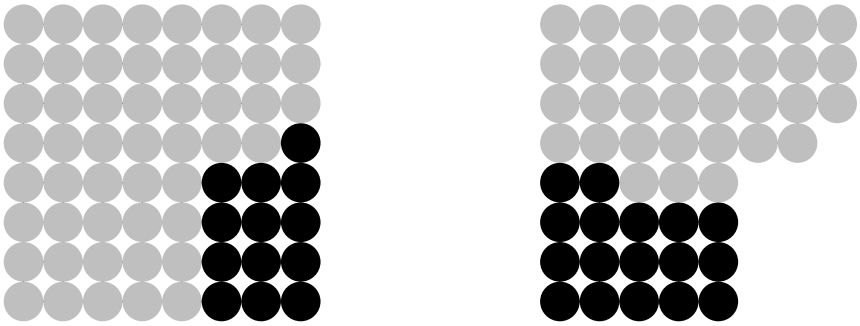
- Citons tout d'abord l'existence de machines parallèles "à la carte", dont les nœuds peuvent être équipés individuellement de coprocesseurs de calcul, mémoire, et périphériques supplémentaires, ainsi que l'utilisation croissante de grappes de machines hétérogènes agencées en "machines parallèles virtuelles" [7, 34]. Les programmes spécifiquement conçus pour ces machines peuvent donc présenter des inhomogénéités qui reflètent celles de la machine. C'est typiquement le cas des programmes SPMD pour lesquels on effectue la répartition des données en fonction de la machine cible avant le placement.

- Un deuxième facteur est lié à notre algorithme lui-même. En effet, le comportement décrit ci-dessus peut se produire aux feuilles de l'arbre de bipartitionnement même dans le cas de graphes de processus faiblement déséquilibrés, par exemple lors du placement de quatre processus de poids 3, 1, 1, et 1 sur quatre processeurs homogènes. Remarquons cependant que cela n'arrive que quand on ne peut atténuer par le nombre les disparités de poids des processus, c'est-à-dire quand on a environ autant de processus que de processeurs.

La gestion des processus déséquilibrés peut alors s'envisager de deux manières antagonistes.

- Puisque le processus le plus lourd pénalise de toute façon les processus légers, il est inutile de répartir la charge de ceux-ci sur l'ensemble des processeurs. Mieux vaut laisser des processeurs oisifs, qui peuvent être attribués à d'autres utilisateurs de la machine parallèle, et équilibrer la charge sur ceux que l'on utilise effectivement. Cette approche correspond à une exploitation multi-utilisateurs concurrente de la machine parallèle.

- Puisque les processeurs du domaine ont été réservés, il vaut mieux tous les utiliser en y répartissant le plus possible les processus. Ainsi, ceux-ci pourront accéder avec le minimum



a. Construction d'un sous-domaine à 13 sommets (dessinés en noir) sur la machine complète.

b. Construction d'un sous-domaine à 17 sommets (dessinés en noir) sur la sous-machine résultante.

Figure 11.6: Construction de sous-domaines sur l'architecture de grille bidimensionnelle $\mathrm{M}_2(8,8)$ par placement du graphe d'architecture sur la chaîne à deux sommets valués.

de risque de contension aux ressources indépendantes de chaque processeur (puissance de calcul, mémoire, accès disque, ...), qu'ils auraient à partager s'ils se trouvaient groupés sur moins de processeurs. Cette approche correspond à une exploitation en traitement par lots ("*batch*") de la machine parallèle.

Nous avons choisi d'implanter la seconde approche, parce qu'elle permet de tirer parti de tous les processeurs qui ont été réservés pour l'exécution d'un programme tout en autorisant un fonctionnement analogue à celui de la première approche.
En effet, soit S le graphe source à placer sur le graphe cible T correspondant à la partie de la machine parallèle qui a été réservée pour son exécution. Si l'on souhaite que les processeurs effectivement utilisés soient chargés comme celui accueillant le processus le plus gros, il faut réduire le domaine de placement à $T' \subseteq T$ tel que

$$|V(T')| = \max\left(|V(T)|, \left\lfloor \frac{\sum_{v_S \in V(S)} (w(v_S))}{\max_{v_S \in V(S)} (w(v_S))} \right\rfloor\right) \quad ,$$

cette restriction de domaine s'effectuant avant le placement proprement dit.

Une fois le cardinal de T' connu, la construction de T' à partir de T peut se faire en utilisant notre placeur. Pour cela, on place le graphe cible T sur la chaîne à deux sommets, l'un de poids $|V(T')|$, et l'autre de poids $|V(T)| - |V(T')|$. L'unique bipartitionnement qui a lieu sert à minimiser la communication entre les deux sous-parties, donc à maximiser la communication au sein de chaque sous-partie. On prend alors comme sous-domaine T' la sous-partie de T placée sur le sommet de poids $|V(T')|$, comme illustré en figure 11.6.

Nous avons réalisé l'approche choisie grâce à une procédure de limitation de poids, dont le but est de masquer le poids des sommets les plus lourds afin que ceux-ci n'influent pas trop sur l'équilibrage de charge.
Au début de chaque travail, on calcule le *poids effectif* de chaque processus, qui correspond au

minimum de du poids réel du processus et d'une valeur plafond. Celle-ci est ajustée itérativement de manière à être égale au double de la moyenne des poids effectifs des processus. Ainsi, la moyenne est centrée par rapport à la distribution des poids effectifs obtenue.
L'utilisation des poids effectifs par les algorithmes de bipartitionnement revient, dans le cas de graphes déséquilibrés, à calculer l'équilibrage de charge plus en terme de nombre de processus que de poids de processus, ce qui réduit l'impact de cas pathologiques tels que celui décrit précédemment.

11.4 Quelques méthodes de bipartitionnement de graphes

Nous présentons dans cette section les méthodes de bipartitionnement de graphes utilisées par notre programme. Pour simplifier leur description, nous supposons que les deux sous-domaines sur lesquels s'applique le bipartitionnement sont de tailles et poids égaux. Lorsque cela n'est pas le cas, des mécanismes compensateurs sont mis en œuvre, qui sont le plus souvent basés sur l'utilisation de pondérations dépendant des poids relatifs des sous-domaines.

11.4.1 L'algorithme aléatoire de bipartitionnement (AAB)

Cet algorithme glouton fournit un bipartitionnement aléatoire sans tenir compte de l'équilibrage de charge ni de la minimisation de la communication. Chaque sommet est affecté indépendamment à l'un ou l'autre sous-domaine, avec une probabilité proportionnelle aux poids relatifs de ceux-ci. Cette méthode sert à obtenir des mesures de référence (dilatation moyenne des arêtes, charge moyenne des sommets) servant à l'évaluation des autres méthodes.

Trivialement, pour tout graphe S', $\mathcal{C}_E(\mathrm{AAB}(S')) = |V(S')|$ et $\mathcal{C}_T(\mathrm{AAB}(S')) = |V(S')|$.

11.4.2 L'algorithme glouton de bipartitionnement (AGB)

Cet algorithme glouton fournit un bipartitionnement équilibré sans tenir compte de la communication. Son principe est simple : les sommets étant triés par poids décroissants et les deux sous-domaines étant vides, on affecte successivement chaque sommet au sous-domaine de plus petit poids de sommet cumulé (compensé si les deux sous-domaines sont de poids différents).
Cette méthode donne rarement une solution d'équilibrage optimal (voir figure 11.7), mais la différence entre son résultat et la solution optimale est majorée par le plus petit poids de sommet contenu dans le domaine de plus grand poids cumulé. Il donne donc souvent une solution "bien" équilibrée.

Cet algorithme est la méthode de bipartitionnement gloutonne initiale par défaut de toutes les stratégies implantées dans notre programme. Lorsque l'ensemble des sommets est déjà trié (ce qui est le cas des graphes homogènes), sa complexité en temps est linéaire par rapport au nombre de sommets du graphe de processus. Pour tout graphe S', on a $\mathcal{C}_E(\mathrm{AGB}(S')) = |V(S')|$, et $\mathcal{C}_T(\mathrm{AGB}(S')) = |V(S')|$.

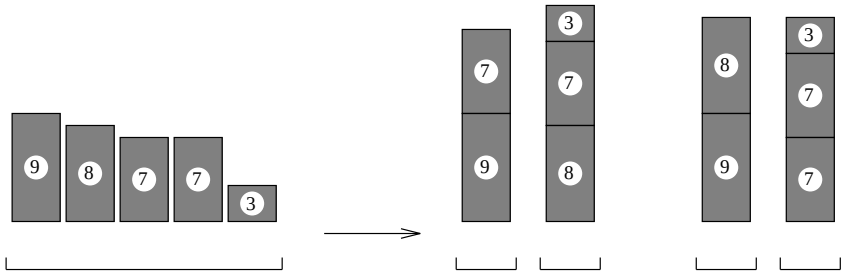


Figure 11.7: Affectation non-optimalement équilibrée par l'algorithme glouton de bipartitionnement, et affectation optimale.

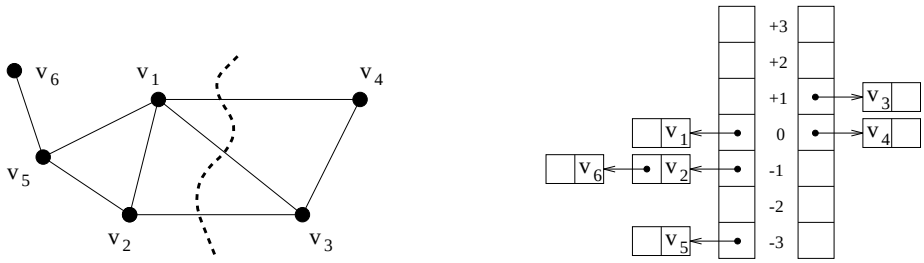
11.4.3 L'algorithme de Gibbs, Poole, et Stockmeyer (GPS)

L'algorithme de Gibbs, Poole, et Stockmeyer [40], que nous avons succintement décrit en page 15, nous sert ici à calculer un bipartitionnement initial équilibré orienté vers la minimisation de la communication. Nous utilisons pour cela le découpage en couches généré par l'algorithme.
Dans un premier temps, on applique l'algorithme de Gibbs, Poole, et Stockmeyer au graphe source, pour calculer un découpage en couches de celui-ci. Ceci fait, on parcourt couche après couche l'ensemble des sommets à partir de la couche 0, en affectant les sommets rencontrés au premier sous-domaine, jusqu'à ce que la moitié des poids aient été traités. Les sommets restants sont alors affectés au deuxième sous-domaine.

Le principe de cette heuristique reprend donc celui utilisé par Gibbs, Poole, et Stockmeyer dans la conception de leur algorithme : on suppose que le découpage en couches généré par l'algorithme minimise la taille des couches, et donc le cocycle de chaque couche.
Cette idée a déjà été appliquée par George et Liu [39] pour renuméroter de grands systèmes linéaires par la méthode des dissections emboîtées. Ils utilisent la couche médiane comme séparateur sommet du graphe, et appliquent récursivement leur algorithme aux deux sous-graphes résultants.

Dans le cas des graphes de maillage réguliers bidimensionnels, il y a une forte analogie entre les chemins de longueur diamètre partant du nœud pseudo-périphérique choisi et les axes principaux calculés par les méthodes inertielles. En ce sens, cette heuristique représente le moyen de faire intervenir la géométrie dans le bipartitionnement, sans pour autant faire explicitement appel aux coordonnées des sommets comme c'est le cas dans les méthodes inertielles [48]. À la différence de celles-ci, elle peut donc être utilisée lorsque l'information géométrique n'est pas disponible.

Soit S' le graphe à bipartitionner. La complexité en espace de la méthode GPS est celle du stockage du graphe et du vecteur des distances des sommets au nœud pseudo-périphérique, donc $\mathcal{C}_E\,(\mathrm{GPS}(S'))$ est en $O(|V(S')| + |E(S')|)$.
Chaque étape de l'algorithme initial de Gibbs, Poole, et Stockmeyer nécessite l'accès à tous les voisins de chaque sommet pour propager l'information de distance ; ceci se fait en parcourant



a. Bipartition courante d'un graphe homogène.

b. Listes et tableaux de gains correspondant à cette bipartition.

Figure 11.8: Structures de données utilisées par l'heuristique de Fiduccia et Mattheyses.

toutes les arêtes incidentes aux sommets. Une fois que l'algorithme de Gibbs, Poole, et Stockmeyer a terminé, l'affectation des sommets aux parties se fait en un unique parcours de $V(S')$.
Puisque l'algorithme termine dès que l'exécution d'une étape n'augmente pas le nombre de couches, le nombre maximal d'étapes est majoré par $\mathrm{diam}(S')$, et $\mathcal{C}_T\,(\mathrm{GPS}(S'))$ est en $O(\mathrm{diam}(S')\,|E(S')|+|V(S')|)$. En pratique, le nombre de ces étapes est très petit [40] ; il est en particulier inférieur à 8 pour toutes nos expérimentations. De fait, $\mathcal{C}'_T\,(\mathrm{GPS}(S'))$ est en $O(|V(S')|+|E(S')|)$.

11.4.4 L'heuristique de Fiduccia et Mattheyses améliorée (FMA)

La première heuristique que nous avons implantée est une amélioration de l'algorithme de Fiduccia et Mattheyses permettant de gérer les graphes de processus valués.

11.4.4.1 L'heuristique de Fiduccia et Mattheyses

L'heuristique de bipartitionnement de graphe de Fiduccia et Mattheyses [30] est une amélioration en temps quasi-linéaire de l'algorithme de Kernighan et Lin [53]. Son but est de minimiser la coupe entre deux sous-ensemble de sommets, tout en maintenant la différence de leurs cardinaux dans une tolérance prédéfinie par l'utilisateur. Partant d'une solution équilibrée de coupe quelconque, l'algorithme essaye, à chacune de ses itérations, de réduire la coupe de la solution courante.

L'algorithme maintient pour chaque sommet une valeur de *gain*, qui représente la valeur dont la coupe serait réduite si le sommet était placé dans l'autre sous-domaine (cette valeur peut donc être négative). Les sommets de même gain placés dans le même sous-domaine sont groupés dans des *listes chaînées de gain*, qui sont indexées dans deux *tableaux de gain*, comme illustré dans la figure 11.8.

À chaque itération, l'algorithme construit un *ordonnancement* d'autant de sommets que possible. Pour cela, il choisit dans les deux tableaux de gain le sommet de gain le plus élevé (qui peut être négatif) et dont le mouvement ne déséquilibrera pas la charge des sous-domaines au delà de la tolérance prédéfinie. Le sommet choisi est déplacé, et les gains de ses voisins sont modifiés en conséquence. Ceci est répété jusqu'à ce que tous les sommets aient été choisis ou qu'aucun mouvement de sommets non encore choisis ne permette de rester dans la tolérance d'équilibrage. La nouvelle solution est alors construite à partir de la solution courante en validant le déplacement d'autant

de sommets de l'ordonnancement que nécessaire pour obtenir le gain cumulé positif le plus grand à partir du début de l'ordonnancement. Si l'ordonnancement ne génère pas de gain cumulé positif, l'algorithme termine.
Le choix de sommets de gain négatif, avec l'espoir que cela permettra de déplacer ensuite des sommets générant un gain positif supérieur, permet le franchissement de *minima locaux* de la fonction de coupe.

On a donc l'algorithme suivant.

```
bipartition_processus_FM (P0, P1, D0, D1, M)
Ensemble_De_Processus    P0, P1;                /* Bipartition initiale (par AGB) */
Ensemble_De_Processeurs  D0, D1;                /* Les deux sous-domaines         */
Placement_Partiel        M;                     /* Placement partiel courant      */
{
  Processus               p;                    /* Processus courant                          */
  Processus               pvois;                /* Processus voisin du processus courant      */
  Ordonn_Position         omeilleur;            /* Position du meilleur gain cumule           */
  Ensemble_De_Processus   Pgarde;               /* Ensemble de processus vraiment deplaces    */
  Ensemble_De_Processus   Premet;               /* Ensemble de processus a remettre en place */

  faire {                                       /* Boucle sur les ordonnancements          */
    gain_calcule (P0, P1, D0, D1, M);           /* Calcule les gains de tous les processus */
    table_ajoute (P0, P1);                      /* Ajoute les processus dans les 2 tables  */
    ordonn_init  ();                            /* Initialise l'ordonnancement             */

    tant que ((p = table_meilleur ()) != NULL) { /* Tant qu'on trouve des sommets a deplacer */
      table_supprime (p);                       /* On enleve le sommet des tables           */
      ordonn_ajoute  (p);                       /* On ajoute le sommet a l'ordonnancement   */
      deplace        (P0, P1, p);               /* On change le sommet de partie            */

      tant que ((pvois = voisin (p)) != NULL) {  /* Pour tous les voisins du sommet          */
        si (table_present (pvois)) {            /* Si le voisin est toujours dans les tables */
          table_supprime (pvois);               /* On le retire des tables                  */
          gain_modifie   (pvois);               /* On recalcule son gain                    */
          table_ajoute   (pvois);               /* On le replace dans les tables            */
        }
      }
    }                                           /* Fin de la boucle sur les sommets */

    omeilleur = ordonn_meilleur ();             /* Cherche la position de meilleur gain cumule  */
    Pgarde    = ordonn_tronque  (omeilleur);    /* Garde les deplacements entre 1 et omeilleur  */
    Premet    = ordonn_liste    () - Pgarde;    /* On doit remettre les autres sommets en place */
    pour tout (p dans Premet)                   /* Pour tout sommet a remettre en place         */
      deplace (P0, P1, p);                      /* On le re-deplace dans sa partie initiale     */

  } tant que (Pgarde != VIDE);                  /* Tant qu'on a vraiment deplace des sommets */

  retour (P0, P1);                              /* Retourne la nouvelle partition calculee */
}
```

11.4.4.2 Limitations et améliorations

La quasi-linéarité de l'algorithme de Fiduccia et Mattheyses découle de sa capacité à trouver un sommet de plus grand gain en temps quasi-constant, ce qui n'est possible dans l'algorithme initial qu'au prix de deux limitations.

- Premièrement, il est supposé que le degré du graphe est petit et que les arêtes sont de coût unitaire. Les gains des sommets du graphe S à bipartitionner sont compris entre $-\Delta(S)$ et $+\Delta(S)$, avec $\Delta(S)$ petit, ce qui fait que la recherche dans les tableaux de gain de la liste non-vide de plus grand gain prend un temps quasi-constant, vu le peu d'indices à parcourir.

- Deuxièmement, tous les sommets sont de poids égal, ce qui fait que déplacer le sommet en tête d'une liste chaînée de gain est équivalent du point de vue de l'équilibrage à déplacer n'importe quel autre sommet de la liste.

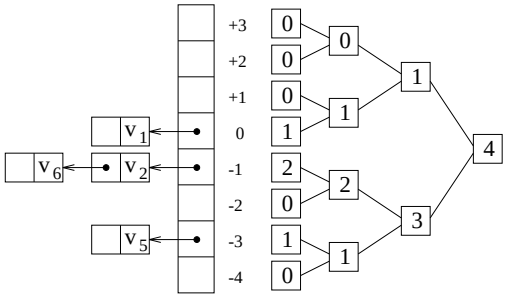
Ceci n'est plus vrai lorsque les sommets et arêtes sont valués, et que le gain prend en compte la dilatation des arêtes. Les valeurs de gain couvrent alors un grand intervalle, et déplacer un sommet du sous-domaine le plus chargé vers le sous-domaine le moins chargé peut augmenter le déséquilibre au lieu de le réduire (c'est par exemple le cas en déplaçant le sommet de poids 3 dans l'affectation non équilibrée de la figure 11.7).

La gestion de grands gains pose deux problèmes.

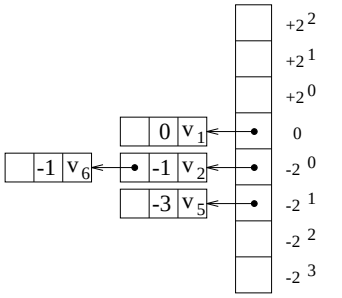
- Le premier est celui du temps d'accès à la liste chaînée de plus grand gain. Pour le résoudre, nous avons testé l'ajout à chaque tableau de gain d'une structure pyramidale donnant le nombre de sommets présents dans chaque sous-arbre (voir figure 11.9.a). Grâce à elle, la recherche de la liste chaînée de plus grand gain s'effectue en un temps logarithmique par rapport à la taille du tableau.
  Comme la taille du tableau est majorée par la plus grande adresse mémoire représentable en machine, le temps de recherche de la liste est majoré par le nombre de bits d'adresse de la machine sur laquelle est lancée le placeur. Ce nombre est une constante architecturale petite (inférieure à 64 sur toutes les machines construites à ce jour), ce qui donne un temps de recherche constant.

- Cependant, la solution ci-dessus ne fait qu'amplifier le deuxième problème, qui est celui de l'occupation mémoire. Comme la plus grande valeur de gain possible lors du placement d'un graphe S sur un graphe T peut atteindre $\mathrm{diam}(T)\,\Delta(S) \max_{e_S \in E(S)} w(e_S)$, la taille des tableaux de gains peut devenir prohibitive pour les graphes valués.

Pour conserver à la table de gain une taille raisonnable garante d'un accès rapide en temps quasi-constant, nous avons finalement opté pour une indexation logarithmique des listes de gains, illustrée en figure 11.9.b. Nous justifions notre choix par l'argumentation suivante. Les différences de comportement entre l'algorithme initial et celui avec table logarithmique semblent mineures, puisque dans les deux cas les sommets de plus grands gains sont traités avant ceux de petits gains. Lorsque les sommets regroupés dans une même liste d'index logarithmique ne sont pas voisins, en déplacer un ne modifie pas le gain des autres, et l'ordre de leur déplacement n'a pas d'importance vis-a-vis du type d'indexation. S'ils sont voisins, le choix d'un sommet plutôt que d'un autre peut éventuellement modifier les choix ultérieurs. Cependant, l'approximation qui en résulte nous semble du même ordre que celle inhérente à l'heuristique de Fiduccia et Mattheyses. De plus, lorsque la



a. Tableau de gain d'indice linéaire flanqué d'une structure pyramidale sommant les cardinalités des listes. Le parcours des branches non-nulles les plus à gauche permet de trouver la liste non-vide de plus grand gain.



b. Tableau de gain d'indice logarithmique par rapport au gain des sommets. Une liste d'index strictement positif i contient les sommets dont le gain est compris entre $\lfloor 2^{i-2}+1 \rfloor$ et 2^{i-1}.

