Laboratoire Bordelais de Recherche en Informatique
UMR 5800 - Université Bordeaux I, 351, cours de la Libération,
33405 Talence CEDEX, France

Document de synthèse de
l'activité scientifique de

Cyril Gavoille

Structures de Données Compactes et Distribuées

présenté le 7 décembre 2000 en vue d'obtenir
une habilitation à diriger des recherches en informatique
devant le jury composé de :

RASPAUD	André	*Président*
COURCELLE	Bruno	*Rapporteur*
HABIB	Michel	"
PELEG	David	"
KÖNIG	Jean-Claude	*Examinateur*
MÉTIVIER	Yves	"

Je remercie Isabelle, mon épouse, pour toute son attention.

Je dédie cette thèse
à mes parents.

Avant-propos

Ce document est une présentation de mes activités de recherches en informatique. J'ai pris mes fonctions de Maître de Conférences à l'Université de Bordeaux I le 1$^{\text{er}}$ septembre 1996, et, depuis cette date, mes travaux se déroulent au sein de l'équipe Combinatoire et Algorithmes Distribués du Laboratoire Bordelais de Recherche en Informatique (LaBRI) dans le thème « Graphes & Applications » dirigé par André Raspaud.

Ce document se veut être une synthèse. Pour garantir au lecteur une certaine densité de résultats je passerai sous silence ceux qui, plus originaux, ne s'inscrivent pas vraiment dans le cadre des structures de données compactes et distribuées. Il s'agit de l'article [CFG99] qui propose un algorithme de reconnaissance quasi-linéaire (à un facteur $\log^{O(1)} n$ près) du graphe de Knödel à $2n$ sommets, ainsi que l'article [BFG$^+$00] qui concernent des algorithmes de reconnaissance en temps linéaire pour les graphes de Cayley Abéliens coloriés suivant leurs générateurs, et quasi-linéaire pour un graphe de Cayley quelconque colorié suivant ses générateurs.

Mes recherches concernent les problèmes d'optimisations dans les graphes (paramètres de graphes et structures de données distribuées) pour des applications liées aux réseaux communiquants.

Mes recherches peuvent être vues sous deux angles différents.

1. Soit le calcul de paramètres sur les graphes. Par exemple, le calcul de la compacité d'un graphe, c'est-à-dire le nombre minimum d'intervalles k pour obtenir un routage par intervalles de plus courts chemins avec au plus k intervalles par arcs.
2. Soit la détermination d'algorithmes et de structures de données efficaces pour des problèmes de communications dans les réseaux.

Ces deux aspects se ramènent à des problèmes d'optimisations dans les graphes. Toutefois, ils se distinguent par le fait qu'on mette en avant la théorie ou les applications. Personnellement, je ne prendrai pas le risque de distinguer dans mes travaux la théorie des graphes de ses applications aux réseaux communiquants. Je pense que les problèmes de communications peuvent se résoudre par l'emploi de la théorie des graphes, tout comme la compréhension de la structure des graphes est stimulée et enrichie par les applications qui donnent de l'importance à certains paramètres inédits définis sur les graphes. La dualité "théorie et applications" va se retrouver tout au long de ce document.

Les chapitres 1 et 2 abordent le problème du routage compact. Le chapitre 1 présente une solution spécifique (le routage par intervalles), alors que le chapitre 2 aborde le cas général en étendant les mesures de complexité du chapitre précédent : espace mémoire, longueur des routes, temps de calcul du routage. Le problème du routage compact s'énonce de la façon suivante : étant donné un graphe (réseau de processeurs communiquants), déterminer un algorithme de routage dont la structure de données soit la plus compacte possible. Cet algorithme doit permettre à chaque message émanant d'une source quelconque, d'atteindre sa destination. Il doit de plus être distribué dans le sens où la route se construit de proche en proche uniquement par des décisions locales prises par les nœuds rencontrés. S'ajoutent à ceci des contraintes de performance du routage proprement dit (plus courts chemins ou proches des plus courts), de minimisation de l'espace mémoire local des nœuds et du temps de calcul pour le routage local.