Figure 11.9: Les deux structures testées permettant de rechercher un élément de (plus) grand gain en temps constant.

bipartition initiale est déjà structurée en fonction de la topologie du graphe (ce qui est le cas avec notre méthode GPS), la zone où s'effectuent les choix est réduite, et risque peu d'être déplacée.
Par définition, la dimension du tableau d'indice logarithmique est bornée par le nombre de bits codant l'entier de plus grande taille, qui ne dépasse actuellement pas 256 sur toutes les machines existantes. Le temps d'accès à la liste non-vide de plus grand gain est donc constant par rapport aux variables du problème. De plus, en mémorisant le plus grand index de liste utilisé par les opérations d'insertion, on évite le parcours systématique du tableau à chaque recherche.
Les expérimentations que nous avons menées pour valider notre choix sont présentées en section 12.3.4.

Comme aucune structure de données ne pouvait éviter le traitement des sommets dont le poids provoquerait un déséquilibre, et comme nous ne voulions pas parcourir toute la liste de plus grand gain pour rechercher le sommet générant le meilleur équilibrage, nous avons choisi de ne parcourir la liste que jusqu'à ce qu'un sommet dont le mouvement soit légal soit trouvé.

11.4.4.3 Évaluation de la complexité de l'algorithme amélioré

Soit S' le sous-graphe de processus passé à l'algorithme. Les structures de données internes à l'algorithme occupent un espace de complexité $O(|V(S')| + |E(S')|)$, et sont initialisées en un temps $O(|V(S')| + |E(S')|)$. Comme les seules modifications des structures de données au cours de l'exécution de l'algorithme sont des modifications de chaînage, $\mathcal{C}_E$ (FMA(S')) est en $O(|V(S')|+|E(S')|)$.

Le corps de l'algorithme consiste en k_P itérations de l'étape de minimisation de coupe. Durant une telle itération, k_V sommets sont choisis pour construire l'ordonnancement, la sélection d'un sommet prenant k_G pas de temps constant. Dans le pire des cas, k_G peut être égal au nombre de sommets non encore choisis, et k_V au nombre total de sommets. Dans ce cas, toutes les arêtes du

sous-graphe sont parcourues pour propager successivement les gains à tous les voisins des sommets. On obtient alors un processus de construction de l'ordonnancement en temps $O(|V(S')|^2+|E(S')|)$. De même, l'algorithme ne termine que si la valeur de la fonction de coût n'a pas été améliorée au cours d'une étape. Comme le gain peut ne diminuer que de 1 à chaque fois, k_P est majoré par $|V(S')|\,\Delta(S') \max_{e_{S'} \in E(S')} w(e_{S'})$, qui est la plus grande valeur possible de la fonction de coût f'_C. On en déduit que $\mathcal{C}_T\,(\mathrm{FMA}(S')) = O\left(\left(|V(S')|^2 + |E(S)|\right)\ |V(S')|\,\Delta(S') \max_{e_{S'} \in E(S')} w(e_{S'})\right)$.
Fort heureusement, lors de la sélection des sommets, les sommets candidats sont rapidement trouvés lorsque peu d'entre eux ont encore été choisis, et peu de sommets doivent être parcourus lorsque presque tous ont déjà été choisis. De fait, nous avons trouvé k_G petit sur tous nos cas de test, avec une valeur moyenne mesurée égale à 30, et un k_V proche de $|V(S')|$. Le comportement en temps de la construction de l'ordonnancement est donc en $O(|V(S')| + |E(S')|)$ pour chacune des k_P itérations. Les mêmes résultats expérimentaux montrent que l'algorithme converge très rapidement. En effet, sur toutes nos expériences, nous avons trouvé k_P inférieur à 20, avec une moyenne de 3. De ce fait, $\mathcal{C}'_T\,(\mathrm{FMA}(S'))$ est en $O(|V(S')| + |E(S')|)$.

11.4.5 L'algorithme de chaînage arrière ("*Backtracking*") (BT)

Le principe de cet algorithme a été présenté en section 8.2.1. Cette méthode ne devant être appelée qu'en post-traitement après qu'une bonne solution ait déjà été calculée, nous avons simplement implanté la variante d'exploration en profondeur.

Comme le temps d'exécution de cette méthode est très long et que sa complexité en temps pénalise fortement le déroulement de l'algorithme de placement, son utilisation a été limitée par défaut aux bipartitionnements manipulant moins de 32 processus. On la destine ainsi à l'affinage des placements aux feuilles de l'arbre de bipartitionnement, mais ceci n'est possible que lorsque l'ensemble de processus n'est pas beaucoup plus grand que l'ensemble de processeurs.

Pour tout graphe S', $\mathcal{C}_E\,(\mathrm{BT}(S'))$ est en $O(|V(S')|)$, et $\mathcal{C}_T\,(\mathrm{BT}(S'))$ est en $O(2^{|V(S')|})$.

11.4.6 L'algorithme glouton de réduction de déséquilibre (AGRD)

Cet algorithme glouton a pour but de réduire le déséquilibre de charge pouvant exister entre les deux sous-domaines, en augmentant le moins possible le coût de communication.
Son principe n'est pas sans ressemblance avec celui de l'algorithme glouton de bipartitionnement présenté au paragraphe 11.4.2 : on parcourt l'ensemble des sommets dans l'ordre des poids décroissants, en déplaçant les sommets rencontrés si cela permet de réduire le déséquilibre de charge (les sommets choisis appartiennent donc toujours au sous-domaine le plus chargé). S'il existe plusieurs candidats de même poids et appartenant à la même partie, ceux-ci sont insérés dans une structure de table de gain analogue à celles employées dans notre heuristique de Fiduccia et Mattheyses Améliorée. On déplace alors les sommets par ordre de gain décroissant, tant qu'il en résulte une

réduction du déséquilibre. Dans le cas contraire, le tableau est purgé et l'algorithme se poursuit en considérant les sommets de poids immédiatement inférieurs.

Cette méthode est utilisée en post-traitement des heuristiques de bipartitionnement lorsque l'obtension de sous-domaines équilibrés est primordiale. En particulier, elle est nécessaire lorsqu'on effectue la décomposition d'un graphe d'architecture par plongement sur le graphe complet de même ordre, afin qu'un sommet du graphe d'architecture soit affecté à chaque sommet du graphe complet. De même, lors du placement du graphe de machine sur la chaîne à deux sommets, elle permet d'obtenir sur chaque sommet du graphe cible le nombre exact de processeurs demandés (voir section 11.3.2, page 86).

Les structures de données étant identiques à celles utilisées par la méthode FMA, $\mathcal{C}_E$ (AGRD(S')) est en $O(|V(S')|+|E(S')|)$. Puisque chaque sommet n'est considéré qu'une seule fois, $\mathcal{C}_T$ (AGRD(S')) est en $O(|V(S')|)$.

11.4.7 Stratégies de placement et comportement expérimental

Comme nous l'avons dit précédemment, les différentes méthodes de bipartitionnement utilisées peuvent être combinées pour donner ce que nous appelons des stratégies. Les stratégies de placement actuellement implantées sont les suivantes :

- *FM* : appel à l'algorithme glouton de bipartitionnement (appelé par défaut), suivi de l'heuristique de Fiduccia et Mattheyses ;
- *FM+EX* : stratégie précédente, suivie de l'algorithme glouton de réduction de déséquilibre ;
- *GP+FM* : appel à la méthode GPS, suivi de l'heuristique de Fiduccia et Mattheyses ;
- *GP+FM+EX* : stratégie précédente, suivie de l'algorithme glouton de réduction de déséquilibre. C'est la stratégie par défaut de notre programme ;
- *RA+FM+EX* : appel à l'algorithme aléatoire de bipartitionnement, suivi de l'heuristique de Fiduccia et Mattheyses puis de l'algorithme glouton de réduction de déséquilibre.

Dans nos tests, notre programme sera référencé par "DRB{*nom de stratégie*}".

Toutes les méthodes de bipartitionnement de graphes que nous avons étudiées ont des comportements en temps et en espace en $O(|E(S')|+|V(S')|)$ (à l'exception de la méthode BT, dont nous avons donc restreint l'utilisation à de petits cas). Par conséquent, toutes les stratégies implantées dans notre programme ont des comportements en temps et en espace en $O(|E(S')|+|V(S')|)$.
Les méthodes algorithmiques de bipartitionnement de domaines que nous avons présentées ont un comportement constant en temps et en espace, et génèrent des bipartitionnements équilibrés. Il en va de même des fonctions d'accès aux décompositions de domaines obtenues par placement.

Par analogie avec les propositions 10.39 et 10.40, on peut donc énoncer les deux propositions suivantes.

Proposition 11.41. *$\mathcal{C}'_E\,(DRB(S,T))$ est en $O(|E(S)|)$.*

Remarque. Lorsque la topologie de la machine cible est décrite par fichier, il faut prendre en compte l'espace de stockage de la matrice des distances, qui est en $O(|V(T)|^2)$. Dans ce cas, $\mathcal{C}'_E\,(\mathrm{DRB}(S,T))$ est en $O(|E(S)| + |V(T)|^2)$.

Proposition 11.42. *$\mathcal{C}'_T\,(DRB(S,T))$ est en $O(|E(S)|\,\lceil \log_2(|V(T)|)\rceil)$.*

La section 12.3.1 est consacrée à la vérification expérimentale de ces résultats.

Chapitre 12

Évaluation des performances

12.1 Critères de performance

La qualité d'un placement est liée au respect des critères que nous avons choisis : équilibrage de la charge de calcul entre les processeurs, et minimisation du coût de communication inter-processeurs modélisé par la fonction f_C. Pour évaluer en fonction de ceux-ci les placements obtenus, nous définissons les paramètres suivants.

- L'équilibrage de charge est mesuré au moyen des trois paramètres quantitatifs $\min_{map}$, $\max_{map}$, et μ_{map}, et du paramètre qualitatif δ_{map}, définis ci-dessous :

$$\min_{map} \stackrel{\text{def}}{=} \min_{v_T \in V(T)} c(v_T) \quad , \qquad \max_{map} \stackrel{\text{def}}{=} \max_{v_T \in V(T)} c(v_T) \quad ,$$

$$\mu_{map} \stackrel{\text{def}}{=} \frac{\sum_{v_T \in V(T)} c(v_T)}{|V(T)|} = \frac{\sum_{v_S \in V(S)} w(v_S)}{|V(T)|} \quad , \qquad \delta_{map} \stackrel{\text{def}}{=} \frac{\sum_{v_T \in V(T)} (c(v_T) - \mu_{map})}{\mu_{map}\,|V(T)|} \quad .$$

  $\min_{map}$, $\max_{map}$, et μ_{map} sont respectivement les charges de calcul minimale, maximale, et moyenne placées sur les processeurs de l'architecture cible. δ_{map}, à valeur dans $[0;2[$, est l'écart relatif à la moyenne des charges des processeurs, et mesure donc le déséquilibre de charge moyen du placement. Il est égal à 0 si le placement est idéalement équilibré, et tend vers 2 à mesure que le placement devient déséquilibré. Remarquons que μ_{map} ne dépend pas du placement, mais seulement des graphes source et cible.

- Le coût de communication d'un placement est mesuré par le paramètre μ_{exp}, coût moyen de communication, défini comme la somme des produits du coût des arêtes du graphe source par leur dilatation, divisée par leur nombre. On peut également considérer μ_{dil}, dilatation moyenne des canaux du graphe source résultant du placement sur le graphe cible :

$$\mu_{exp} \stackrel{\text{def}}{=} \frac{\sum_{e_S \in E(S)} (w(e_S)\,|\rho_{S,T}(e_S)|)}{\sum_{e_S \in E(S)} w(e_S)} = \frac{f_C}{\sum_{e_S \in E(S)} w(e_S)} \quad , \qquad \mu_{dil} \stackrel{\text{def}}{=} \frac{\sum_{e_S \in E(S)} |\rho_{S,T}(e_S)|}{|E(S)|} \quad .$$

Le rapport de ces deux quantités, noté $\delta_{exp} = \frac{\mu_{exp}}{\mu_{dil}}$, est un paramètre qualitatif du comportement du placement permettant d'évaluer la localité des communications. Sa valeur appartient à $]0; +\infty[$, et est inférieure à 1 si l'algorithme de placement a réussi à placer les processus qui communiquent beaucoup entre eux plus près les uns des autres que les processus qui communiquent peu. Il est de fait égal à 1 si toutes les arêtes ont même poids. Par extension, nous lui attribuons la valeur 0 lorsque μ_{dil} est égal à 0, car ceci implique que μ_{exp} vaut également 0, ce qui est l'idéal de la minimisation de la communication.

12.2 Jeux de test

Les graphes sources utilisés pour tester notre programme appartiennent à trois classes distinctes.

- La première est constituée de graphes homogènes de topologies classiques : grilles multidimensionnelles, hypercubes, graphes de De Bruijn.

- La seconde est composée de maillages triangulaires utilisés dans la résolution par méthodes d'éléments finis de problèmes de dynamique des fluides ou de mécanique des structures. Les calculs par éléments finis étant identiques en tous les points de tels graphes, ceux-ci sont tous homogènes.
  Le graphe *efBump*, représenté en figure 12.1, est le maillage d'une surface déformée par un choc. Les graphes *ef3elt* (figure 12.2) et *ef4elt* sont des maillages irréguliers bidimensionnels de l'espace environnant des sections d'ailes d'avion composées respectivement de trois et quatre éléments (corps de l'aile et volets). *efEqu* est la description géométrique tridimensionnelle d'une pièce en forme d'équerre, et *efRotor* est celle du moyeu d'un rotor d'hélicoptère. Enfin, le graphe *efPwt* est le maillage tridimensionnel du conduit torique d'une soufflerie d'essais aérodynamiques. Ce dernier maillage nous a été fourni par Horst Simon afin que nous puissions comparer nos résultats aux siens, et ne sera utilisé par nous qu'à cette occasion. Ce maillage présente une imperfection, car l'un de ses sommets est déconnecté des autres ; son degré minimal est donc nul.

- La troisième classe contient des graphes de communication interprocessus correspondant à une implantation parallèle d'un solveur de matrices creuses par factorisation de Cholesky par blocs [15].
  Le graphe *deInit* représente la partition des inconnues d'une matrice induite par une méthode de dissections emboîtées. Les graphes *deRaf1*, *deRaf2*, *deRaf3*, et *deRaf4* sont obtenus successivement à partir de celui-ci par décomposition des processus les plus lourdement chargés en sous-graphes de processus plus légers, afin d'obtenir une meilleure granularité du problème, au prix de la création de sommets et –surtout– d'arêtes supplémentaires. Comme ces graphes sont construits par décomposition par blocs du système linéaire à résoudre, les coûts de calcul et de communication des blocs (mesurés respectivement en nombre d'opérations numériques et de données à transmettre entre blocs) sont exactement connus.

Les caractéristiques de tous ces graphes sont récapitulées dans le tableau 12.1.

Nom	Famille	Sommets	Arêtes	δ	Δ	min $(w(v))$	max $(w(v))$	min $(w(e))$	max $(w(e))$
$M(d_0, d_1)$	M2	$d_0 * d_1$	$2d_0d_1 - d_0 - d_1$	2	4	1	1	1	1
UB$(2, d)$	UB	2^d	$2^{d+1} - 3$	2	4	1	1	1	1
H(d)	HY	2^d	$2^{(d-1)}$	d	d	1	1	1	1
ef3elt	E.F. 2D	4720	13722	3	9	1	1	1	1
ef4elt	E.F. 2D	15606	45878	3	10	1	1	1	1
efBump	E.F. 2D	9800	28989	3	8	1	1	1	1
efEqu	E.F. 3D	62631	366559	3	32	1	1	1	1
efRotor	E.F. 3D	99617	662431	5	125	1	1	1	1
efPwt	E.F. 3D	36519	144794	0	15	1	1	1	1
deInit	D. E.	2047	7750	2	186	167	686298	6	4560
deRaf1	D. E.	2453	47659	2	444	167	270419	1	1403
deRaf2	D. E.	2815	84406	2	542	167	83652	1	1104
deRaf3	D. E.	3093	105713	2	584	167	38749	1	444
deRaf4	D. E.	3470	135148	2	633	167	25910	1	264

Table 12.1: Caractéristiques des graphes sources utilisés pour les tests.

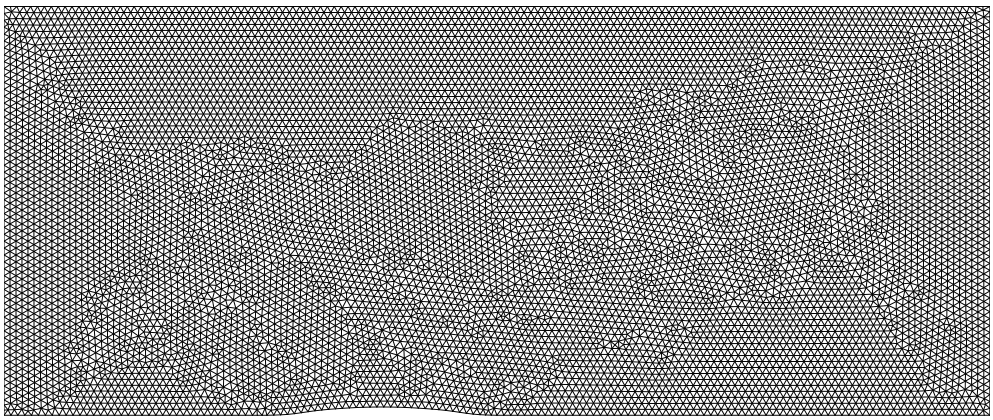
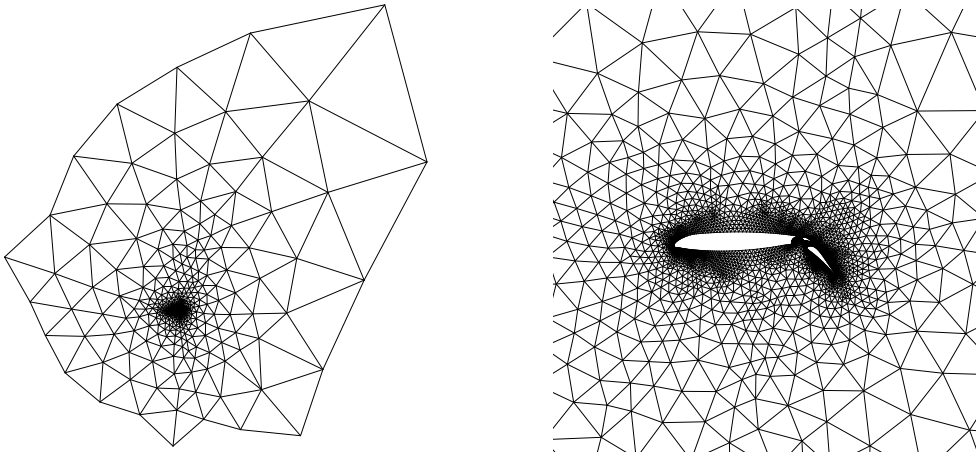


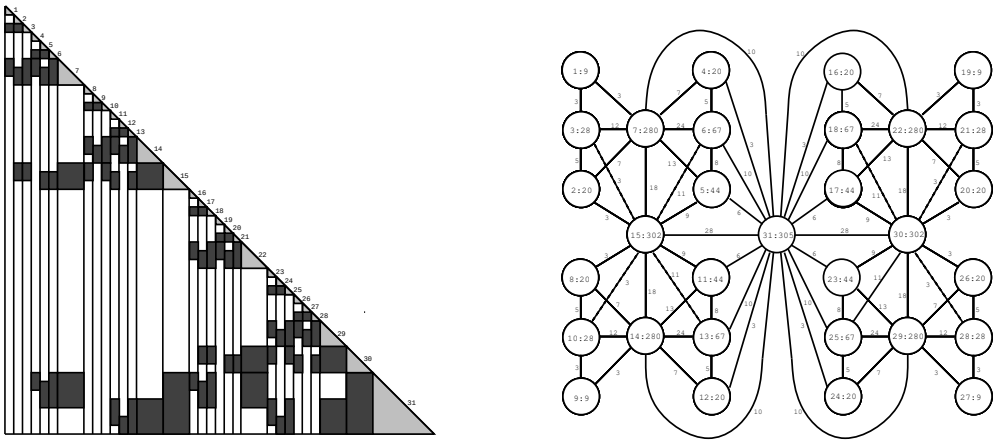
Figure 12.1: Vue d'ensemble du maillage *efBump*.



a. Vue d'ensemble du maillage.

b. Détail de la zone centrale.

Figure 12.2: Vues d'ensemble et de détail du maillage *ef3elt*.



a. Système linéaire creux factorisé par blocs.

b. Graphe de calcul valué associé.

Figure 12.3: Génération du graphe de calcul valué correspondant au programme parallèle de résolution d'un système linéaire creux par factorisation de Cholesky par blocs.

12.3 Analyse des performances

La machine utilisée pour effectuer l'ensemble de nos tests est une SUN 4/90 dotée de 32 Mo de mémoire centrale et de 64 Mo de disque de "*swap*". Tous les programmes ont été compilés avec `gcc -O`.

Les temps mesurés sont les temps CPU totaux (utilisateur et système) pris par le calcul des placements, à l'exclusion des temps de chargement des données et de sauvegarde des résultats.

Sauf mention explicite, tous les tests ont été effectués en appliquant un séquencement par niveaux.

Pour tous les bipartitionnements, la tolérance d'équilibrage de charge est fixée à 0.005 fois la charge effective moyenne.

12.3.1 Temps d'exécution

L'étude menée sur le corps de l'algorithme et sur les différentes méthodes de bipartitionnement de processus prédit un comportement en temps du placeur en $O(|E(S)|\ \log_2(|V(T)|))$. Afin de vérifier expérimentalement ce résultat, deux séries de tests ont été menées.

La première concerne la linéarité en temps par rapport au nombre d'arêtes du graphe source. Pour vérifier celle-ci, nous avons placé sur plusieurs topologies les graphes appartenant aux différentes familles de graphes sources. Les résultats, dont certains sont représentés en figures 12.4, 12.5, et 12.6, confirment nos prévisions.

En effet, les courbes obtenues, représentées ici avec une échelle logarithmique en abscisses et en ordonnées, sont pratiquement linéaires. Remarquons qu'à chaque famille de graphes est associée une pente caractéristique. Les graphes de De Bruijn et les grilles bidimensionnelles sont à ce point de vue très proches les uns des autres. Notons encore à ce propos la différence de nature existant

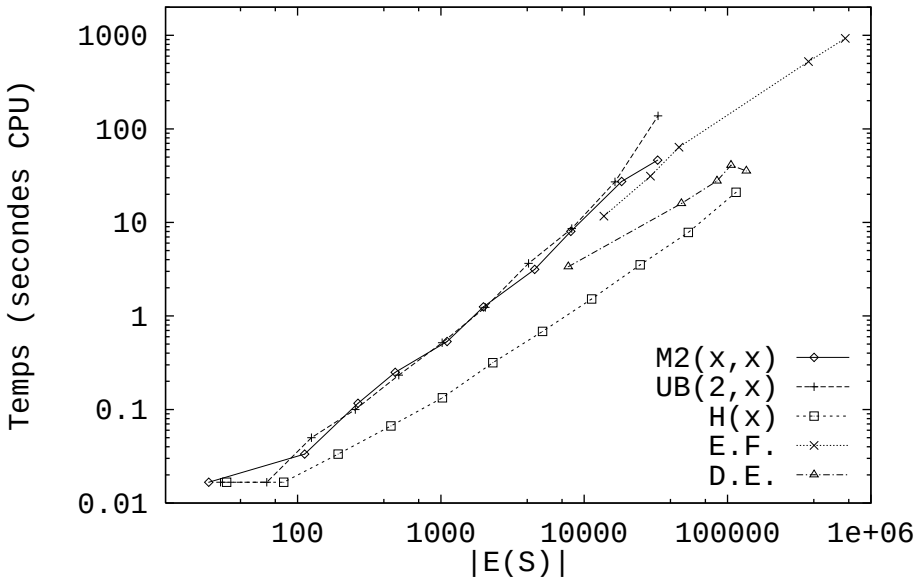


Figure 12.4: Temps d'exécution du placement sur H(4) avec la stratégie FM.

entre les graphes d'éléments finis en dimensions deux et trois. Bien que nous les ayons groupés au sein de la même courbe, les premiers (situés le plus à gauche de la courbe) semblent avoir une pente plus élevée que ces derniers (situés à l'extrême droite des figures).

La deuxième série de tests avait pour but de vérifier que le temps d'exécution évolue comme le logarithme en base deux du nombre de processeurs du graphe cible. Pour ce faire, nous avons placé certains graphes sources sur des graphes cibles de tailles croissantes. Nous dessinons les courbes de mesures obtenues avec une échelle logarithmique sur l'axe des abscisses et une échelle linéaire sur l'axe des ordonnées, afin qu'un comportement logarithmique se traduise par une courbe rectiligne. Pour toutes les familles de graphes cibles étudiées (hypercubes, grilles bidimensionnelles, graphes de De Bruijn), les courbes de résultats sont de formes analogues à celles de la figure 12.7. Ces courbes présentent toutes une section initiale droite, suivie d'un coude, et d'une autre section droite, infinie, comme le montre de façon parfaite la figure 12.8.a. L'explication de ce phénomène est la suivante.

- La première section droite de la courbe correspond bien au comportement logarithmique que nous attendions. En effet, dans ce cas de figure, chaque tâche de bipartitionnement gère de nombreux processus et se scinde en deux sous-tâches qui se les répartissent équitablement. De fait, le temps d'exécution d'une sous-tâche est à peu près égal à la moitié de celui de sa tâche père, le nombre de sous-tâches doublant à chaque niveau. On trouve donc bien le comportement logarithmique souhaité. Ce régime de fonctionnement est illustré par la partie haute de la figure 12.8.b.

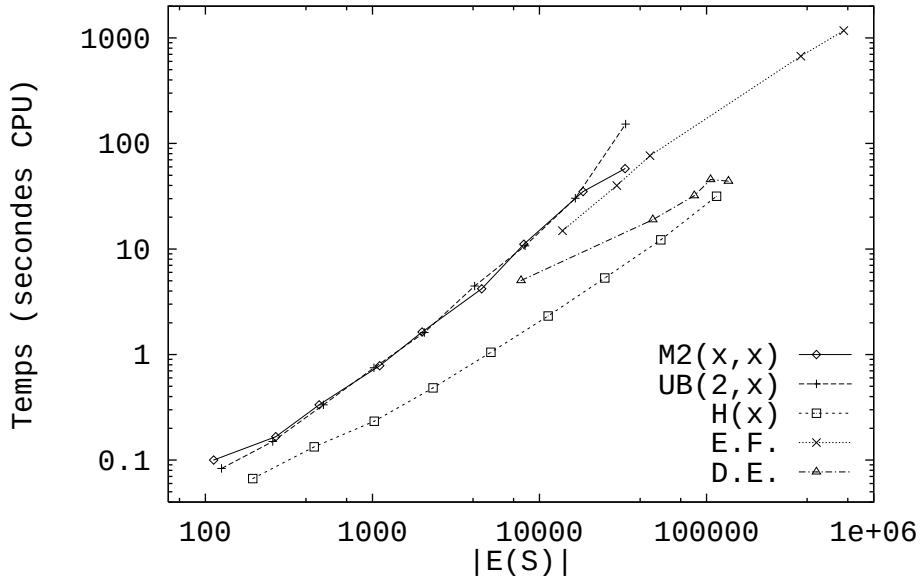


Figure 12.5: Temps d'exécution du placement sur $M_2(8,8)$ avec la stratégie FM.

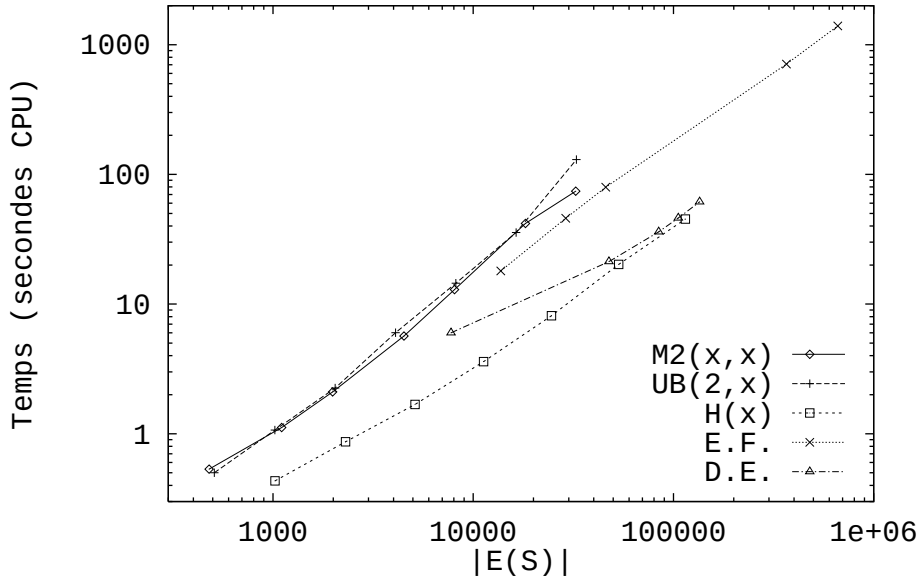


Figure 12.6: Temps d'exécution du placement sur UB$(2,8)$ avec la stratégie FM (la décomposition de UB$(2,8)$ a été obtenue par placement préalable sur le graphe complet).

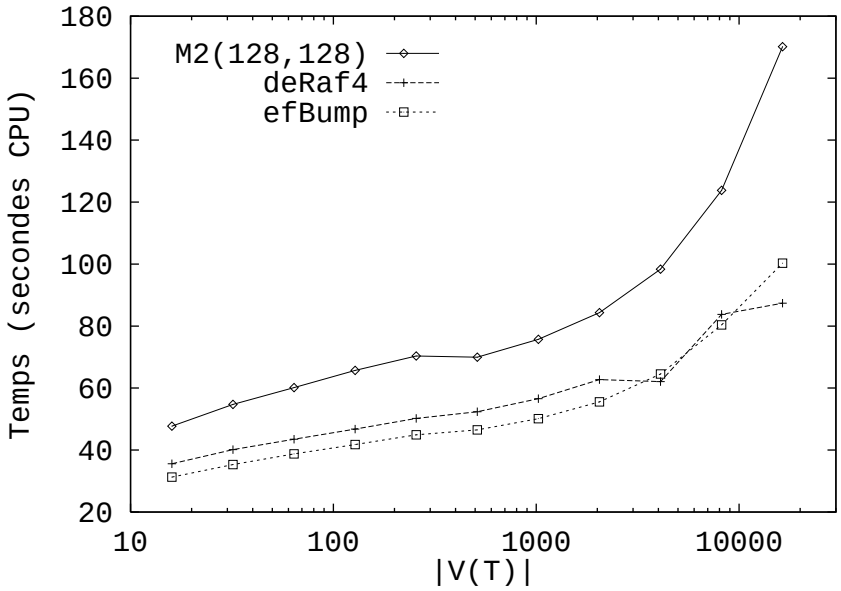


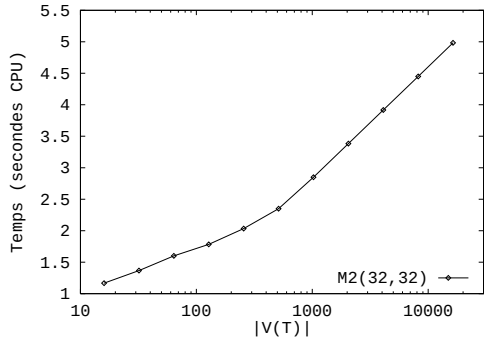
Figure 12.7: Temps d'exécution du placement sur H(x) avec la stratégie FM.

- Le coude apparaît lorsque le nombre de processeurs de l'architecture équivaut au nombre de processus à placer. Chaque tâche de bipartitionnement n'a alors à gérer qu'un tout petit nombre de processus, et le temps d'exécution de chacune d'elles est alors dominé par le temps pris par leur phase d'initialisation, qui est constant. Comme le nombre de tâches continue toujours à doubler, sans dépasser cependant le nombre de processeurs, on quitte le régime logarithmique pour se rapprocher du régime linéaire. Ceci est représenté dans la partie médiane de la figure 12.8.b.

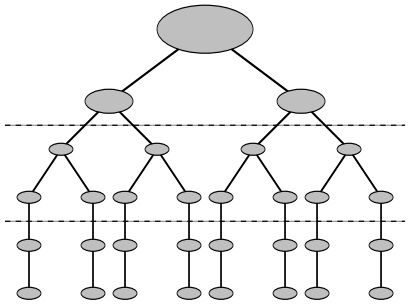
- La deuxième section droite apparaît lorsque le nombre de processeurs dépasse le nombre de processus. Dès le moment où chaque domaine à bipartitionner ne contient plus qu'un unique processus, le bipartitionnement s'effectue également en temps constant, mais un seul sous-travail est généré à chaque fois, comme indiqué dans section inférieure de la figure 12.8.b. On retrouve alors un comportement logarithmique par rapport au nombre de processeurs.

Le temps d'exécution de notre placeur évolue donc bien comme le logarithme du nombre de processeurs de l'architecture cible, dans la mesure où le temps d'exécution des tâches de bipartitionnement reste linéaire par rapport au nombre de processus qu'elles contiennent. Il ressort de ces tests qu'il n'est pas intéressant de placer un programme parallèle sur un domaine comportant plus de processeurs que le programme de processus, puisque cela génère des traitements inutiles. Mieux vaut dans ce cas bipartitionner au préalable le domaine de placement pour en limiter la taille au nombre de processus à placer, en utilisant par exemple la méthode que nous avons décrite page 86.

Remarquons que les temps d'exécution dépendent peu, à nombre de processeurs égal, de la



a. Temps d'exécution du placement de $M_2(32, 32)$ sur $H(x)$ avec la stratégie FM.



b. Schéma d'exécution en trois phases des tâches de bipartitionnement lorsque le nombre de processeurs est supérieur au nombre de processus.

Figure 12.8: Influence du nombre de processeurs sur le comportement en temps du programme de placement.

topologie du graphe cible. En revanche, à nombre de processus égal, le type du graphe source conditionne fortement le comportement de l'algorithme de Fiduccia et Mattheyses. En effet, la topologie du graphe source influe sur la manière dont les gains sont propagés dans le graphe lors du déplacement d'un sommet, et conditionne donc le déclenchement et l'entretien de cascades de mouvements.

12.3.2 Séquencements en profondeur et par niveaux

Nous avons décrit en section 10.3 les deux séquencements possibles des tâches de bipartitionnement. Pour évaluer leur influence sur la qualité du placement, nous avons mesuré l'expansion moyenne obtenue avec chaque séquencement lors du placement des graphes sources de test sur les familles de graphes cibles.

À quelques exceptions près, le séquencement par niveaux donne toujours de meilleurs résultats que le séquencement en profondeur. Notre interprétation est que le séquencement en profondeur revient à placer finement une partie du graphe source sur une partie de l'architecture, puis à terminer le placement en plaçant les autres parties du graphe source par rapport à la partie déjà placée. Si la première partie du placement est correcte, on peut donc placer finement les suivantes, et beaucoup réduire la communication.
Avec le séquencement par niveaux, on place l'ensemble du graphe source en même temps, en raffinant globalement le placement. Comme la fonction de distance ne donne pas de résultats initialement précis, le placement peut être moins efficace dans le cas de petits graphes cibles. À l'inverse, pour de grands graphes cibles, ce contrôle global du placement permet d'éviter de persévérer trop loin dans un mauvais choix (c'est l'effet de rétroaction que nous avons déjà évoqué).

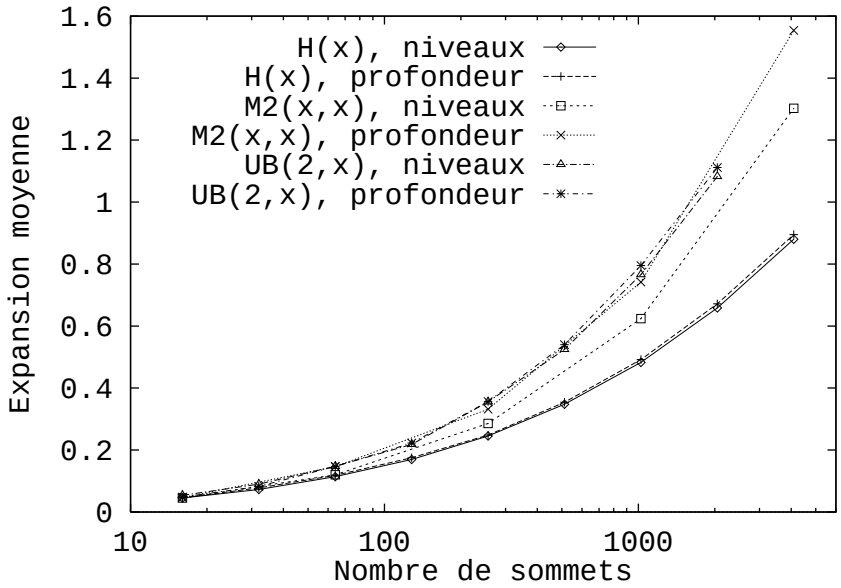


Figure 12.9: Expansion moyenne du placement du graphe *efEqu* sur $\mathrm{H}(x)$ et $\mathrm{M}_2(x,x)$ avec la stratégie FM et avec les séquencements par niveaux et en profondeur.

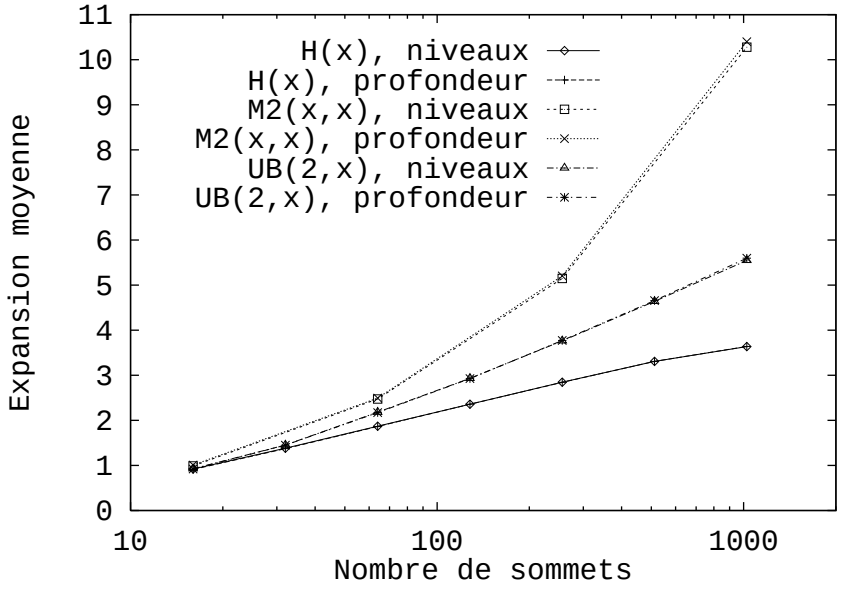


Figure 12.10: Expansion moyenne du placement du graphe *deRaf4* sur $\mathrm{H}(x)$ et $\mathrm{M}_2(x,x)$ avec la stratégie FM et avec les séquencements par niveaux et en profondeur.

Pour les graphes d'éléments finis (voir figure 12.9) placés sur l'hypercube et les graphes de De Bruijn, l'expansion des placements calculés en profondeur n'est supérieure que d'au plus 3 pourcents à celle calculée par niveaux, parce que la densité des liens de communication et le faible diamètre limitent les conséquences d'un mauvais choix. En revanche, le surcoût de communication peut atteindre 20 pourcents pour la grille bidimensionnelle, dont le grand diamètre par rapport au nombre de sommets pénalise fortement les mauvais placements.
Pour les graphes de dissections emboîtées, les différences entre les deux séquencements ne sont pas significatives (voir figure 12.10). Cela est dû selon nous à la très forte densité de ces graphes, qui atténue les disparités entre bons et mauvais placements.

Au vu de ces résultats, nous avons implanté le séquencement par niveaux par défaut, et tous les tests réalisés, à l'exception de ceux ci-dessus, l'ont été avec celui-ci.

12.3.3 Décomposition d'architectures

Nous avons présenté en section 11.1.3 une méthode permettant d'utiliser notre placeur pour calculer les arbres de bipartitionnement des architectures cibles. Nous étudions ici l'impact de la décomposition de l'architecture sur la qualité du placement.

Notre premier objet d'étude a été la grille bidimensionnelle, dont les décompositions peuvent être facilement représentées et interprétées.
La décomposition que nous y utilisons par défaut, et qui est implantée algorithmiquement dans le placeur, est la méthode de dissections emboîtées décrite en section 11.2.2. Cette implantation algorithmique a pour conséquence que, quand on bipartitionne un domaine carré, on coupe toujours selon la même dimension, ce qui donne, pour les grilles carrées dont le côté est une puissance de deux, une décomposition très régulière.
Lors d'une décomposition par placement, comme on ne considère à chaque étape que la localité des communications, la direction de la coupe d'un domaine carré n'est pas fixée, et l'on peut alors obtenir sur deux domaines carrés de même niveau des bipartitionnements d'orientation différente, comme le montre la figure 12.11. Lorsque la grille considérée n'a plus pour dimensions des puissances de deux, les résultats diffèrent encore plus : alors que la décomposition algorithmique s'effectue à une ligne près, la décomposition par placement respecte l'égalité à un près du nombre des sommets placés sur chaque sous-domaine (figure 12.12).

Afin de comparer ces deux décompositions, nous avons effectué le placement des graphes sources sur plusieurs grilles de tailles diverses, décomposées de ces deux manières.
Tous les résultats obtenus sont analogues à ceux présentés en figure 12.13. Si, pour les grilles dont les côtés sont des puissances de deux, la décomposition obtenue par placement donne des résultats comparables à ceux des dissections emboîtées, il n'en va pas de même pour celles dont les dimensions sont impaires.

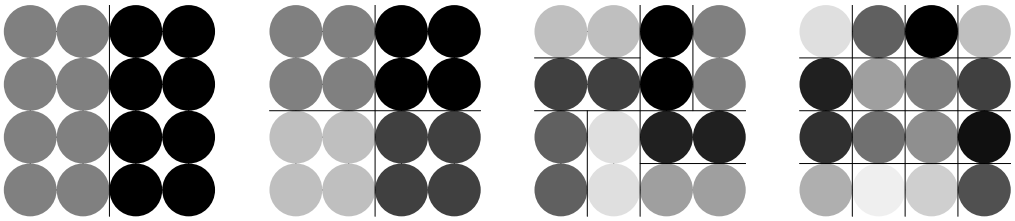


Figure 12.11: Étapes successives de la décomposition de l'architecture $M_2(4,4)$ par placement sur K(16) avec la stratégie FM+EX. À la différence de la méthode de dissections emboîtées, l'orientation des coupes des domaines carrés peut varier.

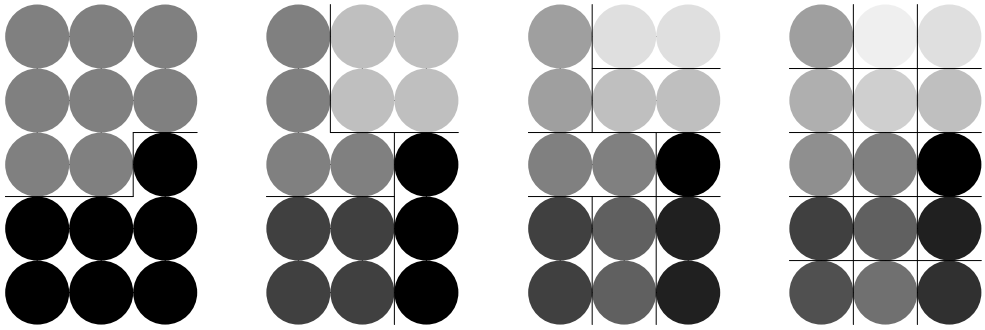


Figure 12.12: Étapes successives de la décomposition de l'architecture $M_2(3,5)$ par placement sur K(15) avec la stratégie FM+EX.

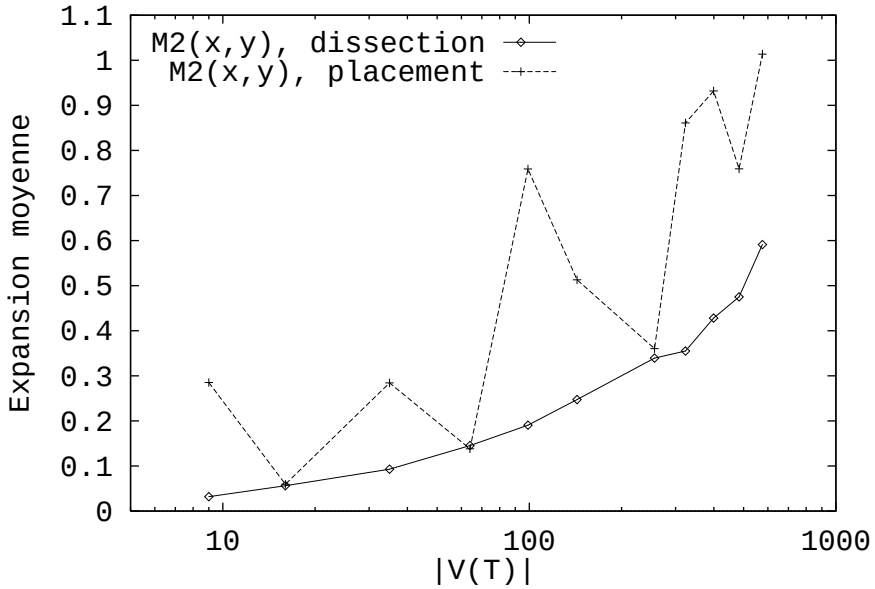


Figure 12.13: Expansion moyenne du placement du graphe *efBump* avec la stratégie FM sur des grilles bidimensionnelles $M_2(x,y)$ décomposées par dissections emboîtées et par placement.