Au chapitre 3, variante de la représentation implicite des graphes : l'étiquetage de distance. Étant donné un graphe d'une certaine famille, il s'agit de fournir un étiquetage des sommets de sorte que la distance entre deux sommets quelconques puisse être calculée (ou approximée) par une fonction utilisant uniquement les deux étiquettes des sommets comme paramètres. En particulier, la fonction peut dépendre de la famille de graphe, mais en aucun cas du graphe même dont sont extraites les étiquettes. Ce problème a des applications pour les réseaux communiquants, chaque nœud ayant ainsi la possibilité d'estimer des distances, sans échange de messages ni table coûteuse en espace mémoire local.

Enfin, dans le dernier chapitre « Généralisation et perspectives », il apparaît que les deux problèmes développés précédemment (routage compact et étiquetage de distance) sont comparables en de nombreux points, voire constituent deux aspects d'un problème plus général. Ainsi, tout au long des chapitres 1, 2 et 3, allons nous étudier trois grands types de paramètres :

1. **L'espace mémoire :** compacité et nombre d'intervalles (au chapitre 1), ou taille de la structure de données (taille des tables de routage, longueur des étiquettes de distance).
2. **La qualité de la solution générée :** dilatation ou longueur de la route, facteur d'étirement (pour le routage compact), approximation de la distance (pour l'étiquetage de distance).
3. **Le temps de calcul de la solution :** latence ou temps de calcul local de la fonction de routage, temps de décodage de la distance (pour l'étiquetage de distance).

En terme de méthode, la présentation et la structure de ce document viennent se calquer sur mes recherches, dans le sens où nous irons du plus spécifique (le routage par intervalles), au plus général (les structures de données compactes et distribuées).

Table des matières

CHAPITRE 1

Routage par intervalles

Dans ce chapitre, nous allons montrer que la méthode du routage par intervalles est à la base de nombreuses stratégies de routage. Elle est même au cœur des problèmes de structures de données compactes et distribuées dépassant le cadre du routage.

1.1 Le problème du routage

Le problème que l'on se pose est le suivant : étant donné un graphe (réseau de processeurs communiquant) fixé à l'avance, fournir dans chaque nœud (ou routeur) un algorithme de routage aussi efficace que possible. Il convient d'expliciter ce qu'on entend par « algorithme de routage » et par « efficace ». Un *algorithme de routage* est une fonction (calculable) qui à chaque message et chaque nœud détermine le lien sur lequel le message doit être transmis, ceci en fonction de sa destination contenue dans l'en-tête du message. Le terme *efficace* regroupe un ensemble de qualités telles que : les routes générées par l'algorithme sont des plus courts chemins dans le graphe en question ; le temps de calcul de la fonction est faible ; le nombre de routes utilisant le même lien est faible, *etc.* Nous reviendrons sur ces définitions au chapitre suivant.

Par exemple, on peut donner l'algorithme standard de routage dans l'hypercube. Notons x le nom binaire (dans l'hypercube) du sommet courant, et y la destination constituant l'en-tête du message.

> « Si $y = x$, alors le message a atteint sa destination. Sinon le renvoyer sur l'arête de dimension i si les mots x et y diffèrent au i-ème bit. »

Si chaque routeur possède une copie de cet algorithme, alors nous obtenons un algorithme distribué dans le sens où la décision est prise localement en chaque sommet. Cet algorithme est dit *de plus courts chemins* car, on le vérifie facilement, la route générée est la plus courte possible. Pour le sommet x donné, on note $R_x(y)$ la fonction qui pour toute destination y retourne l'entier i défini par l'algorithme ci-dessus. La fonction R_x possède un argument, y, mais l'algorithme définissant R_x ne dépend que de x. Autrement dit la fonction R_x peut être mise en œuvre par un programme de taille $\log n + O(1)$ bits[1] dans chaque sommet, n étant le nombre de sommets de l'hypercube considéré. En effet, il suffit de coder le nom binaire x dans le programme implémentant la fonction R_x. À ce titre cette fonction de routage possède une mise en œuvre relativement compacte.

La ou l'une des spécificités de ce type de problème est qu'il s'agit de fournir, pour un graphe donné, une fonction, ou plus exactement une collection de fonctions, pour chaque sommet. Ses fonctions s'expriment sur un domaine fini, les réseaux sont supposés avoir au plus n sommets. Par conséquent, sans contrainte supplémentaire, une solution générale consiste à prouver l'existence de ces fonctions puis de les tabuler dans chaque routeur satisfaisant ainsi la contrainte d'efficacité. Cependant, tabuler complètement une fonction est souvent une solution inefficace quant à l'espace requi par de telles tables (il faut stocker toutes les valeurs possibles !). Il s'agit donc de donner des fonctions de routage calculables avec des structures de données aussi compactes que possible.