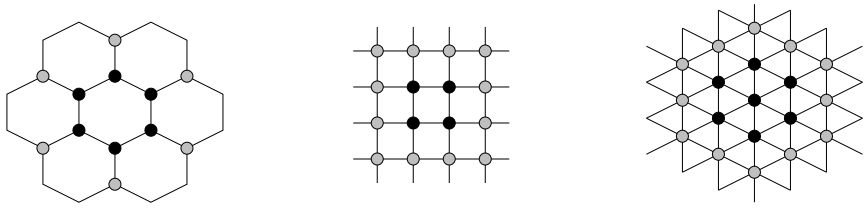


Figure 12.14: Plus petits ensembles stables pour différents types de maillages réguliers homogènes. Transformer un unique sommet noir en sommet gris n'améliore pas la fonction de coût, c'est-à-dire ne diminue pas le nombre d'arêtes dont les extrêmités sont de couleur différente.

L'inefficacité de la méthode de décomposition par placement pour les grilles bidimensionnelles a deux origines.

- La première provient de la difficulté de l'algorithme de Fiduccia et Mattheyses à manipuler les grilles bidimensionnelles en tant que graphe source. Pour appuyer notre propos, considérons le plus petit ensemble connexe de sommets d'un graphe alloués au même sous-domaine et tel que cet ensemble soit stable, c'est-à-dire que le déplacement d'un des sommets de l'ensemble n'améliore pas la fonction de coût. Si l'on étudie les plus petits ensembles stables pour différents types de maillages réguliers homogènes (voir figure 12.14), on voit que celui de la grille bidimensionnelle est petit comparativement aux autres. Avec la grille bidimensionnelle, les ensembles stables sont donc plus faciles à obtenir que pour les autres types de maillages, ce qui augmente les chances de l'algorithme d'être piégé dans des minima locaux de la fonction de coût. Ce comportement avait déjà été diagnostiqué par Gilbert et Zmijewski pour l'heuristique de Kernighan et Lin [42].

- La deuxième raison tient au mode de découpage que nous utilisons. Lorsqu'une grille de dimensions impaires est bipartitionnée pour la première fois, la séparation entre les deux domaines laisse apparaitre une "marche d'escalier" (voir figure 12.12). Au cours des bipartitionnements suivants, ce phénomène peut s'amplifier, et conduire à l'apparition de sous-domaines en forme de "L". Pour bipartitionner ceux-ci en minimisant la coupe, l'algorithme affecte à l'un des sous-domaines la section coudée du "L", et à l'autre ses deux extrémités. On obtient alors des sous-domaines non-connexes, qui perturbent le calcul de la fonction de distance et vont à l'encontre de notre principe de localité de la communication. Le découpage non-connexe des domaines en "L" est favorisé par la 4-connexité de la grille, pour laquelle découper selon une diagonale coûte deux fois plus cher que de couper selon une dimension.

Pour l'hypercube, les résultats sont beaucoup plus proches les uns des autres (voir figure 12.15). La première raison à cela est que la structure de l'hypercube induit toujours avec l'algorithme de Fiduccia et Mattheyses un découpage régulier selon une unique dimension (remarquons à ce propos que le plus petit ensemble stable de l'hypercube homogène $\mathrm{H}(x)$ est $\mathrm{H}\left(\left\lceil \frac{x}{2} \right\rceil\right)$, car tout sommet ayant moins de $\left\lceil \frac{x}{2} \right\rceil$ voisins appartenant au même sous-domaine que lui provoque une diminution de la coupe s'il est déplacé).
De plus, si les toutes les bipartitions d'un niveau ne s'effectuent pas nécessairement selon la même

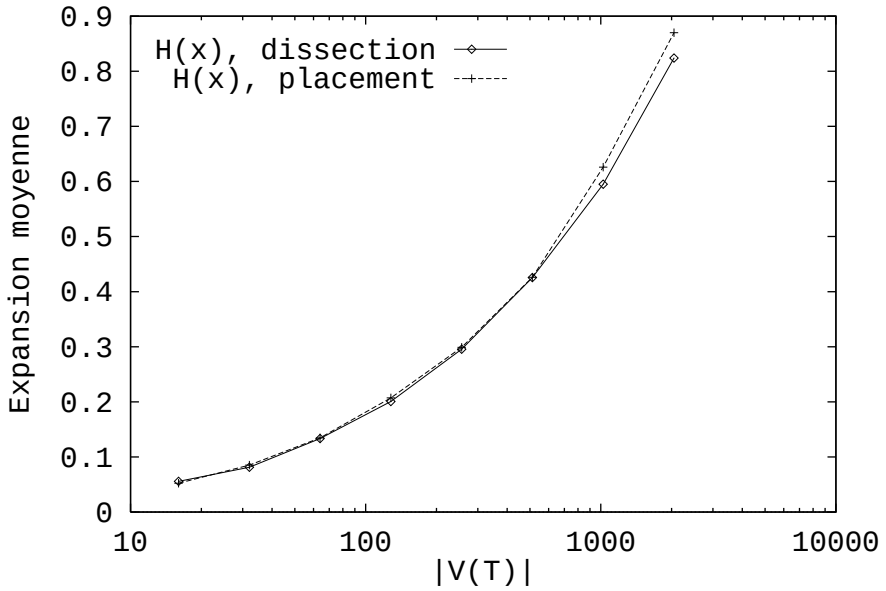


Figure 12.15: Expansion moyenne du placement du graphe *efBump* avec la stratégie FM sur des hypercubes décomposés par dissections emboîtées et par placement.

dimension, le fort degré de l'hypercube compense les disparités que l'asymétrie du découpage induit dans la fonction de distance.

Pour les graphes de De Bruijn, nous avons comparé la décomposition algorithmique décrite en section 11.2.4 à celle obtenue par placement. Comme on peut le voir en figure 12.16, la deuxième donne de bien meilleurs résultats que la première. Ceci était prévisible pour les raisons que nous avons déjà citées, qui sont que la décomposition algorithmique ne privilégie ni la localité des communication dans les sous-domaines, ni la minimisation de la coupe entre les sous-domaines.

Il ressort de nos tests que la décomposition d'architecture par placement peut donner de bons résultats si la décomposition obtenue respecte les propriétés de connexité et de minimisation de la coupe entre sous-domaines. Elle permet de traiter de manière automatique des graphes pour lesquels aucune décomposition simple n'est connue, tels les graphes de De Bruijn.
Les résultats obtenus sur la grille sont intéressants à plus d'un titre. Tout d'abord, ils montrent la forte corrélation existant entre la qualité de la décomposition du graphe cible et celle des placements obtenus. De plus, ils semblent indiquer que la conservation des propriétés topologiques des sous-domaines et la minimisation de la coupe sont préférables à la stricte exactitude de leurs cardinaux. Ensuite, ils confirment l'utilité des méthodes algorithmiques de bipartitionnement de domaines, qui permettent d'implanter des algorithmes de bipartitionnement spécifiques et optimisés en fonction des topologies de machines.

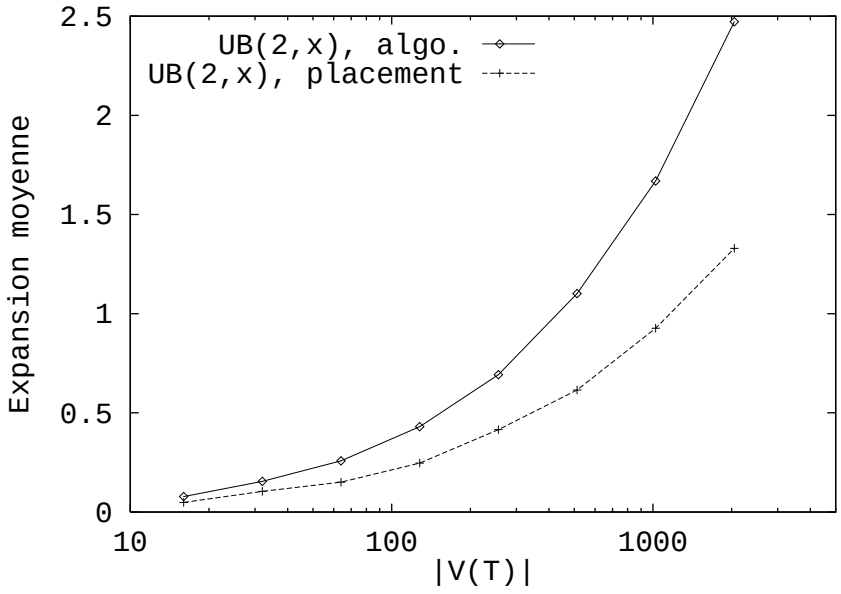


Figure 12.16: Expansion moyenne du placement du graphe *efBump* avec la stratégie FM sur des graphes de De Bruijn décomposés par la méthode algorithmique et par placement.

12.3.4 Améliorations de la méthode de Fiduccia et Mattheyses

Nous avons décrit en section 11.4.4.2 l'emploi de tableaux de gain indicés logarithmiquement pour stocker les sommets de grands gains. Pour valider ce choix, nous comparons les expansions résultant du placement des graphes de test sur différentes familles d'architectures, avec les tableaux de gains à indices linéaires et logarithmiques.
Pour avoir le plus d'éléments de comparaison possible, nous effectuons nos tests avec plusieurs stratégies, choisies de manière à donner à l'heuristique de Fiduccia et Mattheyses différentes bipartitions initiales. En revanche, pour ne pas perturber nos mesures, aucun post-traitement n'est effectué après l'appel à la méthode FMA.

Tous les résultats que nous obtenons sont analogues à ceux présentés en figure 12.17. Les expansions moyennes des placements sur l'hypercube et le graphe de De Bruijn calculés en utilisant l'une ou l'autre indexation diffèrent d'au plus 5 pourcents, dans un sens ou dans l'autre. Pour la grille bidimensionnelle, les écarts sont plus importants et peuvent atteindre 10 pourcents, car le grand diamètre de ce graphes amplifie les différences.
En fonction du graphe étudié, certaines stratégies conduisent à des résultats meilleurs dans le cas linéaire que dans le cas logarithmique, ou réciproquement. Ceci illustre la sensibilité de l'heuristique de Fiduccia-Mattheyses à la bipartition initiale, et montre que l'approximation que nous faisons en utilisant l'indexation logarithmique est du même ordre que celle inhérente à l'algorithme lui-même.

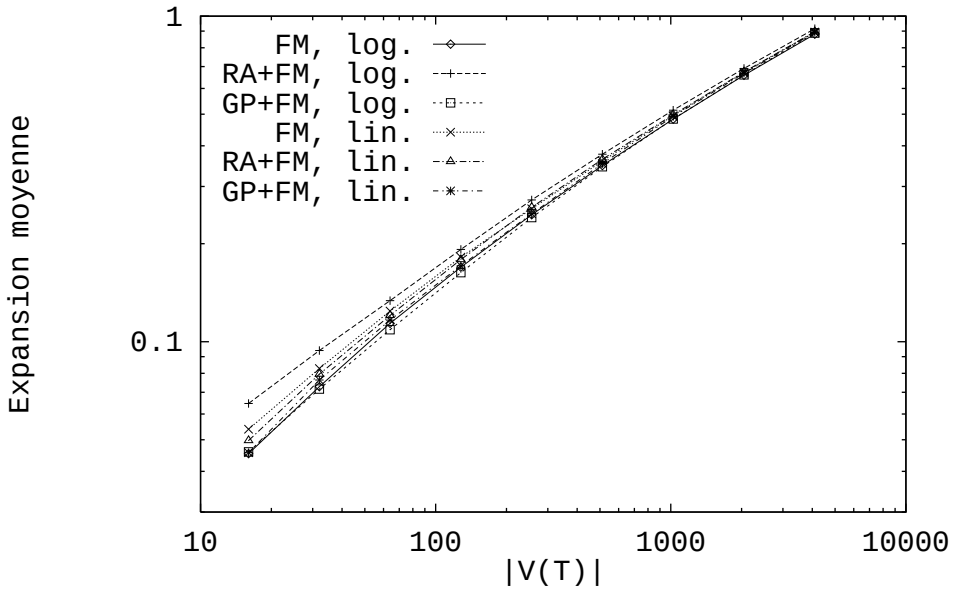


Figure 12.17: Placements du graphe *efEqu* sur l'hypercube au moyen de différentes stratégies, avec tableaux de gains à indices linéaires et logarithmiques.

Tous ces résultats indiquent que, du point de vue de l'heuristique de Fiduccia-Mattheyses, les deux structures de données sont équivalentes. Cependant, le graphe *deRaf4* n'a pu être placé sur la grille $M_2(32, 32)$ dans le cas linéaire, malgré l'allocation d'une table à 2^{19} éléments.
Ceci justifie donc pleinement le choix des tableaux à indexation logarithmique, qui n'influent pas sur la qualité des bipartitions obtenues, et réduisent considérablement l'espace occupé par les tableaux.

12.3.5 Comparaison des différentes stratégies

Nous avons énuméré en section 11.4.7 les stratégies de placement implantées dans notre programme. Afin d'étudier leurs caractéristiques, nous réalisons le placement de nos graphes de tests sur plusieurs topologies (hypercubes, grilles bidimensionnelles, et graphes complets) au moyen de chacune d'elles.
Les résultats obtenus diffèrent significativement selon le type de graphe source considéré. Nous présentons ci-dessous les résultats qualitatifs relatifs aux deux familles de graphes sources étudiées, que nous illustrons par des mesures effectuées avec une architecture cible hypercubique (voir figures 12.18, 12.19, 12.20, et 12.21).

Intéressons-nous tout d'abord aux graphes de dissections emboîtées. La figure 12.18 montre l'expansion moyenne obtenue lors du placement de *deRaf4* sur des hypercubes de tailles croissantes. On peut y remarquer les points suivants.

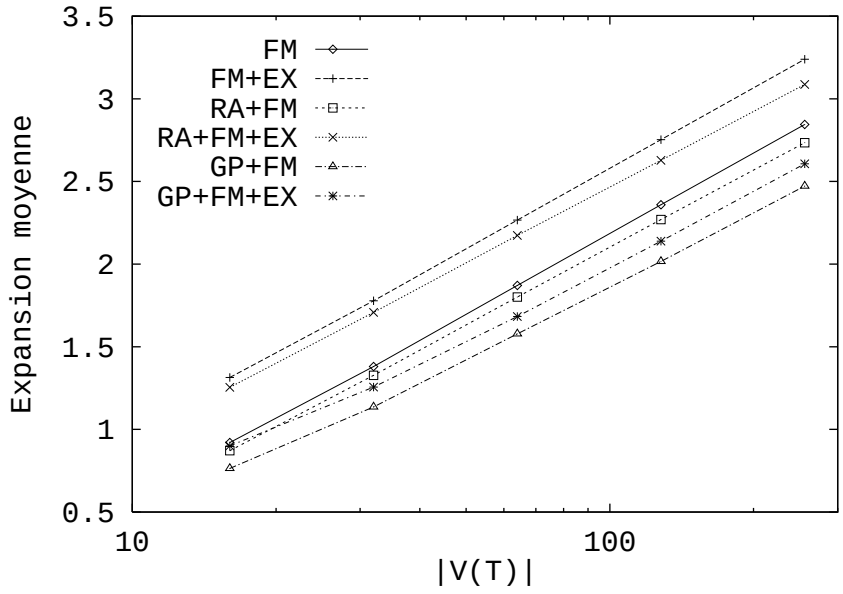


Figure 12.18: Expansion moyenne du placement du graphe *deRaf4* sur H(x) avec différentes stratégies.

- Dans tous les cas, l'ajout en post-traitement de la méthode AGRD diminue la qualité des placements obtenus. La raison en est que comme ces graphes sont de grand degré, très denses, et ont des valuations importantes sur leurs arêtes, la méthode FMA tend à minimiser la fonction de coût en déséquilibrant le graphe autant que possible, de manière à contracter le plus d'arêtes possible sur les mêmes processeurs. Par conséquent, forcer l'exactitude des cardinaux génère de gros surcoûts de communication.

- L'utilisation de la méthode GPS pour calculer la bipartition initiale des tâches améliore toujours la qualité des placements obtenus. La structure très fortement hiérarchisée des graphes de dissections emboîtées cadre ici très bien avec le groupement par couches, ce qui concourt à la minimisation de la communication.

- Les stratégies FM et FM+EX donnent respectivement de moins bons résultats que les stratégies RA+FM et RA+FM+EX. Ceci provient du fait que les bipartitions initiales calculées par la méthode AGB sont de moins bonne qualité pour les graphes de dissections emboîtées que celles calculées aléatoirement. En effet, comme les poids les plus forts sont situés au sommet de la structure pyramidale du graphe (voir figure 12.3.b), l'AGB parcourt cette structure niveau par niveau en plaçant les sommets sur l'un ou l'autre sous-domaine. Il en résulte une imbrication très forte des sous-domaines, qui génère une très grande coupe.

Les temps correspondant à ces placements sont représentés en figure 12.19. On y retrouve les conséquences des phénomènes que nous avons décrits ci-dessus.

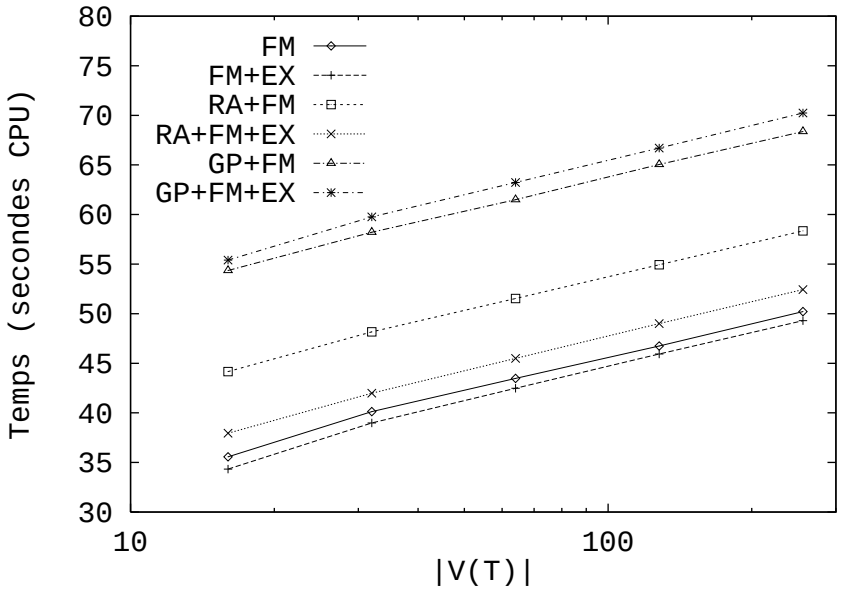


Figure 12.19: Temps d'exécution du placement du graphe *deRaf4* sur H(x) avec différentes stratégies.

- Les temps mis par les stratégies utilisant la méthode AGB sont plus courts, car les bipartitions initiales générées, moins bonnes que celles obtenues par les méthodes AAB ou GPS, limitent plus les possibilités de l'heuristique de Fiduccia et Mattheyses, qui termine donc plus tôt.
- Les stratégies employant la méthode GPS prennent plus de temps que les autres, ce qui est dû tant au phénomène précédent qu'au surcoût induit par l'exécution de la méthode. Celui-ci n'est cependant pas très important dans l'absolu.

Les résultats obtenus pour les graphes de maillage sont très différents des précédents du point de vue qualitatif. La figure 12.20 donne les résultats d'expansion obtenus lors du placement de *efRotor* sur des hypercubes de tailles croissantes. On peut y faire les constatations suivantes.

- Tout d'abord, on remarque que les stratégies de bipartitionnement utilisées n'ont que peu d'influence sur la qualité des placements obtenus. Ceci est dû aux propriétés topologiques des graphes maillés triangulaires, qui se prêtent bien à l'heuristique de Fiduccia et Mattheyses.
- Ensuite, à l'inverse de ce que nous avons observé pour les graphes de dissections emboîtées, l'appel à l'AGRD n'augmente pas l'expansion moyenne des placements. Ceci s'explique ici encore par les propriétés topologiques des graphes de maillage. Pour équilibrer un bipartitionnement, l'AGRD déplace les sommets qui pénalisent le moins la fonction de coût. Très souvent, ce sont ceux ayant déjà un ou plusieurs voisins dans l'autre sous-domaine, car plus un sommet possède de voisins dans l'autre sous-domaine, plus le gain de son déplacement est grand. L'équilibrage des sous-domaines se traduit alors par un léger déplacement de la coupe entre les deux sous-domaines, qui ne remet pas en cause la qualité du bipartitionnement.

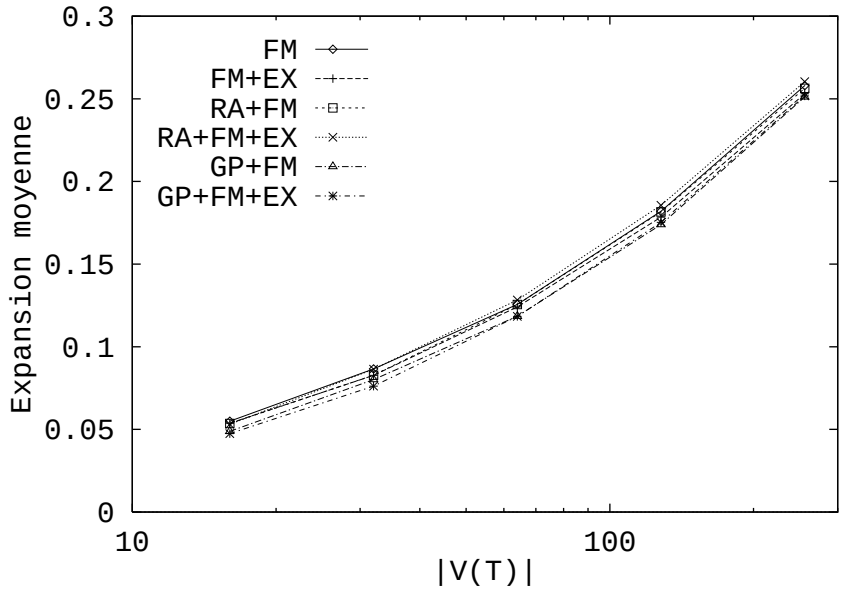


Figure 12.20: Expansion moyenne du placement du graphe *efRotor* sur H(x) avec différentes stratégies.

- De même, le calcul des bipartitions initiales par la méthode AGB n'augmente pas l'expansion moyenne des placements. En effet, comme les graphes sont homogènes, l'ordre dans lequel les sommets sont choisis par l'AGB peut être considéré comme aléatoire[†], ce qui donne des résultats analogues à ceux fournis par l'AAB.

Les résultats les plus intéressants sont les mesures de temps effectuées sur ces mêmes placements, qui sont dessinées en figure 12.21.

- Le premier résultat majeur est que le pré-traitement par la méthode GPS, s'il influe peu sur la qualité des placements, améliore grandement leur temps de calcul, de 20 à 30 pourcents selon les cas. Pour les graphes maillés, l'analogie géométrique avec les méthodes inertielles est très forte, et l'algorithme GPS donne des bipartitions initiales de très bonne qualité et de coupe faible, qui réduisent d'autant le travail de l'heuristique de Fiduccia et Mattheyses.

- Un autre résultat, plus surprenant, est l'influence de l'AGRD employé en post-traitement de l'heuristique de Fiduccia-Mattheyses. Si son emploi fait augmenter le temps de traitement entre les stratégies FM et FM+EX, il procure des gains en temps significatifs aux stratégies RA+FM+EX et GP+FM+EX, pouvant atteindre 15 pourcents. Remarquons à ce propos que la stratégie GP+FM+EX, qui est la plus rapide, donne aussi les meilleurs résultats.
  Nous retrouvons ce comportement sur tous les graphes de maillage tridimensionnels, mais

[†]Ce n'est en fait pas tout-à-fait le cas. L'ordre de parcours des sommets est celui dans lequel ils sont stockés sur fichier, qui n'est pas aléatoire car les logiciels de maillage procèdent souvent par raffinement spatial local. Deux sommets numérotés consécutivement sont donc souvent très proches l'un de l'autre.

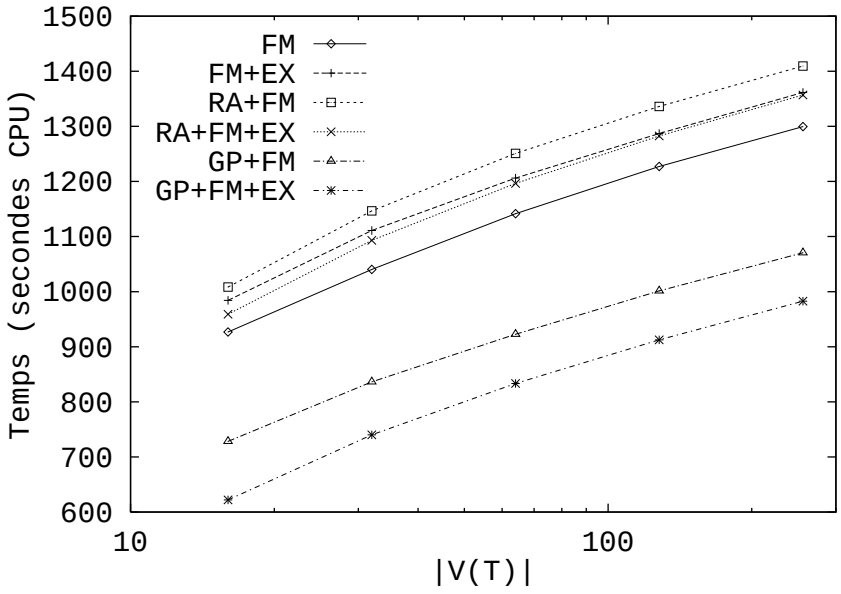


Figure 12.21: Temps d'exécution du placement du graphe *efRotor* sur H(x) avec différentes stratégies.

pas sur les graphes de maillage bidimensionnels, pour lesquels l'appel à la méthode AGRD fait légèrement augmenter le temps d'exécution. En tout état de cause, nous ne pouvons pas expliquer l'origine de ce phénomène.

Pour tous les graphes que nous avons étudiés, l'emploi de la méthode GPS a grandement amélioré l'efficacité de notre programme, que ce soit en terme d'expansion moyenne ou de temps de calcul. La grande supériorité de cette méthode sur les méthodes inertielles est qu'elle ne nécessite pas de connaître les coordonnées géométriques des sommets du graphe considéré, et peut donc être utilisée pour tout type de graphe.

12.3.6 Comparaison avec d'autres méthodes

Les sections précédentes ont été consacrées à l'évaluation des performances intrinsèques de notre programme. Nous cherchons maintenant à comparer les résultats que nous obtenons à ceux fournis par d'autres programmes de même nature.

La communauté des numériciens utilise depuis de nombreuses années des programmes de placement par décomposition de graphes pour répartir les calculs de différences et d'éléments finis associés aux sommets de graphes de maillage. Elle dispose de nombreux graphes de test répertoriés, facilement accessibles, et correspondant à de grands cas réels, qui servent de référence aux différents auteurs pour présenter et comparer les résultats de leurs placeurs.
Les méthodes de bipartitionnement actuellement les plus réputées dans cette communauté sont

les méthodes spectrales, que nous avons présentées dans l'introduction de cette partie. Dans cette section, nous comparons donc les résultats de notre programme avec ceux fournis par deux programmes utilisant la bissection spectrale.

12.3.6.1 L'Échange Cyclique par Paires

Hammond a proposé dans sa thèse [44] un algorithme de placement destiné principalement aux machines hypercubiques[‡], appelé "échange cyclique par paires" ("*Cyclic Pairwise Exchange*", ou CPE). À partir d'un placement initial équilibré, il procède par échange de paires de processus entre sommets voisins au sens d'une dimension donnée, dimension après dimension.
L'intérêt de cette méthode est qu'elle est distribuée, et utilise efficacement le réseau de communication de la machine hypercubique sur laquelle il a été expérimenté. Chaque processeur choisit localement le processus lui appartenant et dont l'échange selon la dimension courante améliorerait le plus (ou pénaliserait le moins) la fonction de coût. Après rendez-vous avec le processeur voisin au sens de la dimension courante, et si la somme des gains de chaque déplacement est positive, c'est-à-dire si l'échange améliore effectivement la fonction de coût, celui-ci est réalisé.

La fonction de coût utilisée par Hammond, qu'il appelle Λ_1, est celle que nous utilisons : c'est la somme, pour toutes les arêtes du graphe source, du volume de communication transitant sur l'arête multiplié par la dilatation de celle-ci. Dans la première partie de sa thèse, Hammond montre la forte corrélation existant entre les valeurs de Λ_1 et le temps effectif pris par les programmes ainsi placés, validant ainsi le choix de cette fonction dans le cas du placement de programmes d'éléments finis sur la machine SIMD Connection Machine 2.
Il compare alors les qualités des placements obtenus par CPE à partir de placements initiaux calculés par différentes méthodes : placement aléatoire, placement "naïf" (c'est-à-dire selon l'ordre dans lequel les sommets du graphe sont numérotés), et placement issu d'une méthode de bipartitionnement spectral récursif (RSB). Les meilleurs résultats sont obtenus lorsque le placement initial est effectué par méthode RSB, ce qui n'est guère surprenant.
Hammond donne, dans les pages 58 à 63 de sa thèse, les valeurs du paramètre $\overline{\Lambda} \stackrel{\text{def}}{=} \frac{\Lambda_1}{4096}$ obtenues lors du placement sur l'hypercube de dimension 8, avec la méthode RSB+CPE, de plusieurs graphes de test. Ces résultats étant présentés sous forme graphique, nous ne disposons pas des valeurs exactes de $\overline{\Lambda}$, mais nous en donnons cependant une minoration.
Des définitions de Λ_1 et $\overline{\Lambda}$ données dans [44, pages 16 et 48], nous pouvons déduire une expression de $\overline{\Lambda}$ en fonction des paramètres que nous mesurons :

$$\overline{\Lambda} = \frac{2\,|E(S)|\,\mu_{exp}}{4096} \quad .$$

Les graphes de test de Hammond étant des graphes d'éléments finis et son placement effectuant un bipartitionnement strict, nous avons utilisé notre programme DRB avec la stratégie GP+FM+EX. Les résultats obtenus avec le programme RSB+CPE de Hammond et le nôtre sont

[‡] L'échange dimensionnel réalisé dans l'algorithme CPE est basé sur la définition de couplages parfaits disjoints dans l'hypercube. On pourrait envisager de l'appliquer à d'autres topologies, en y définissant les couplages adéquats.

Graphe	$\overline{\Lambda}$ RSB+CPE	$\overline{\Lambda}$ DRB{GP+FM+EX}
efBump	> 4	3.9467
ef4elt	> 5	4.5195
efEqu	> 50	44.3164
efRotor	> 90	81.4298

Table 12.2: Coût de communication des placements sur H(8) calculés par les programmes RSB+CPE et DRB{GP+FM+EX}

présentés dans le tableau 12.2.
Pour tous les graphes étudiés, les coûts de communication obtenus avec notre programme sont inférieurs d'environ 10 pourcents à ceux du programme RSB+CPE.
Ce premier test montre que, du point de vue de la qualité, notre stratégie de bipartitionnement GP+FM+EX donne des résultats équivalents à une méthode de bissection spectrale associée à la méthode d'optimisation globale CPE.

12.3.6.2 La bissection spectrale récursive multi-niveaux

Nous avons succintement décrit en page 65 le principe des méthodes de bissection spectrale récursive. Le handicap majeur de ces méthodes est leur lenteur, due au volume des calculs à effectuer pour obtenir le vecteur de Fiedler de la matrice associée au graphe.
Pour y remédier, Barnard et Simon ont proposé une nouvelle méthode de calcul du vecteur de Fiedler, de type "multi-niveaux" [4]. Elle consiste à contracter le graphe initial en une séquence de graphes de plus en plus petits, à calculer le vecteur de Fiedler sur le plus petit graphe de la séquence, puis à extrapoler de proche en proche les résultats obtenus pour tous les graphes de la séquence, y compris le graphe initial.
Les temps d'exécution de l'algorithme de Barnard et Simon (que nous noterons "ML-RSB", pour "*Multi-Level RSB*") sont meilleurs d'un ordre de grandeur que ceux des méthodes de bissection spectrale récursive classiques, pour une qualité équivalente.

Nous comparons dans cette section les performances de notre programme par rapport à celles du ML-RSB. Pour nous placer dans un cadre expérimental similaire, nous prenons le graphe complet comme topologie cible, et mesurons la coupe résultant du placement plutôt que son expansion (dans le cas de graphes homogènes placés sur des graphes complets, l'expansion moyenne est égale à la coupe divisée par le nombre d'arêtes du graphe).
Les temps donnés par Barnard et Simon sont mesurés sur une station Silicon Graphics dotée d'un processeur IP6 cadencé à 20 MHz et de 16 Mo de mémoire centrale. Pour ce test seulement, nous avons nous-même utilisé une station Silicon Graphics dotée d'un processeur IP6 cadencé à 12 MHz et de 16 Mo de mémoire centrale[§]. En multipliant les temps mesurés par $\frac{12}{20}$, nous obtenons des valeurs de temps estimées pouvant être comparées à celles de Barnard et Simon.

[§]Cette machine est environ quatre fois moins puissante que la machine Sun 4/90 que nous avons utilisée par ailleurs.

Les tests effectués par ces auteurs portent sur quatre graphes de test, dont seulement trois nous sont connus ; ce sont les graphes *efBump*, *ef4elt*, et *efPwt*. Nous donnons dans les tableaux 12.3, 12.4, et 12.5 les résultats obtenus pour ces graphes avec les programmes ML-RSB et DRB{GP+FM+EX}.

Parties	ML-RSB		DRB{GP+FM+EX}	
	Coupe	Temps	Coupe	Temps estimé
2	117	3.97	129	5.092
4	274	7.14	398	8.897
8	481	11.5	709	12.01
16	770	16.0	1255	14.83
32	1199	20.4	1916	17.53
64	1863	26.2	2820	19.79
128	2798	31.5	4082	22.14

Table 12.3: Taille totale de la coupe et temps d'exécution du placement sur K(x) du graphe *efBump*, calculé par les programmes ML-RSB et DRB{GP+FM+EX}. N.B. : ces temps sont relatifs à une machine de type Silicon Graphics.

Parties	ML-RSB		DRB{GP+FM+EX}	
	Coupe	Temps	Coupe	Temps estimé
2	164	11.3	143	26.92
4	492	21.2	391	48.46
8	808	30.7	734	65.74
16	1375	41.2	1097	83.80
32	2131	53.4	1811	98.65
64	3252	68.8	2950	111.7
128	4925	88.0	4633	122.3

Table 12.4: Taille totale de la coupe et temps d'exécution du placement sur K(x) du graphe *ef4elt*, calculé par les programmes ML-RSB et DRB{GP+FM+EX}. N.B. : ces temps sont relatifs à une machine de type Silicon Graphics.

Parties	ML-RSB		DRB{GP+FM+EX}	
	Coupe	Temps	Coupe	Temps estimé
2	376	28.0	369	18.55
4	942	48.1	743	31.92
8	1684	72.4	1648	57.86
16	3302	106	3168	76.87
32	6273	162	6100	107.9
64	9790	203	9324	131.2
128	13535	249	13021	151.2

Table 12.5: Taille totale de la coupe et temps d'exécution du placement sur K(x) du graphe *efPwt*, calculé par les programmes ML-RSB et DRB{GP+FM+EX}. N.B. : ces temps sont relatifs à une machine de type Silicon Graphics.

Pour le graphe *efBump*, les résultats que nous obtenons avec notre programme sont moins bons

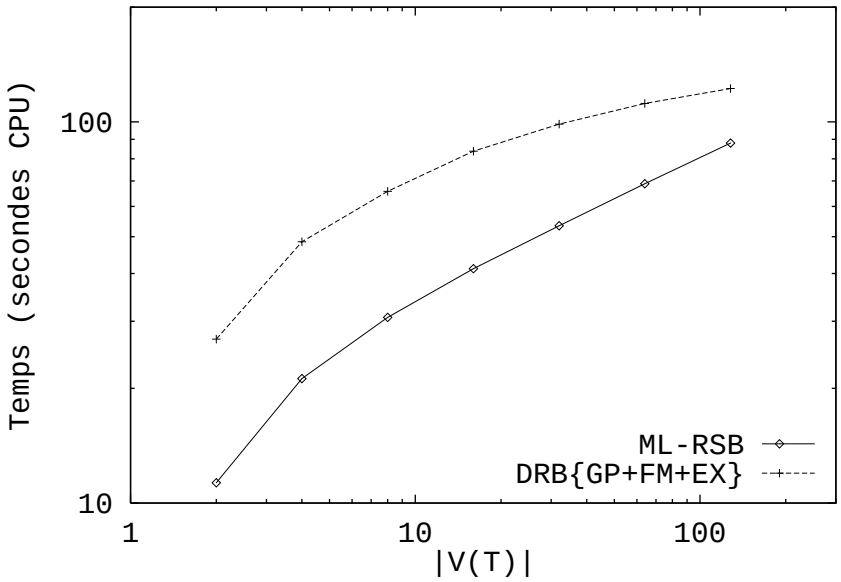


Figure 12.22: Temps d'exécution du placement sur K(x) du graphe *ef4elt*, calculé par les programmes ML-RSB et DRB{GP+FM+EX}.

que ceux fournis par le programme ML-RSB. Ceci s'explique par le comportement de l'algorithme de Gibbs, Poole, et Stockmeyer vis-à-vis de ce graphe, du fait de la forme rectangulaire de celui-ci. Lors du premier bipartitionnement, le nœud pseudo-périphérique choisi est l'un des coins du rectangle, car la distance entre ce sommet et celui qui lui est diagonalement opposé est l'une des plus grandes possibles. Partant de ce nœud, les couches sont donc orientées diagonalement par rapport aux côtés du rectangle. En dépit de l'optimisation effectuée ensuite par l'heuristique de Fiduccia et Mattheyses, la coupe conserve en partie cette orientation, et est donc plus grande que celle que l'on obtiendrait en coupant parallèlement au petit côté du rectangle. Ce phénomène, en se reproduisant sur les sous-graphes trapézoïdaux ainsi obtenus, et en s'amplifiant à chaque bipartitionnement, pénalise notre programme par rapport à la méthode spectrale. La grande régularité et la forme simple du maillage permettent en effet à celle-ci un calcul précis des ventres de vibration, c'est-à-dire des coupes, qui sont orientées parallèlement aux petits côtés des rectangles obtenus par bipartitionnements successifs.
Le comportement de notre programme vis-à-vis du graphe *efBump* est détaillé dans la section A.1 de l'annexe A, page 126.

Pour les graphes *ef4elt* et *efPwt*, nous obtenons, suivant le nombre de parties demandé, des coupes de tailles inférieures de 5 à 20 pourcents à celles calculées par le programme ML-RSB.

Dans deux cas sur trois, les temps d'exécution de notre programme sont inférieurs à ceux mis par le programme ML-RSB. Remarquons à ce propos que l'évolution de ces temps en fonction du nombre de parties n'est pas identique pour les deux programmes.

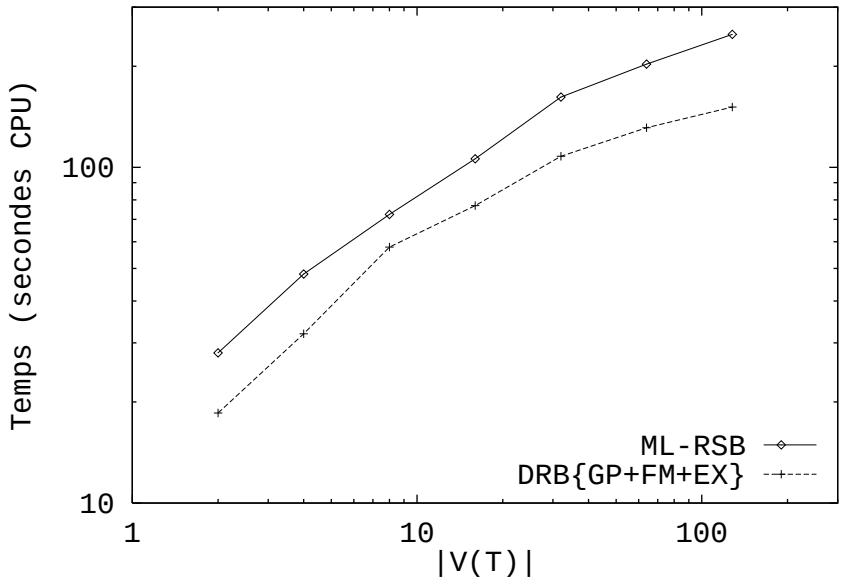


Figure 12.23: Temps d'exécution du placement sur K(x) du graphe *efPwt*, calculé par les programmes ML-RSB et DRB{GP+FM+EX}.

Ainsi, pour le graphe *ef4elt*, si le temps estimé de notre programme est 2.5 fois plus important que celui obtenu par Barnard et Simon pour 2 parties, il n'est plus supérieur que de 40 pourcents pour 128 parties. Comme le montre la figure 12.22, le comportement logarithmique de notre programme par rapport à la taille du graphe cible joue ici en notre faveur, alors que le programme ML-RSB semble rapidement atteindre un régime linéaire.
Pour le graphe *efPwt*, le temps mis par le programme ML-RSB est supérieur de 51 pourcents au nôtre pour 2 parties, et de 65 pourcents pour 128 parties. L'augmentation de l'écart entre les temps des deux programmes est visible en figure 12.23, où le comportement logarithmique de notre programme apparaît nettement. Ici encore, l'ordre de grandeur du comportement expérimental en temps du programme ML-RSB semble supérieur au nôtre.

Ces résultats, qui demandent à être confirmés sur d'autres exemples, sont très encourageants. Ils montrent que notre programme DRB{GP+FM+EX} offre une qualité de placement équivalente, voire supérieure, à celle d'un algorithme de bissection spectrale récursive multi-niveaux, en des temps aussi petits, et qui semblent asymptotiquement meilleurs.

Chapitre 13

Conclusion et perspectives

13.1 Conclusion

Nous avons présenté dans cette partie la structure d'un programme de placement par bipartitionnement récursif conjoint des graphes de processus et d'architecture. Le traitement des méthodes de bipartitionnement de domaines et de graphes comme des boîtes noires permet d'utiliser n'importe quelle méthode de bipartitionnement, mais aussi de placer sur tout type d'architecture cible.

La première heuristique de bipartitionnement que nous avons implantée est celle de Fiduccia et Mattheyses, que nous avons améliorée afin de pouvoir traiter les graphes valués. Nous montrons que les modifications que nous apportons ne modifient pas la qualité des solutions obtenues, et conservent la linéarité en temps qui fait la force de cet algorithme.
Le calcul de bipartitions initiales judicieuses au moyen d'un algorithme de type Gibbs, Poole, et Stockmeyer modifié nous a permis d'augmenter la qualité des placements obtenus (pour les graphes de dissections emboîtées), ainsi que le temps de calcul de l'heuristique de Fiduccia et Mattheyses (pour les graphes de maillages, pour lesquels le gain peut atteindre 30 pourcents).

La linéarité du comportement en temps de toutes les méthodes de bipartitionnement de graphes que nous utilisons fait que le comportement en temps de notre programme de placement évolue comme le produit du nombre d'arêtes du graphe source et du logarithme du nombre de sommets du graphe cible. À nombre de processeurs fixé, il a donc un comportement linéaire par rapport à la taille du problème, ce qui le classe parmi les programmes rapides.

Le placement sur un graphe cible donné requiert le calcul préalable d'une décomposition récursive de celui-ci. Pour que les placements obtenus soient efficaces, cette décomposition doit respecter certains critères, tels la connexité des sous-domaines et la minimisation de leur coupe.
Nous proposons une méthode automatique de calcul des décompositions du graphe cible, basée

sur le placement du graphe cible sur le graphe complet de même ordre. Pour la valider, nous comparons les résultats de placements obtenus sur des hypercubes, graphes de De Bruijn, et grilles bidimensionnelles décomposés soit par notre méthode, soit de manière classique (pour les graphes de De Bruijn, la décomposition de référence est une décomposition algorithmique en sous-graphes partiels).
Si l'on obtient pour les deux premières familles des résultats comparables, voire supérieurs, à ceux calculés avec les décompositions algorithmiques, la qualité des placements effectués sur les grilles bidimensionnelles décomposées par placement n'est pas satisfaisante. Ceci est dû au respect de l'exactitude stricte des cardinaux des sous-domaines lors du calcul de nos décompositions, qui se fait au détriment de la minimisation de la coupe. Ce phénomène est amplifié par la 4-connexité de la grille, qui peut provoquer la déconnexion des sous-domaines calculés, et facilite le piégeage de notre heuristique dans des minima locaux de la fonction de coût de communication que nous utilisons.

Nous avons comparé les placements que nous obtenons avec ceux issus d'autres programmes. Les résultats que nous obtenons sont qualitativement comparables à ceux issus des meilleurs algorithmes de bissection spectrale, en un temps équivalent à ceux des méthodes les plus rapides. Le rapport entre la qualité de nos résultats et le temps de calcul est donc très intéressant.