Dans toute la suite, sauf précisions particulières, nous allons considérer les réseaux comme des graphes connexes et non orientés. Cependant la plupart des résultats s'étendent naturellement aux graphes orientés et fortement connexes.

1.2 Principe du routage par intervalles

Le routage par intervalles a été introduit il y a 15 ans par Santoro et Khatib [SK85] puis étendu par van Leeuwen et Tan [vLT87]. Cette méthode consiste à :

1. numéroter tous les sommets du graphe par des entiers distincts allant de 1 à n ;
2. affecter à chaque lien de sortie un (ou plusieurs) intervalle(s) d'entiers[2] de sorte qu'au sommet x un message à destination y sera envoyé sur le lien associé à l'intervalle $[a, b]$ (ou à l'union d'intervalles) contenant le numéro d'y.

Il faudrait ajouter à cette définition rapide que l'ensemble des intervalles associés à x doit couvrir l'ensemble $\{1, \dots, n\}$ et qu'ils sont deux à deux disjoints. Sans plus insister, cette stratégie fournit un algorithme de routage très simple similaire à ceux que l'on peut rencontrer dans les couloirs des hôtels, illustré aussi par la figure 1.1.

Il faut noter que cette méthode de routage est particulièrement bien adaptée au cas des arbres. Comme le montre la figure 1.1, il suffit d'une numérotation des sommets suivant un parcours en profondeur-d'abord, pour obtenir un routage par intervalles avec au plus

1. Dans toute la suite du document nous noterons $\log n$ le logarithme en base deux de n.
2. On s'autorise aux intervalles cycliques, c'est-à-dire modulo n, comme $[a, b]$ avec $a > b$ qui représente l'ensemble des entiers $\{a, a+1, \dots, n, 1, 2, \dots, b\}$.