13.2 Perspectives

La décomposition par placement d'architectures cibles nous semble une méthode intéressante, en dépit des mauvais résultats obtenus sur la grille 4-connexe.
Nous comptons modifier notre algorithme de décomposition par placement pour de tester des décompositions d'architectures ne respectant pas la stricte égalité des cardinaux, afin de privilégier la minimisation de la coupe par rapport à l'équilibrage. Nous comptons également étudier les propriétés de la grille 8-connexe vis-à-vis de la décomposition d'architecture.

Les résultats d'expérimentation étant prometteurs, nous souhaitons diffuser l'outil général que constitue notre placeur le plus largement possible. En particulier, nous comptons intégrer notre programme à l'environnement parallèle ADAM [2], et le mettre à la disposition de la communauté scientifique.

Lorsque le programme parallèle a un comportement irrégulier et non prédictible dans le temps, le placement statique ne permet pas d'assurer l'équilibrage de charge pendant toute la durée d'exécution du programme. Il faut alors faire appel à un régulateur dynamique de charge, qui assurera l'équilibre par migration de processus entre les processeurs, ou par migration de données entre processus dans le cadre SPMD.
La qualité des résultats fournis, et surtout la rapidité de leur calcul, font de notre programme un

outil intéressant pour le calcul de bipartitionnements initiaux utilisables comme bonne condition initiale par un tel régulateur dynamique de charge.

Nous avons vu que, par placement du graphe source sur le graphe complet, notre placeur peut servir à calculer des découpages du graphe utilisés en algorithmique numérique par les méthodes dites de sous-domaines.
Notre programme peut également être appliqué au calcul de séparateurs utilisés dans l'algorithme des dissections emboîtées. Il suffit pour cela que chaque tâche de bipartitionnement construise un séparateur sommet à partir du séparateur arête fourni par les heuristiques de bipartitionnement, et exclue les sommets de celui-ci des sous-graphes obtenus.

Bien que notre programme soit actuellement séquentiel, son approche "diviser pour résoudre" et la modularité de sa conception le rendent facilement parallélisable. En particulier, le séquencement par niveaux a plusieurs conséquences intéressantes.
Si, entre deux niveaux, une phase de synchronisation est nécessaire aux tâches de bipartitionnement pour effectuer un *échange total* de leurs résultats, afin de mettre à jour les informations de distance, toutes les tâches d'un même niveau peuvent cependant s'exécuter en parallèle. Qui plus est, les premières (et plus lourdes) tâches de bipartitionnement peuvent s'exécuter en parallèle sur plusieurs processeurs de la machine supportant le placeur, le choix d'utiliser la version parallèle ou séquentielle de la tâche étant le fait de la stratégie de placement. Le problème est alors de disposer d'algorithmes de bipartitionnement parallèles efficaces, ce qui est encore au stade de la recherche.
Remarquons que, dans le cas où la machine parallèle qui exécute le placement parallèle est la machine cible, chaque processeur connaît au dernier niveau la liste des processus qui y sont placés. Les processeurs peuvent alors directement demander le chargement et l'exécution des processus. On dispose alors d'un placeur-chargeur automatique en ligne, qui peut être intégré à l'environnement de programmation de la machine parallèle.

Annexe A

Analyse des résultats de quelques placements

Pour illustrer le fonctionnement de notre programme, nous présentons dans cette annexe les résultats du placement des graphes bidimensionnels *efBump* et *ef3elt*, dessinés en figures 12.1 et 12.2 page 99, sur les premiers représentants des familles de graphes complets et d'hypercubes.
La topologie de graphe complet a été choisie afin d'étudier le comportement des algorithmes de bipartitionnement indépendamment de toute influence extérieure, puisque le coût des arêtes appartenant aux cocycles des sous-graphes à bipartitionner n'est alors pas pris en compte. Elle permet également d'étudier le comportement de notre programme par rapport aux méthodes de bissection spectrale. Nous avons choisi l'hypercube comme deuxième topologie car c'est, parmi toutes celles dont nous disposons, celle qui se rapproche le plus des graphes complets pour les petits ordres ; les influences de la topologie sur le placement sont donc visibles par comparaison, mais ne conduisent pas à de trop grandes différences entre les placements obtenus. Un avantage supplémentaire de cette topologie est que ses premiers représentants (carré et cube), sont simples et bien connus, ce qui facilite notre interprétation.
Chaque graphe source est successivement placé sur les graphes K(2), K(4), K(8), ainsi que sur H(2) et H(3) ; en effet, comme K(2) est isomorphe à H(1), le résultat du placement sur le deuxième est identique à celui obtenu sur le premier, et n'est donc pas représenté. Tous ces placements sont effectués en utilisant la stratégie GP+FM+EX.

Il est à noter que, du fait de la méthode de décomposition choisie pour l'architecture hypercubique, le quotientement de l'hypercube H(k) par la partition constituée de tous les sous-domaines obtenus au niveau i de l'arbre de bipartitionnement (avec $0 \leq i \leq k$) donne le graphe H(i). Le placement partiel d'un graphe source réalisé au niveau i de l'arbre de bipartitionnement est donc équivalent au placement de ce graphe sur H(i), pour i compris entre 0 et k. De fait, le placement d'un graphe sur H(k) peut être étudié niveau par niveau en plaçant successivement ce graphe sur les graphes H(i), pour i allant de 0 à k. Ceci est également vrai pour la famille des graphes complets.

Nous représentons le résultat d'un placement en plaquant sur chaque sommet des graphes sources un disque dont la couleur dépend du processeur sur lequel le sommet est placé ; deux sommets placés sur le même processeur sont donc coloriés de la même teinte. Le diamètre des disques est calculé pour s'adapter aux irrégularités de taille des mailles et donner un rendu visuel homogène ; il n'a aucune autre signification.

A.1 Placement du graphe *efBump*

Les figures A.1, A.2, et A.3 montrent les bipartitionnements successifs effectués lorsque seule la minimisation de la coupe est prise en compte à chaque étape.

Comme l'heuristique de Fiduccia et Mattheyses cherche à minimiser localement la coupe, les coupes entre sous-graphes générées par les bipartitionnements ont tendance à suivre les axes principaux du maillage régulier hexagonal sous-jacent. Ceci se voit nettement dans la partie gauche de la figure A.1 et, surtout, dans la figure A.2.

L'influence de l'algorithme de Gibbs, Poole, et Stockmeyer sur le tracé des coupes entre sous-graphes apparaît nettement lors du bipartitionnement du sous-graphe inférieur de la figure A.1, dont le résultat est visible en figure A.2. Le nœud pseudo-périphérique trouvé par l'algorithme est le coin supérieur gauche du sous-graphe, situé à distance diamètre du coin inférieur droit, et donc les couches construites à partir de ce nœud sont orientées diagonalement par rapport aux côtés du rectangle dans lequel s'inscrit le sous-graphe. L'optimisation locale effectuée ensuite par l'heuristique de Fiduccia et Mattheyses ne peut empêcher l'apparition de décrochements successifs dans le tracé de la coupe, qui augmentent d'autant la taille de celle-ci par rapport à un tracé horizontal plus direct. On retrouve également ce phénomène lors du bipartitionnement du sous-graphe supérieur de la figure A.1, visible en figure A.2, ainsi qu'en figure A.3.
Ceci explique les moins bons résultats obtenus avec notre programme sur ce graphe de test par rapport aux méthodes spectrales (voir tableau 12.3).

La figure A.4 diffère peu de la figure A.2. En effet, comme le placement sur H(1) est équivalent au placement sur K(2), les bipartitions initiales calculées par l'algorithme de Gibbs, Poole, et Stockmeyer au deuxième niveau du placement sur H(2) sont identiques à celles obtenues sur K(4). Puisque le graphe quotient obtenu à ce stade est la chaîne P(4), et que celle-ci est un sous-graphe partiel de H(2), la minimisation de la fonction de coût sur l'hypercube par l'heuristique de Fiduccia et Mattheyses est équivalente dans ce cas précis à la minimisation de la fonction de coût sur le graphe complet, et donne donc le même résultat.
La seule différence notable entre les deux figures est que les processeurs sur lesquels sont placés les deux sous-graphes situés dans leur partie haute sont intervertis. Cet échange de processeurs tend à minimiser la communication sur l'hypercube, car il permet de n'avoir que des arêtes de dilatation 1 ; il est équivalent à une numérotation de Gray des sommets du graphe quotient P(4), qui assure son plongement sur H(2) avec dilatation 1. Remarquons que cet échange d'affectation

des sous-graphes, en modifiant les listes de gains, a légèrement perturbé la coupe par rapport à celle de la figure A.2.

L'influence de la topologie cible sur le placement apparaît clairement dans la figure A.5. L'ordre dans lequel les sous-graphes de la figure A.4 sont bipartitionnés est, si on les considère de haut en bas, 1, 2, 4, et 3. Considérons ces sous-graphes dans cet ordre.

- Le bipartitionnement du premier sous-graphe, situé en haut de la figure, ne dépend pas des informations de placement extérieures, car seul un unique sous-graphe lui est voisin.

- En revanche, le bipartitionnement du deuxième sous-graphe utilise l'information issue du bipartitionnement précédent : afin de réduire le nombre d'arêtes de dilatation 2, la coupe générée "prolonge" celle du premier sous-graphe, ce qui n'était pas le cas pour le placement sur le graphe complet.

- Le troisième sous-graphe à être bipartitionné est celui situé au bas de la figure. Le sous-graphe situé au dessus de lui n'ayant pas encore été bipartitionné, on se trouve dans une situation analogue à celle du premier bipartitionnement évoqué ci-dessus. Comme la coupe réalisée au niveau précédent est similaire à celle de la figure A.2, la coupe obtenue est donc identique à celle de la figure A.3.

- Le quatrième sous-graphe à bipartitionner, qui est le deuxième à partir du bas de la figure, utilise l'information de placement des bipartitionnements réalisés tant sur le graphe immédiatement supérieur que sur celui situé juste en dessous de lui. Dans l'idéal, on souhaiterait obtenir un graphe quotient isomorphe à $M_2(4,2)$, qui se plonge avec dilatation 1 dans H(3). Malheureusement, lors du bipartitionnement précédent, comme aucune information de placement n'était utilisable, les deux sous-graphes du bas de la figure A.5 ont été attribués aux deux processeurs du sous-domaine correspondant de manière arbitraire et, dans ce cas précis, contrairement à nos espérances. Afin de minimiser la communication, et donc en premier lieu le nombre d'arêtes de dilatation 2, les deux sous-graphes générés par la coupe ont une forme "croisée", pour avoir le maximum de leurs voisins à distance 1. On remarque à ce propos que l'un des sous-domaines générés n'est pas connexe ; le petit sous-graphe déconnecté sert de "tampon" entre les deux sous-graphes qui l'entourent, et transforme par sa présence un certain nombre d'arêtes de dilatation 2 en un nombre d'arêtes de dilatation 1 inférieur au double du précédent.

Cette étude nous suggère un moyen d'optimiser notre programme. Il consiste à effectuer les bipartitionnements d'un niveau en sélectionnant à chaque fois la tâche dont le sous-graphe possède le plus de voisins dans des sous-graphes déjà traités, afin de propager et d'utiliser au mieux les résultats des bipartitionnements effectués au même niveau.

A.2 Placement du graphe *ef3elt*

Les figures A.6, A.7, et A.8 montrent les bipartitionnements successifs calculés lorsque seule la minimisation de la coupe est prise en compte à chaque étape.
On peut remarquer en figure A.6 la convexité de la coupe entre les deux sous-graphes, qui favorise sa minimisation puisque dans l'espace euclidien le disque maximise la surface par rapport au périmètre, c'est-à-dire le nombre d'arêtes contenues dans le sous-graphe convexe par rapport au nombre d'arêtes de la coupe.
Notons également, en figure A.7, la manière dont l'heuristique de Fiduccia et Mattheyses tire parti de la structure du graphe pour minimiser la coupe : les volets sont utilisés pour séparer les deux sous-graphes issus du sous-graphe convexe.
L'heuristique de Fiduccia et Mattheyses peut conduire à des sous-graphes non connexes lorsque cela permet de minimiser la coupe. Ainsi, en figure A.8, l'une des deux parties issues du sous-graphe prenant appui sous les volets est-elle non-connexe.

La différence majeure entre la figure A.9 et la figure A.7 réside dans la présence de sommets "tampons" entre le sous-graphe situé sous les volets et le sous-graphe englobant. Ces sommets, qui appartiennent au second sous-graphe issu du bipartitionnement du sous-graphe convexe de la figure A.6, reflètent l'influence de la topologie cible sur le placement. Ils permettent en effet, de manière analogue à ce que l'on a déjà observé en figure A.5, de réduire le nombre d'arêtes de dilatation 2 au profit d'arêtes de dilatation 1.
Ce phénomène d'"encapsulation" apparaît de façon encore plus nette en figure A.10 ; à l'avant du bord d'attaque tout d'abord, où la partie inférieure issue du bipartitionnement du sous-graphe englobant de la figure A.9 remonte vers la coupe entre les sous-graphes du bord d'attaque ; sous les volets également, où un sous-graphe déconnecté du sous-graphe englobant supérieur sert de tampon et transforme des arêtes de dilatation 3 en arêtes de dilatation 1 et 2. La différence de structure la plus notable entre cette figure et la figure A.8 reste cependant la manière dont est bipartitionné le sous-graphe situé au dessus des volets. Pour le placement sur le graphe complet, la coupe obtenue est de forme convexe afin d'être minimale, alors que pour le placement sur l'hypercube elle traverse le sous-graphe de part en part. Cette dernière configuration permet en effet de n'avoir sur le cocycle de ce sous-graphe pratiquement que des arêtes de dilatation 1, ce qui ne serait pas le cas si la coupe de la figure A.8 était utilisée. Remarquons également que, à l'image de ce qui se produit pour la figure A.5, les coupes entre sous-graphes tendent à se rejoindre et se prolonger mutuellement, toujours dans le but de réduire le nombre d'arêtes de grande dilatation.

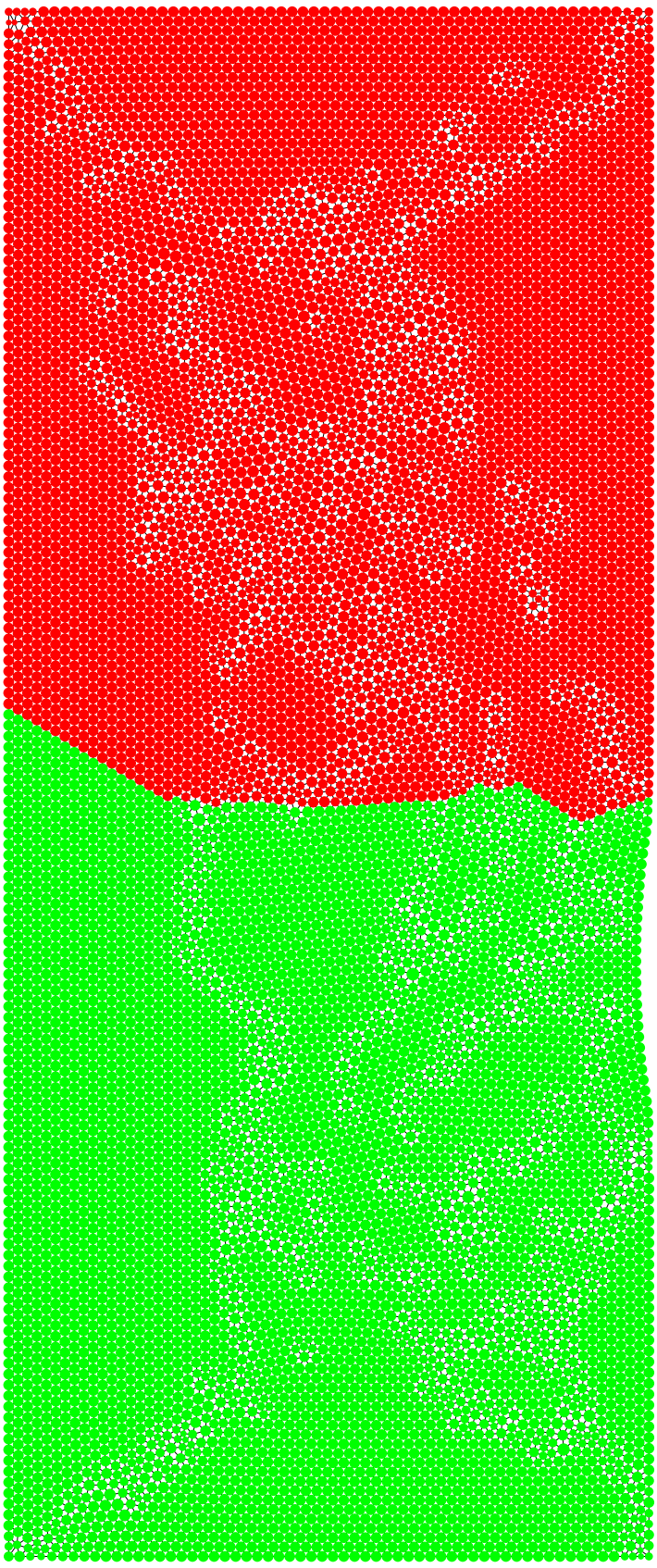


Figure A.1: Résultat du placement du graphe *efBump* sur K(2) avec la stratégie GP+FM+EX.

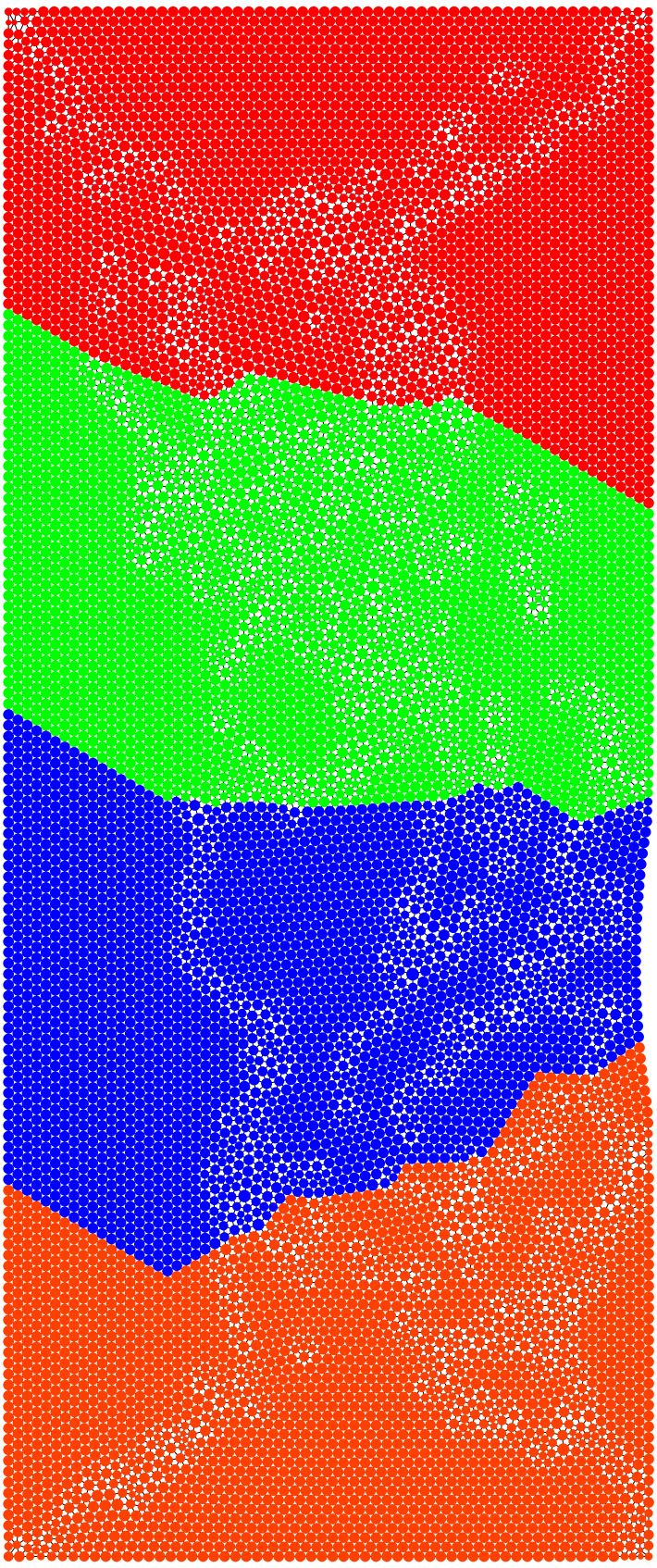


Figure A.2: Résultat du placement du graphe *efBump* sur K(4) avec la stratégie GP+FM+EX.

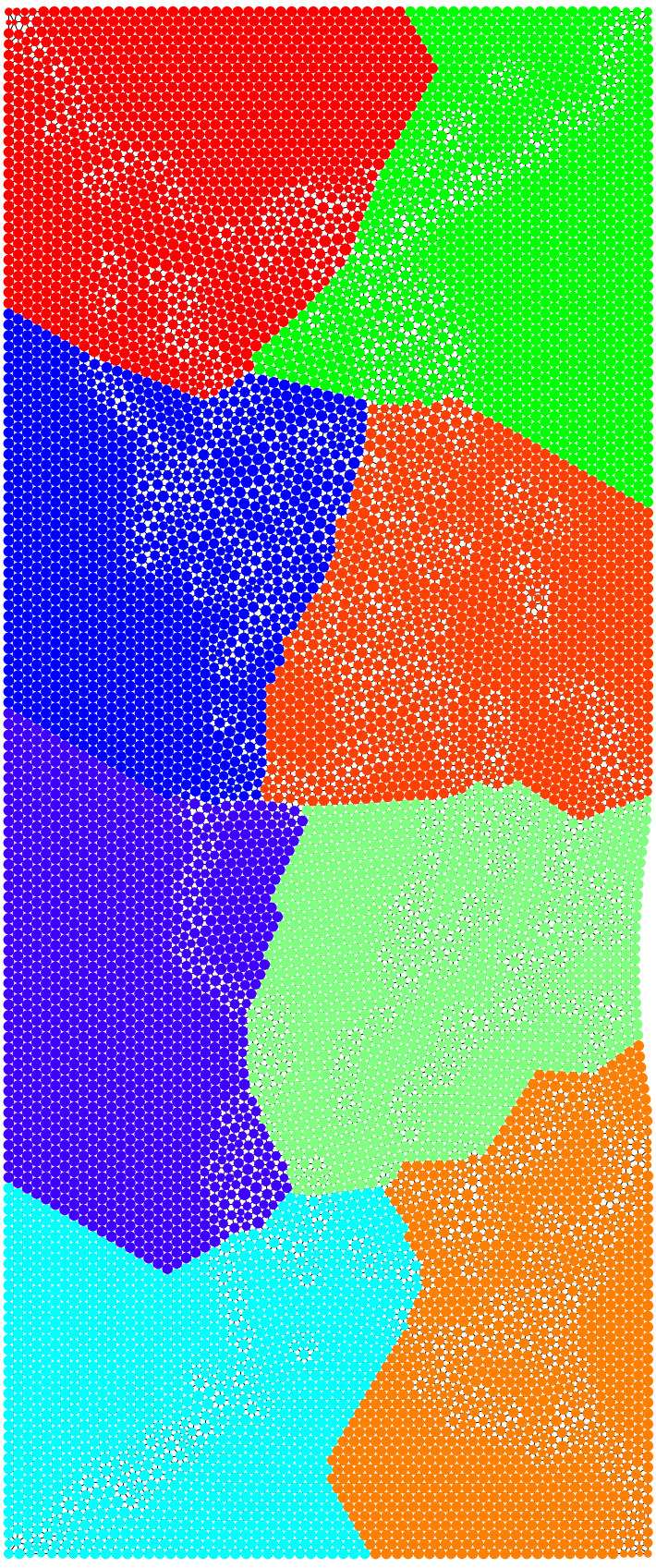


Figure A.3: Résultat du placement du graphe *efBump* sur K(8) avec la stratégie GP+FM+EX.