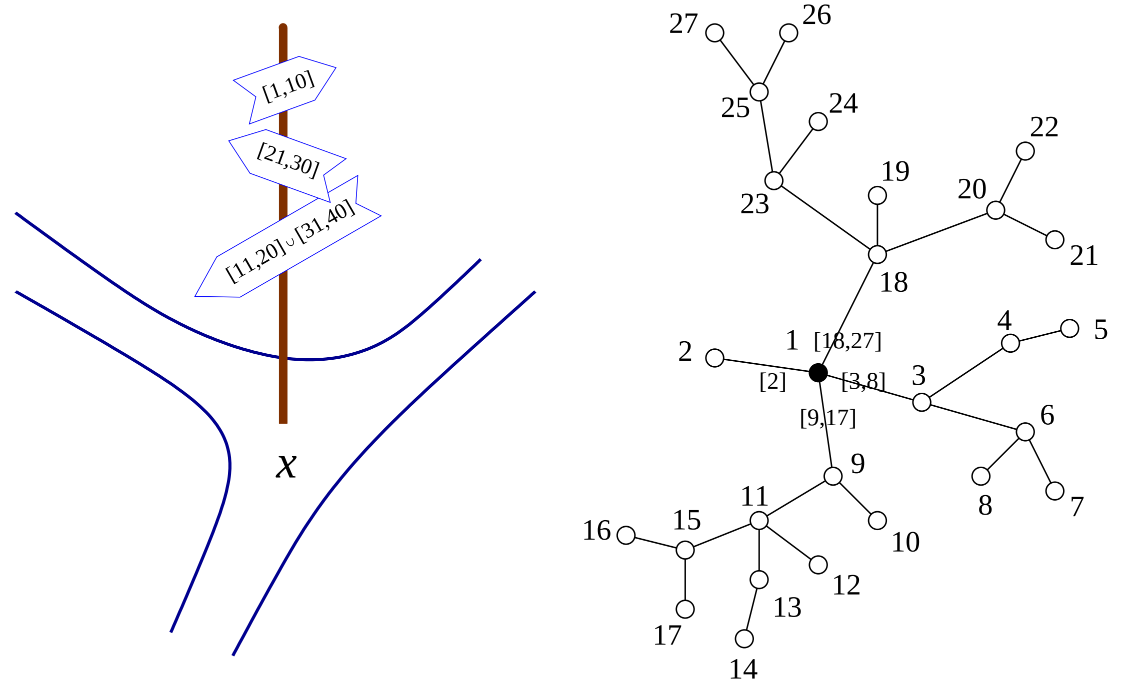


FIG. 1.1 – *Principe du routage par intervalles et application aux arbres.*

un intervalle par arc [3] : chaque arc définit un sous-arbre dont les sommets sont numérotés consécutivement (modulo le nombre de sommets de l'arbre dans le cas de l'arc menant au père), cf. [SK85]. Notons tout de suite que le fait de pouvoir établir un routage par intervalles sur un arbre implique que pour tout graphe (connexe) on peut aussi le faire, en prenant alors un arbre couvrant. De plus, en sommant sur toutes les arêtes de l'arbre, le nombre total d'intervalles est $2(n-1)$, soit $O(n \log n)$ bits d'informations au total, ou encore $O(\log n)$ bits par sommet en moyenne. Nous reviendrons sur ce dernier aspect au chapitre suivant.

Un autre avantage de cette méthode est que l'intervalle contenant la destination peut être déterminé en temps $O(\log n)$ en chaque sommet, par tri des intervalles (voir le paragraphe 2.3.1 page 26 pour plus de détails).

3. Dans la suite nous utilisons le terme d'arc plutôt que de lien étant donné que chaque arête possède en fait deux intervalles, un à chaque extrémité. On peut voir une arête comme l'ensemble de deux arcs opposés. Notons aussi que la plupart des résultats peuvent être appliqués et étendus aux graphes orientés.

1.3 Le routage par intervalles utilisé comme brique de base

Bien que devenu un thème à part entière (l'article [Gav00b] faisant l'état de l'art sur cette méthode, dénombre plus de 80 références bibliographiques), le routage par intervalles sert de lemme dans de nombreux résultats, et a été utilisé par de nombreux auteurs comme « brique de base ». Nous en présentons rapidement quelques uns.

Tout d'abord, Frederickson et Janardan ont donné dans [FJ88] une très belle caractérisation des graphes planaires-extérieurs (*outerplanar*), comme étant les seuls graphes pondérés[4] possédant un routage par intervalles de plus courts chemins et avec un intervalle par arc. Bien sûr, si l'on considère une pondération uniforme (un poids identique sur toutes les arêtes), d'autres familles de graphes entrent dans cette catégorie de graphes, par exemple l'hypercube ou les grilles généralisées (voir [Gav00b]). Les mêmes auteurs proposeront aussi une généralisation aux graphes de genre plus élevé sans toutefois donner une caractérisation.

Dans un cadre plus général, et dans une série d'articles, Awerbuch, Bar-Noy, Frederickson, Janardan, Linial, Peleg, Upfal [ABNLP89, ABNLP90, AP90, FJ86, FJ89, FJ90, PU88, PU89] et d'autres, ont défini des méthodes de routage établissant un compromis entre la quantité d'informations utilisées pour le routage et la longueur des routes générées. Cette longueur est mesurée en terme de facteur d'étirement (rapport maximal entre la longueur de la route et la distance). Les performances de toutes ces méthodes seront reprises et détaillées au chapitre 2. Cependant, le point commun de toutes ces méthodes est qu'elles utilisent le routage par intervalles comme schéma de base. Elles font appel à des décompositions hiérarchiques par régions (*clusters*), décompositions plus ou moins raffinées suivant les variantes (optimisation de l'espace mémoire local ou optimisation de la somme des informations stockées dans le graphe), et les familles de graphes considérés (graphes pondérés ou non, graphes planaires, graphes c-décomposables, *etc*). Elles superposent plusieurs collections d'arbres couvrants, et sont de la forme :

> « Si la destination fait partie de la région locale du sommet courant, utiliser le routage par intervalles dans cette région, sinon utiliser une seconde méthode, définie sur un autre niveau dans la hiérarchie des régions. »

Le routage par intervalles apparaît dans ce cas intéressant, car peu gourmand en espace mémoire. En effet, si un seul intervalle par arc est utilisé pour l'arbre couvrant une région de N sommets, on obtient un total de $O(N)$ intervalles, c'est-à-dire $O(N \log N)$ bits sur toute la région, en sommant sur toutes les arêtes de l'arbre. C'est très peu en comparaison des tables de routage qui nécessitent jusqu'à $O(N \log N)$ bits par sommets (cf. page 17).