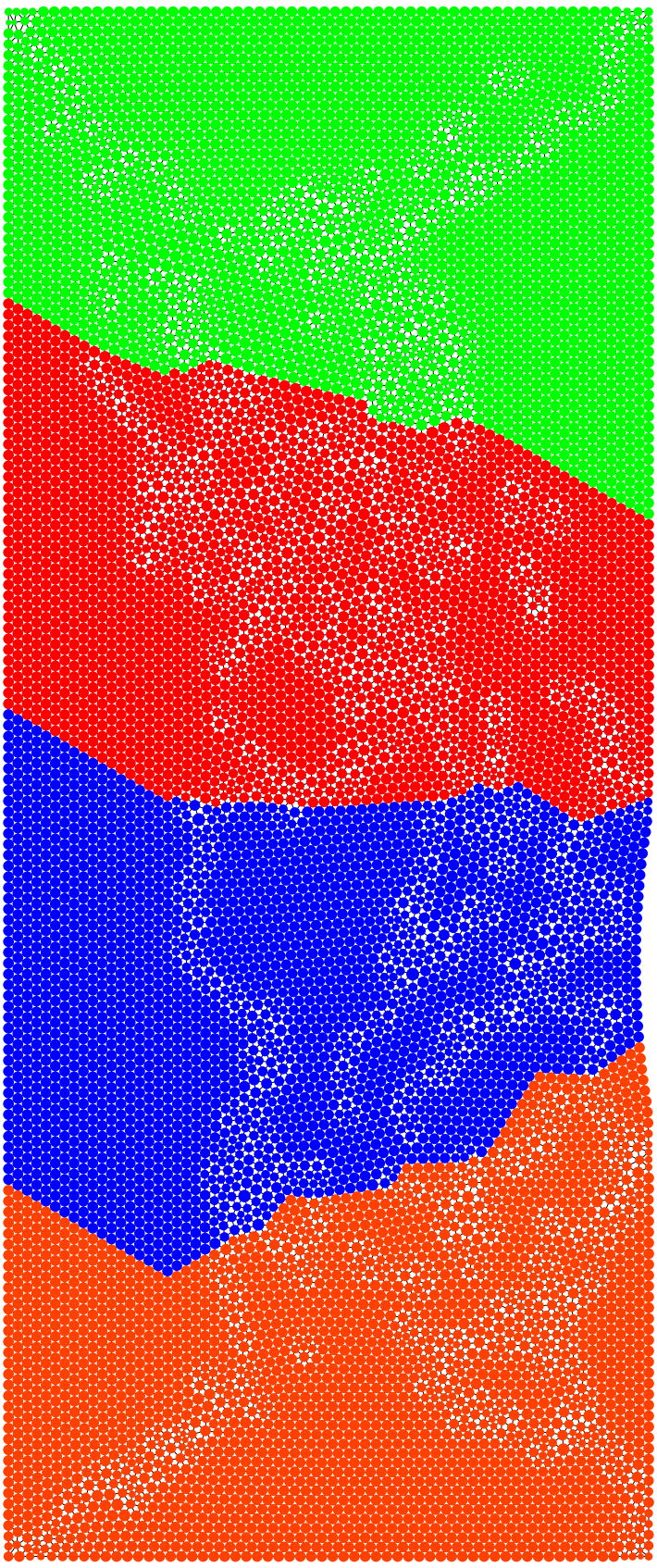


Figure A.4: Résultat du placement du graphe *efBump* sur H(2) avec la stratégie GP+FM+EX.

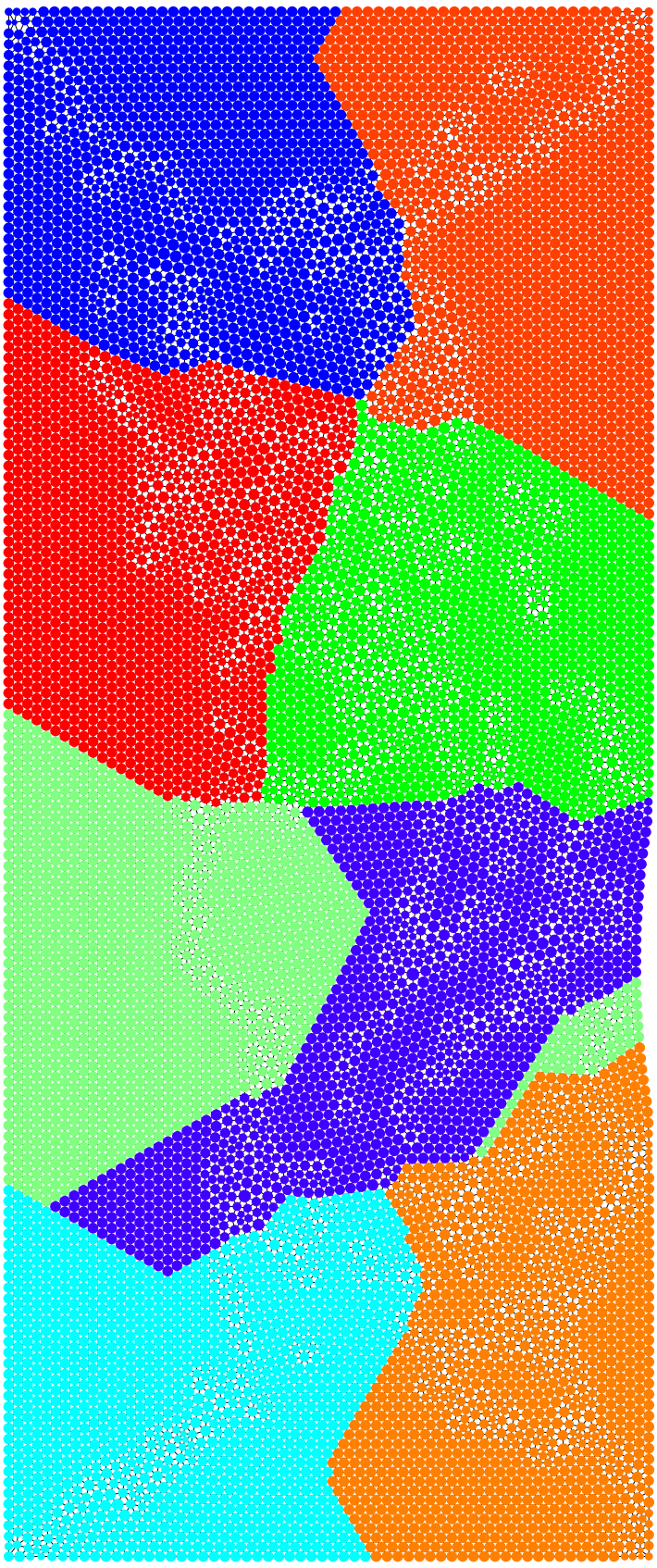


Figure A.5: Résultat du placement du graphe *efBump* sur H(3) avec la stratégie GP+FM+EX.

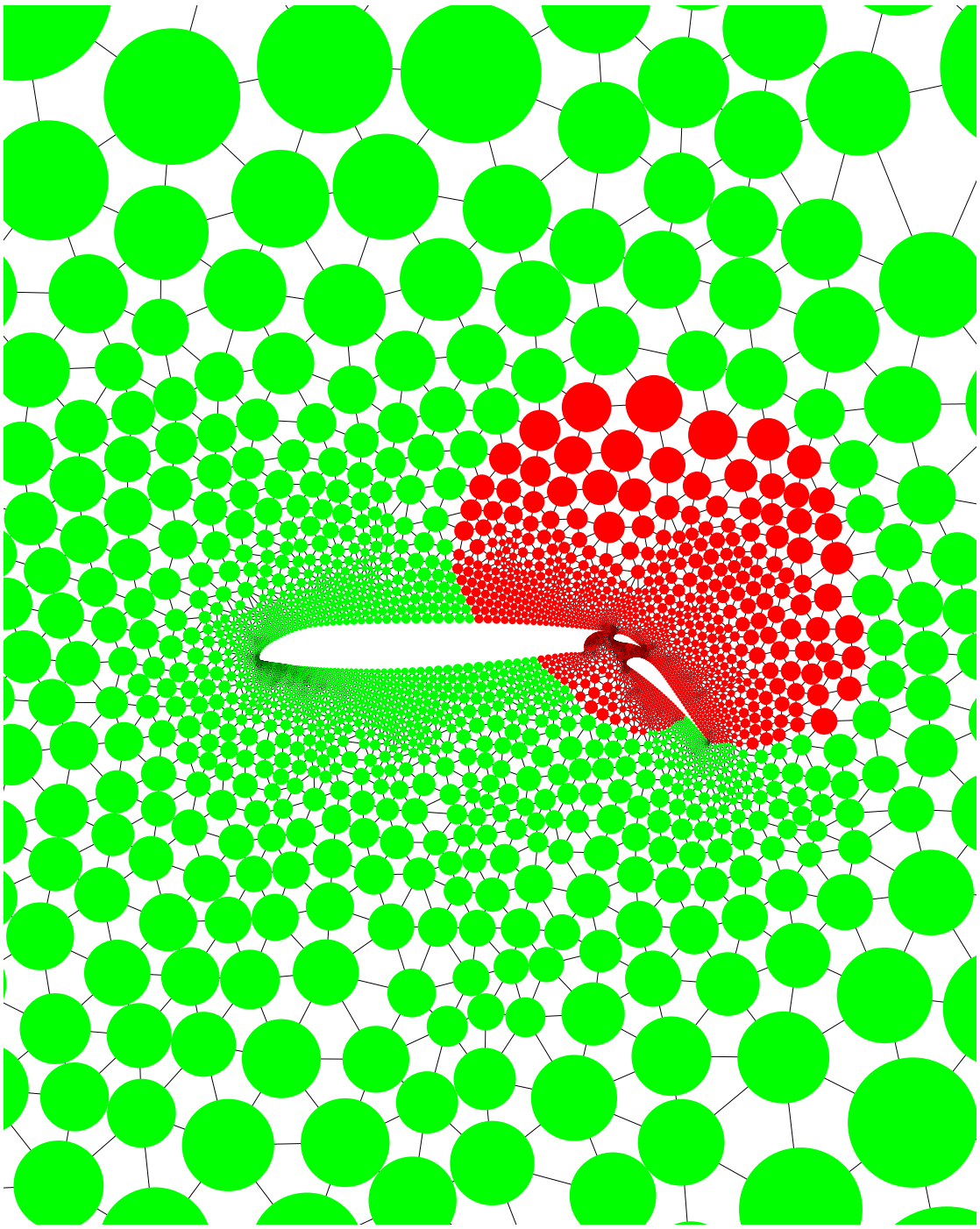


Figure A.6: Résultat du placement du graphe *ef3elt* sur K(2) avec la stratégie GP+FM+EX.

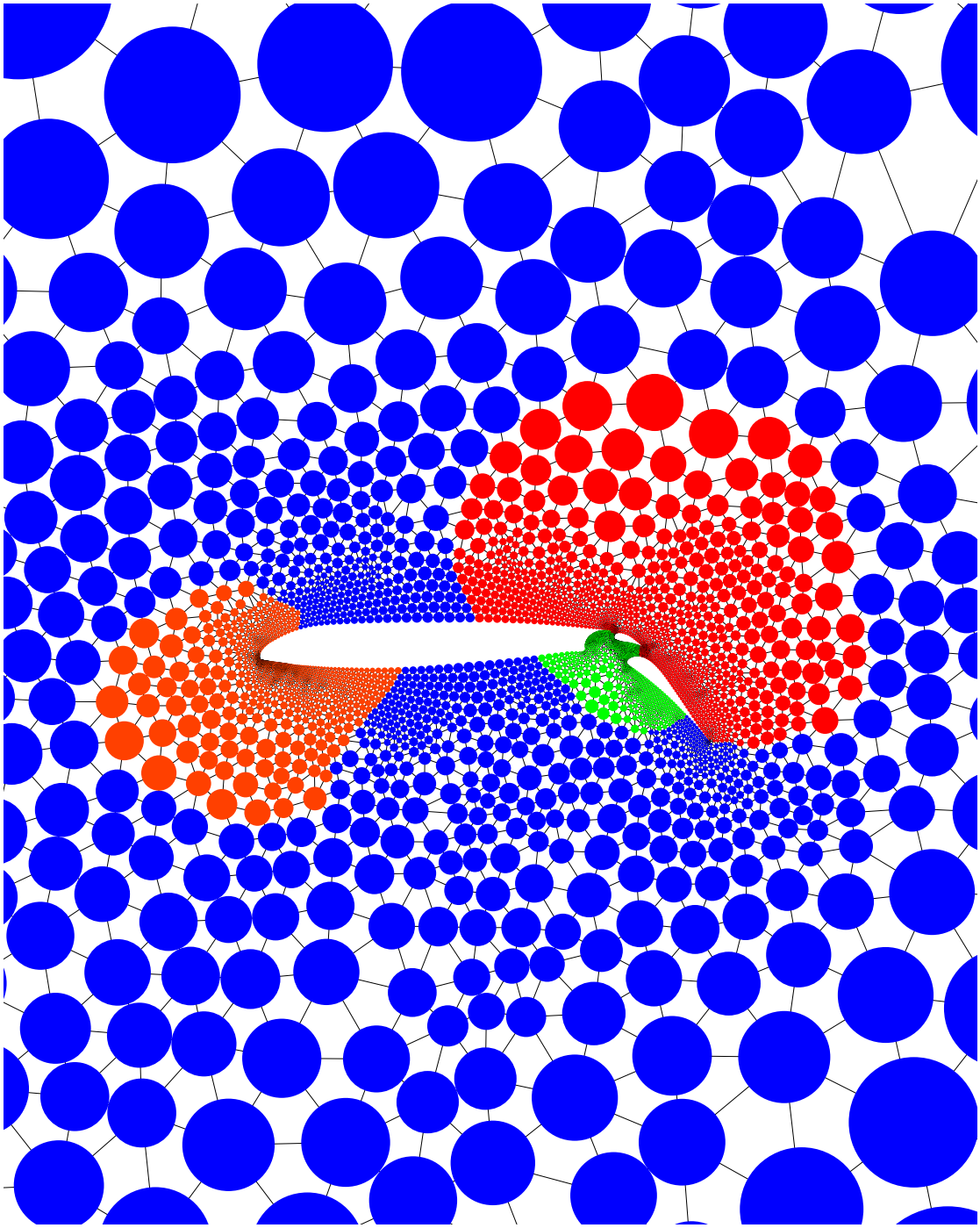


Figure A.7: Résultat du placement du graphe *ef3elt* sur K(4) avec la stratégie GP+FM+EX.

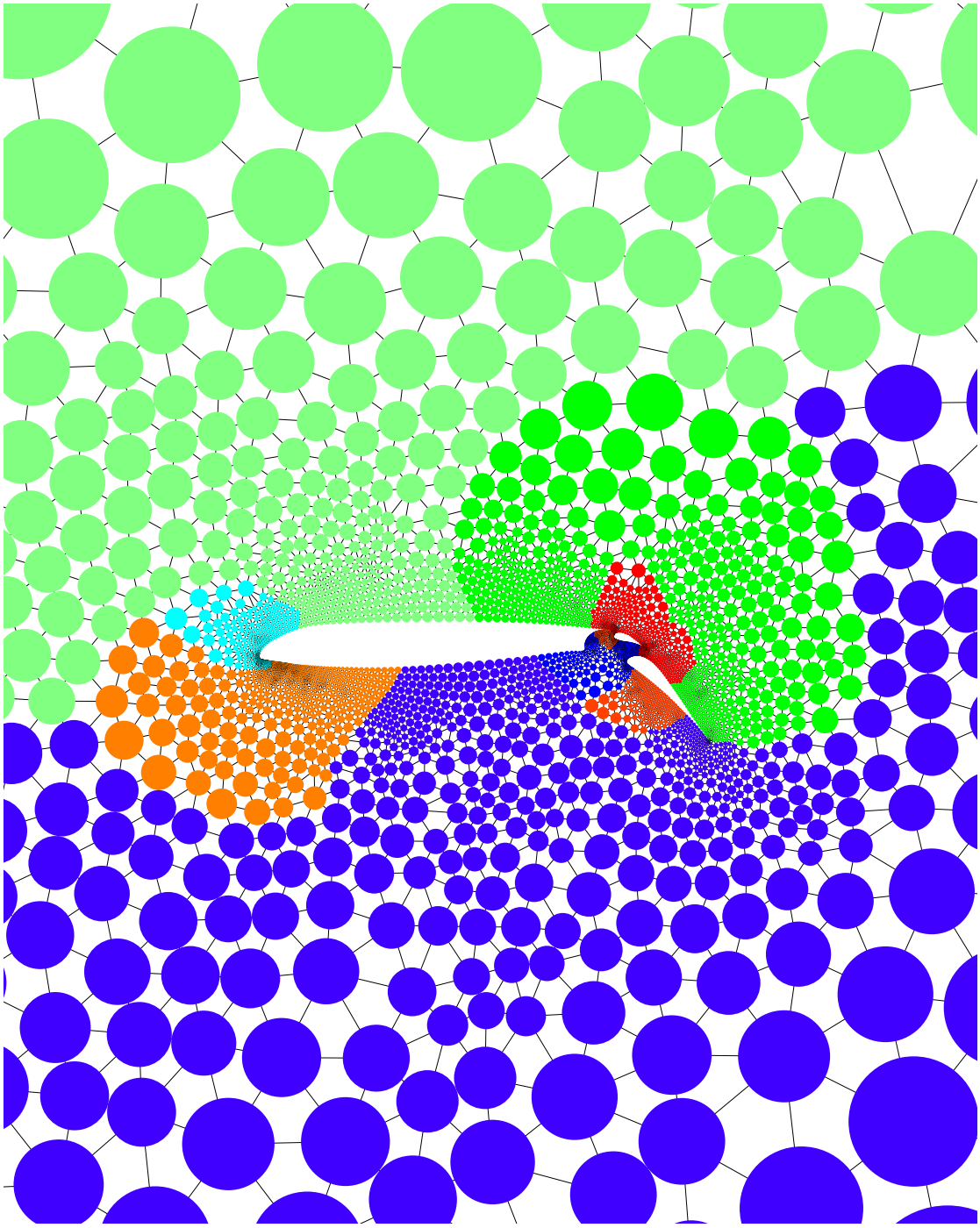


Figure A.8: Résultat du placement du graphe *ef3elt* sur K(8) avec la stratégie GP+FM+EX.

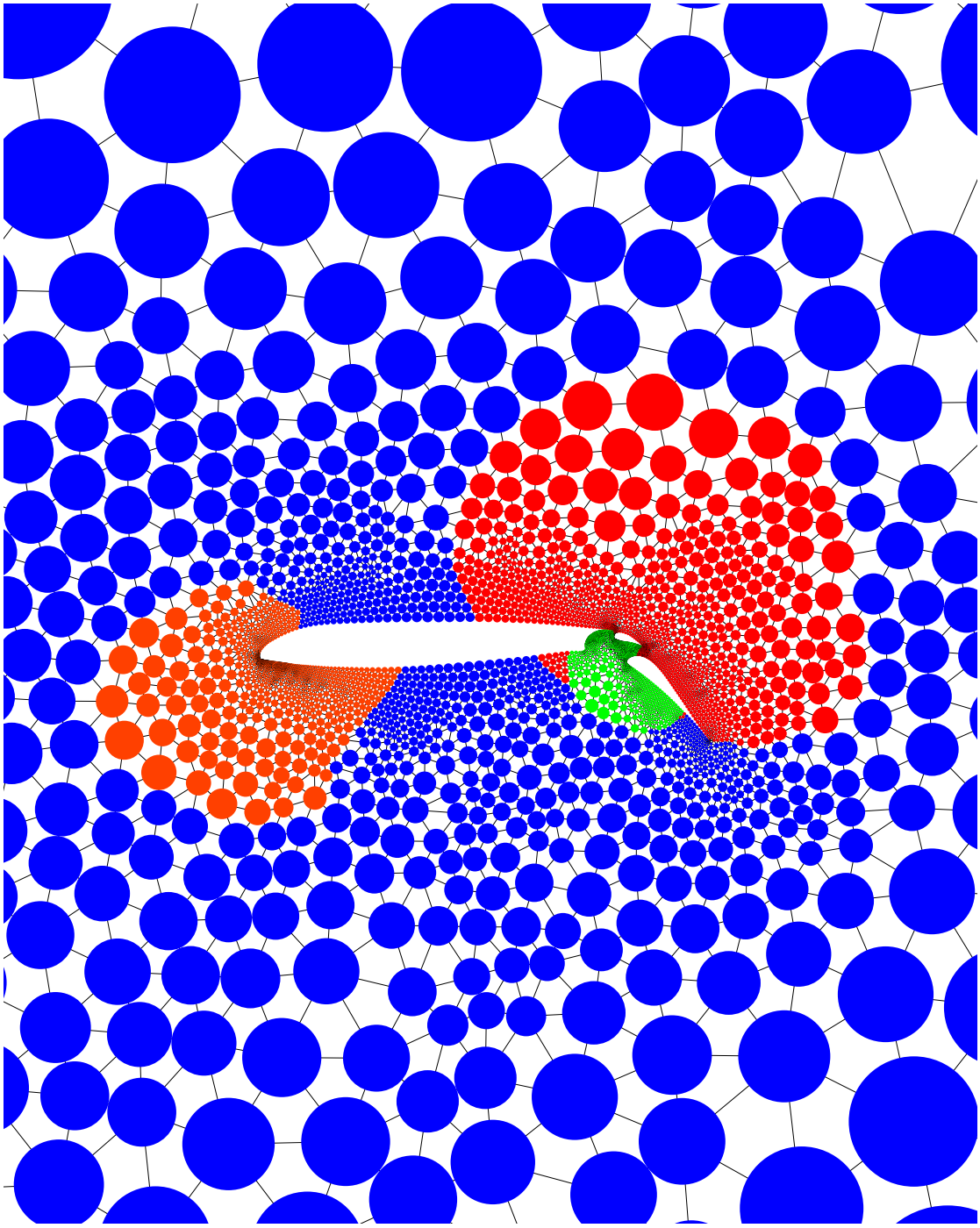


Figure A.9: Résultat du placement du graphe *ef3elt* sur H(2) avec la stratégie GP+FM+EX.

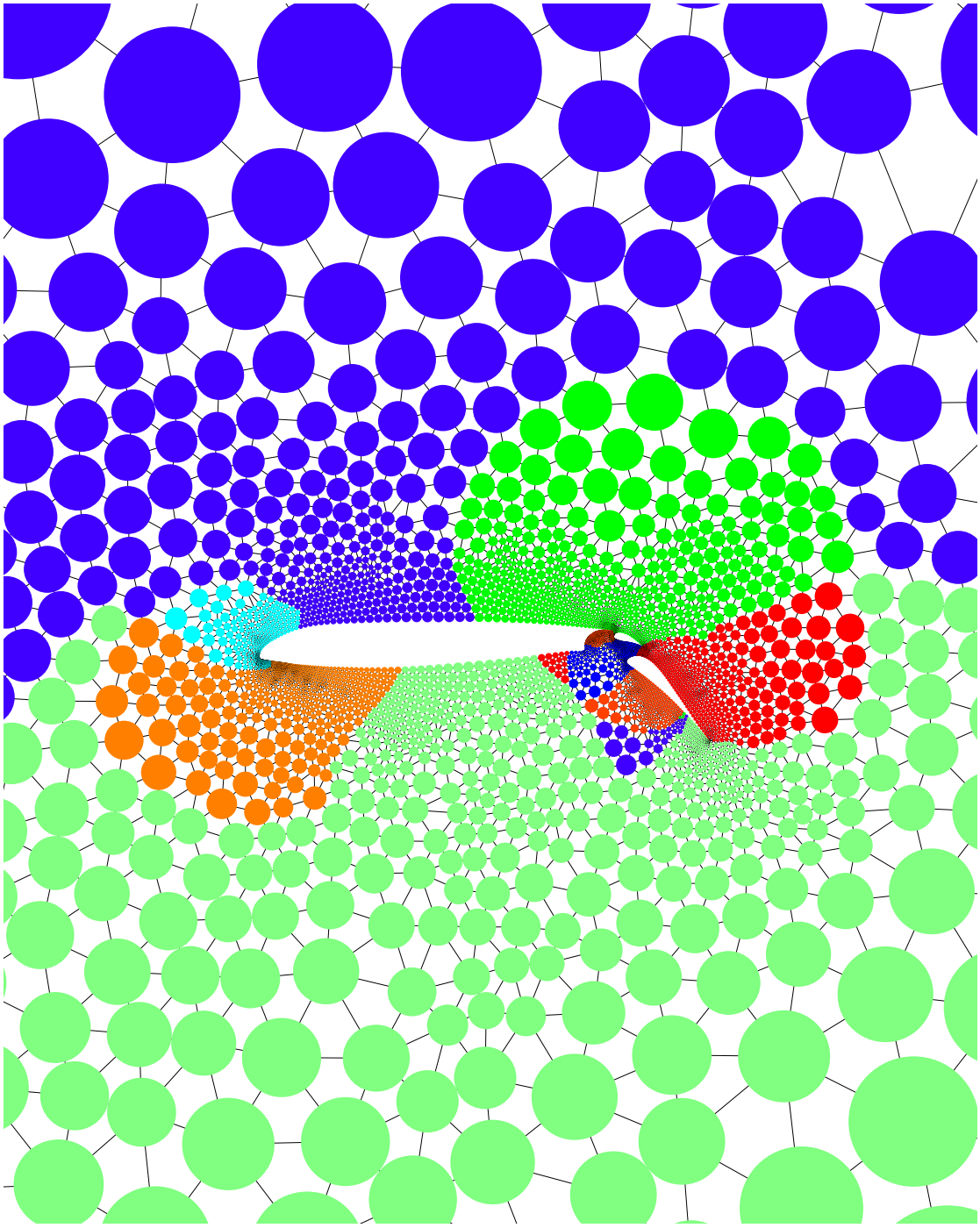


Figure A.10: Résultat du placement du graphe *ef3elt* sur H(3) avec la stratégie GP+FM+EX.

Bibliographie

[1] M. Alabau. *Une expression des algorithmes massivement parallèles sans communication explicite*. Thèse de Doctorat, LaBRI, Université Bordeaux I, 351 cours de la Libération, 33405 Talence, France, septembre 1994.

[2] M. Alabau, S. Chaumette, M.-C. Counilh, J.-M. Lépine, J. Roman, and B. Vauquelin. *A programming environment dedicated to a model of explicit parallelism*, volume 6 of *Advances in parallel computing*, pages 193–212. North-Holland, 1993.

[3] F. André and J.-L. Pazat. Le placement de tâches sur des architectures parallèles. *Technique et Science Informatiques*, pages 385–401, avril 1988.

[4] S. T. Barnard and H. D. Simon. A fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems. *Concurrency: Practice and Experience*, 6(2):101–117, 1994.

[5] D. Barth. *Réseaux d'interconnexion : Structures et Communications*. Thèse de Doctorat, LaBRI, Université Bordeaux I, 351 cours de la Libération, 33405 Talence, France, janvier 1994.

[6] D. Barth, F. Pellegrini, A. Raspaud, and J. Roman. On Bandwidth, Cutwidth, and Quotient graphs. Rapport de recherche, LaBRI, Université Bordeaux I, 351 cours de la Libération, 33405 Talence, France, 1993.

[7] A. Beguelin, J. Dongarra, A. Geist, W. Jiang, R. Manchek, and V. Sunderam. PVM 3.0 user's guide and reference manual. Technical report, Oak Ridge National Laboratory, 1993.

[8] A. Bel Hala. Congestion optimale du plongement de l'hypercube $H(n)$ dans la chaîne $P(2^n)$. *RAIRO Informatique Théorique et Applications*, 27(4):1–17, 1993.

[9] C. M. Bender and S. A. Orszag. *Advanced mathematical methods for scientists and engineers*, chapter 6.4 - Laplace's method and Watson's lemma, pages 261–276. MacGraw-Hill, second edition, 1984.

[10] C. Berge. *Graphes et Hypergraphes*. North Holland Publishing, 1971.

[11] M. J. Berger and S. H. Bokhari. A partitioning strategy for nonuniform problems on multiprocessors. *IEEE Transactions on Computers*, 36(5):570–580, mai 1987.

[12] S. H. Bokhari. On the mapping problem. *IEEE Transactions on Computing*, C-30(3):207–214, 1981.

[13] S. W. Bollinger and S. F. Midkiff. Processor and link assignment in multicomputers using simulated annealing. In *Proceedings of the 11th Int. Conf. on Parallel Processing*, pages 1–7. The Penn. State Univ. Press, août 1988.

[14] T. N. Bui, S. Chaudhuri, F. T. Leighton, and M. Sipser. Graph bisection algorithms with good average case behavior. *Combinatorica*, 7(2):171–191, 1987.

[15] P. Charrier and J. Roman. Partitioning and mapping for parallel nested dissection on distributed memory architectures. In *LNCS 634 – Proceedings of CONPAR'92*, pages 295–306. Springer-Verlag, septembre 1992.

[16] P. Z. Chinn, J. Chvàtalovà, A. K. Dewdney, and N. E. Gibbs. The Bandwidth Problem for Graphs and Matrices – A survey. *Journal of Graph Theory*, 6:223–254, 1982.

[17] F. R. K. Chung. *The Theory and Applications of Graphs*, chapter Some Problems and Results in Labelling of graphs, pages 255–263. John Wiley, 1981.

[18] M. Chung, F. Makedon, I. H. Sudborough, and J. Turner. Polynomial time algorithms for the min-cut problem on degree-restricted trees. *SIAM Journal of Computing*, 14(1):158–177, 1985.

[19] V. Chvátal. A remark on a problem of Harary. *Czechoslov. Math. Journal*, pages 109–112, 1970.

[20] J. Chvàtalovà. Optimal labelling of a product of two paths. *Journal of Discrete Maths*, 11:249–253, 1975.

[21] R. Cypher. Theoretical aspects of V.L.S.I. pin limitations. Technical Report TR 89-02-01, University of Washington, Seattle, Washington, février 1989.

[22] J. de Rumeur. *Communication dans les réseaux de processeurs*. Masson, octobre 1994.

[23] W. Donath and A. Hoffman. Algorithms for partitioning of graphs and computer logic based on eigenvectors of connection matrices. *IBM Technical Disclosure Bulletin*, 15:938–944, 1972.

[24] W. Donath and A. Hoffman. Lower bounds for the partitioning of graphs. *IBM Journal of Research and Development*, 17:420–425, 1973.

[25] F. Ercal, J. Ramanujam, and P. Sadayappan. Task allocation onto a hypercube by recursive mincut bipartitionning. *Journal of Parallel and Distributed Computing*, 10:35–44, 1990.

[26] T. Etzion and A. Lempel. Construction of de Bruijn sequences of minimal complexity. *IEEE Transactions of Information Theory*, IT-30(5):705–709, 1984.

[27] R. Feldmann and W. Unger. The Cube-Connected Cycles network is a subgraph of the Butterfly network. *Parallel Processing Letters*, 2(1):13–19, 1992.

[28] A. Feller. Automatic layout of low-cost quick-turnaround random-logic customs LSI devices. In *Proc. 13th Design Automation Conf.*, pages 79–85, San Francisco, 1976.

[29] W. Feller. *An introduction to probability theory and its applications*, volume 1. Willey, New York, 1950.

[30] C. M. Fiduccia and R. M. Mattheyses. A linear-time heuristic for improving network partitions. In *Proceedings of the 19th Design Automation Conference*, pages 175–181. IEEE, 1982.

[31] M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Math. J.*, 23:298–305, 1973.

[32] M. Fiedler. A property of eigenvectors of non-negative symmetric matrices and its application to graph theory. *Czechoslovak Math. J.*, 25:619–633, 1975.

[33] M. J. Flynn. Some computer organizations and their effectiveness. *IEEE Transactions on Computers*, 21(9):948–960, septembre 1972.

[34] J.-M. Garcia and P. Guyaux. LANDA – Logiciel pour la création et le lancement d'applications distribuées sur réseau local. Rapport LAAS 92218, LAAS-CNRS, 7 avenue du Colonel Roche, 31077 Toulouse, France, juin 1992.

[35] M. R. Garey, R. L. Graham, D. S. Johnson, and D. E. Knuth. Complexity results for bandwidth minimization. *SIAM Journal of Applied Mathematics*, 34(3):477–495, 1978.

[36] M. R. Garey and D. S. Johnson. *Computers and Intractablility: A Guide to the Theory of NP-completeness.* W. H. Freeman, San Francisco, 1979.

[37] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1:237–267, 1976.

[38] F. Gavril. Some NP-complete problems on graphs. In *Proc. 11th conference on information sciences and systems*, pages 91–95. John Hopkins University, Baltimore.

[39] J. A. George and J. W. H. Liu. *Computer solution of large sparse positive definite systems.* Prentice Hall, 1981.

[40] N. E. Gibbs, W. G. Poole, and P. K. Stockmeyer. A comparison of several bandwidth and profile reduction algorithms. *ACM Trans. Math. Software*, 2:322–330, 1976.

[41] J. R. Gilbert and R. E. Tarjan. The analysis of a nested dissection algorithm. *Numerische Mathematik*, 50:377–404, 1987.

[42] J. R. Gilbert and E. Zmijewski. A parallel graph partitioning algorithm for a message-passing multiprocessor. Technical Report TR 87-803, Cornell University, janvier 1987.

[43] E. M. Gurari and I. H. Sudborough. Improved dynamic algorithms for bandwidth minimization and the mincut linear arrangement problem. *Journal of Algorithms*, 5:531–546, 1984.

[44] S. W. Hammond. *Mapping unstructured grid computations to massively parallel computers.* PhD thesis, Rensselaer Polytechnic Institute, Troy, New-York, février 1992.

[45] P. Hansen and W. C. Giauque. Task allocation in distributed processing systems. *Operations Research Letters*, 5(3):137–143, août 1986.

[46] L. H. Harper. Optimal assignement of number of vertices. *J. SIAM*, 12:131–135, 1964.

[47] L. H. Harper. Optimal numberings and isoperimetric problems on graphs. *Journal of Combinatorial Theory*, 1:385–393, 1966.

[48] B. Hendrickson and R. Leland. The *chaco* user's guide. Technical Report SAND93–2339, Sandia National Laboratories, novembre 1993.

[49] B. Hendrickson and R. Leland. Multidimensional spectral load balancing. Technical Report SAND93–0074, Sandia National Laboratories, janvier 1993.

[50] B. Hendrickson and R. Leland. An empirical study of static load balancing algorithms. In *Proceedings of SHPCC'94, Knoxville*, pages 682–685. IEEE, mai 1994.

[51] High Performance Fortran Forum. *High Performance Fortran – Language specification V1.0*, may 1993.

[52] C. A. R. Hoare. Communicating Sequential Processes. *Communications of the ACM*, 21(8):666–677, avril 1978.

[53] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitionning graphs. *BELL System Technical Journal*, pages 291–307, février 1970.

[54] M. Laguna, T. A. Feo, and H. C. Elrod. A greedy randomized adaptative search procedure for the two-partition problem. *Operations Research*, pages 677–687, juillet-août 1994.

[55] T. H. Lai and A. S. Sprague. Placement of the processors of a hypercube. Technical report, 1988.

[56] F. T. Leighton. Complexity issues in V.L.S.I.. optimal layouts for the shuffle exchange graph and other networks. In *Foundations of Computing Series.* The MIT press, 1983.

[57] F. T. Leighton. *Introduction to Parallel Algorithms and Architectures: arrays, trees, hypercubes.* Morgan Kaufman Publisher, 1992.

[58] C. Leiserson. Fat-Trees: Universal networks for hardware-efficient supercomputing. *IEEE Transactions on Computers*, 34(10):892–901, octobre 1985.

[59] C. Leiserson, Z. Abuhamdeh, D. Douglas, C. Feynman, M. Ganmukhi, J. Hill, W. Hillis, B. Kuszmaul, M. Pierre, D. Wells, M. Wong, S. Yang, and R. Zak. The network architecture of the Connection Machine CM-5. Technical report, Thinking Machines, juillet 1992.

[60] M. Lin, R. Tsang, D. H. C. Du, A. E. Klietz, and S. Saroff. Performance evaluation of the CM-5 interconnection network. In *Proceedings of CompCon Spring'93*, 1993.

[61] R. J. Lipton, D. J. Rose, and R. E. Tarjan. Generalized nested dissection. *SIAM Journal of Numerical Analysis*, 16(2):346–358, avril 1979.

[62] J. W. Liu. *On reducing the profile of sparse symmetric matrices.* PhD thesis, University of Waterloo, Waterloo, Ontario, 1975.

[63] V. M. Lo. Heuristic algorithms for task assignment in distributed systems. In *International Conference on Distributed Computer Systems*, pages 30–39. IEEE, 1984.

[64] M. Lothaire. *Combinatorics on Words.* Addison Wesley, 1983.

[65] P. R. Ma, E. Y. S. Lee, and M. Tsuchiya. A task allocation model for distributed computing systems. *IEEE Transactions on Computers*, C-31(1):41–47, janvier 1982.

[66] B. Monien and H. Sudborough. Embedding one interconnection network in another. *Computing Supplements*, 7:257–282, 1990.

[67] T. Muntean and E.-G. Talbi. Méthodes de placement statique des processus sur architectures parallèles. *T.S.I., Technique et Science Informatiques*, 1991.

[68] T. Muntean and E.-G. Talbi. A parallel genetic algorithm for process-processors mapping. *High performance computing*, 2:71–82, 1991.

[69] K. Nakano. Linear layouts of generalized hypercubes. In *Proceedings of the 19th International Workshop on Graph-Theoretic Concepts in Computer Science.* Springer-Verlag, 1993.

[70] K. Nakano, W. Chen, T. Masuzawa, K. Hagihara, and N. Tokura. Cut width and bisection width of hypercube graph. *IEICE Transactions*, J73-A(4):856–862, avril 1990. En Japonais.

[71] D. M. Nicol. Rectilinear partitioning of irregular data parallel computations. *Journal of Parallel and Distributed Computing*, 23:119–134, 1994.

[72] T. Oakenshield. *Recursive Bipartitioning of Trolls with a Two-Handed Axe.* Moria Press, 1947.

[73] C. H. Papadimitriou. The NP-completeness of the bandwidth minimization problem. *Computing*, 16:263–270, 1976.

[74] F. Pellegrini. Bounds for the bandwidth of the d-ary De Bruijn graph. *Parallel Processing Letters – Special Number on Interconnection Networks*, 3(4):431–443, 1994.

[75] F. Pellegrini. Static mapping by dual recursive bipartitioning of process and architecture graphs. In *Proceedings of SHPCC'94, Knoxville*, pages 486–493. IEEE, mai 1994.

[76] F. Pellegrini. *Application de méthodes de partition à la résolution de problèmes de graphes issus du parallélisme.* Thèse de Doctorat, LaBRI, Université Bordeaux I, 351 cours de la Libération, 33405 Talence, France, janvier 1995.

[77] Performance Computing Industries. *CS-2 Product Description*, 1993.

[78] G. Persky, D. Deutsch, and D. Schweikert. LTX–A minicomputer-based system for automated LSI layout. *J. Design Automation and Fault Tolerant Computing*, 1:217–255, 1977.

[79] B. Plateau, P. Raynal, D. Trystram, and F. Vincent. Placement de tâches: un tour d'horizon des techniques efficaces. Technical report, IMAG, 46 avenue F. Viallet, 38031 Grenoble CEDEX, juin 1991.

[80] A. Pothen, H. D. Simon, and K.-P. Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal of Matrix Analysis*, 11(3):430–452, juillet 1990.

[81] J. Roman. Calculs de complexité relatifs à une méthode de dissection emboîtée. *Numerische Mathematik*, 47:175–190, 1985.

[82] C. Roucairol and P. Hansen. Cut cost minimization in graph partitioning. *Numerical and Applied Mathematics*, pages 585–587, 1989.

[83] F. Shahrokhi and L. A. Szekely. An algebraic approach to the uniform concurrent multicommodity flow – Theory and applications. Technical Report CRPDC-91-4, University of North Texas, 1991.

[84] C.-C. Shen and W.-H. Tsai. A graph matching approach to optimal task assignment in distributed computing systems using a minimax criterion. *IEEE Transactions on Computers*, C-34(3):197–203, mars 1985.

[85] H. D. Simon and S.-H. Teng. How good is recursive bipartition. Research report, NASA Ames Research Center, juin 1993.

[86] J. B. Sinclair. Efficient computation of optimal assignments for distributed tasks. *Journal of Parallel and Distributed Computing*, 4:342–362, 1987.

[87] W. F. Smith and I. Arany. Another algorithm for reducing bandwidth and profile of a sparse matrix. In *Proc. AFIPS 1976 NCC*, pages 341–352, Montvale, New Jersey, 1976. AFIP Press.

[88] H. S. Stone. Multiprocessor scheduling with the aid of network flow algorithms. *IEEE Transactions on Software Engineering*, SE 3(2):85–93, janvier 1977.

[89] E.-G. Talbi and T. Muntean. évaluation et étude comparative d'algorithmes d'optimisation combinatoire: application au problème de placement de processus. Rapport de Recherche RR 886-I, LGI-IMAG, 46 avenue F. Viallet, 38031 Grenoble CEDEX, avril 1992.

[90] C. D. Thompson. *A complexity theory for VLSI.* PhD thesis, Carnegie Mellon University, 1980.

[91] S. Trimberg. Automating chip layout. *IEEE Spectrum*, pages 38–45, 1982.

[92] R. van Driessche and D. Roose. Dynamic load balancing with an improved spectral bisection algorithm. In *Proceedings of SHPCC'94, Knoxville*, pages 494–500. IEEE, mai 1994.

[93] R. van Driessche and D. Roose. A graph contraction algorithm for the calculation of eigenvectors of the laplacian matrix of a graph with a multilevel method. Technical Report TW 209, Katholieke Universiteit Leuven, mai 1994.

[94] H. Yoshizawa, H. Kawanishi, and K. Kani. A heuristic procedure for ordering MOS arrays. In *Proc. of Design Automation Conference*, pages 384–393, 1975.

[95] O. Zone, D. Vanderstraeten, and R. Keunings. Un solveur direct parallèle basé sur une décomposition de domaine appliqué aux problèmes d'éléments finis. In *Actes des 6èmes Rencontres Francophones du Parallélisme, RenPar'6, Lyon*, pages 113–118. ENS Lyon, juin 1994.

Résumé

L'objet de cette thèse est d'étudier l'impact des méthodes de quotientement sur plusieurs problèmes issus du parallélisme et modélisés sous forme de graphes.

La première partie de la thèse, essentiellement théorique, est consacrée à l'étude de deux paramètres des graphes, qui sont la largeur de bande et la largeur de coupe. On obtient de nouveaux encadrements de ceux-ci pour plusieurs réseaux d'interconnexion, pour certains au moyen de techniques d'évaluation asymptotique.

La deuxième partie traite du problème du placement statique sur machine parallèle. Nous y présentons un algorithme de placement basé sur le bipartitionnement récursif conjoint du graphe de communication des processus et du graphe modélisant l'architecture de la machine parallèle, qui permet de placer efficacement tout graphe de processus valué sur tout type d'architecture.

Mots clés : diviser pour résoudre, graphe quotient, largeur de bande, largeur de coupe, évaluation asymptotique, placement statique, bipartition.

Abstract

The purpose of this thesis is to study the impact of quotienting methods over several problems related to parallelism and modeled in terms of graphs.

The first part of the thesis, mainly theoretical, is devoted to the study of two parameters of graphs, the bandwidth and the cutwidth. We obtain new lower and upper bounds of the parameters for several interconnection networks, for some of them by means of asymptotic evaluations.

The second part deals with static mapping onto parallel architectures. We present a mapping algorithm based on the dual recursive bipartitioning of both the process communication graph and the graph representing the target architecture, which allows the efficient mapping of any valuated process graph onto any valuated architecture graph.

Keywords : divide and conquer, quotient graph, bandwidth, cutwidth, asymptotic evaluation, static mapping, bipartition.