4. Un graphe pondéré est un graphe dont ses arêtes (ou ses arcs s'il est orienté) possèdent un poids non nul, appelé aussi coût. Sauf précision particulière, on supposera que cette pondération induit une métrique sur le graphe, c'est-à-dire respecte l'inégalité triangulaire.

L'article de Frederickson et Jarnardan [FJ89] ajoute une autre utilisation du routage par intervalles pour le cas spécifique des graphes planaires. Méthode également hiérarchique, à chaque niveau de la décomposition, le séparateur en cycle de Miller est utilisé [Mil86], et les sommets du séparateur sont numérotés consécutivement. Il est montré que le routage entre les sommets d'un niveau donné et ce cycle séparateur, peut être codé en utilisant un routage par intervalles.

Enfin, certains auteurs ont utilisé le routage par intervalles dans le cadre des réseaux dynamiques (ajout/suppression de sommets ou de liens, variation dynamique des poids sur les arêtes). En voici deux exemples :

– Afek, Gafni et Ricklin dans [AGR89] ont simplement adapté le routage par intervalles dans les arbres dynamiques (ajout de sommets seulement). Ils fournissent un protocole distribué de maintien des intervalles dans chacun des sommets de l'arbre. Ce protocole utilise $O(\log n)$ messages de contrôle par sommet (en coût amorti), et génère une structure de données de $O(\log^2 n)$ bits par sommet et $O(\log^3 n)$ bits par arc, où n est la taille finale de l'arbre. De plus, l'algorithme de routage produit est de plus courts chemins. Les techniques s'inspirent aussi de [AAPS87].

– Bodlander, van Leeuwen, Tan et Thilikos dans [BLTT97] ont proposé des caractérisations en terme de largeur arborescente et de mineurs de graphe, pour les graphes pondérés supportant un routage par intervalles avec au plus k intervalles par liens le long de chemins minimaux et ceci quel que soit la variation dynamique des poids (dans ce cadre il n'y a donc pas besoin de mise à jour des intervalles). Il en découle de nombreux algorithmes de numérotation pour l'initialisation de la numérotation des sommets et des intervalles.

1.4 Quelques problèmes clés

Il y a deux objectifs à atteindre dans le routage par intervalles :

1. obtenir des routages le long de chemins les plus courts possibles ;
2. représenter les fonctions locales de routage avec le moins d'espace mémoire possible, c'est-à-dire utiliser le plus petit nombre possible d'intervalles.

Malheureusement, ces deux objectifs s'opposent le plus souvent. Nous n'allons pas reprendre ici tous les résultats connus, mais simplement mentionner les plus significatifs. Je passe délibérément sous silence les problèmes du routage par intervalles prenant en compte la charge des liens, l'interblocage, ou encore le routage sur plusieurs chemins (adaptatif). Pour plus de précisions et de nombreuses références, nous invitons le lecteur à consulter l'article [Gav00b].

1.4.1 Compacité

On définit la *compacité* d'un graphe G comme le plus petit entier k tel que G supporte un routage par intervalles de plus courts chemins avec au plus k intervalles par arc. Chaque intervalle pouvant être codé sur $O(\log n)$ bits, les graphes de compacité bornés possèdent donc des fonctions de routage qui peuvent être mises en œuvre avec au plus $O(d \log n)$ bits d'espace mémoire par sommet de degré d.

Précisons que pour obtenir un routage par intervalles de plus courts chemins il y a au moins deux difficultés :

1. trouver une numérotation des sommets ;
2. trouver des routes réalisant les plus courts chemins.

Trouver une permutation des sommets et des plus courts chemins dans un graphe revient à faire une recherche dans des domaines de cardinalité exponentielle en la taille du graphe. Savoir si un graphe est de compacité 1 est NP-complet [EMZ97], n'est donc pas surprenant, il en est de même pour la compacité 2 [Fla97].

Ceci dit, ces résultats de NP-complétude ne sont pas des obstacles aux méthodes de routage générales utilisant le routage par intervalles comme brique de base (paragraphe 1.3), puisque, dans des familles particulières (comme les arbres, où les graphes planaires-extérieurs, qui sont utilisés par les décompositions hiérarchiques), les numérotations donnant le nombre optimal d'intervalles sont très simples.

Beaucoup d'auteurs se sont intéressés à déterminer la compacité maximale d'un graphe à n sommets, ou bien la compacité maximale d'un graphe d'une famille donnée. Il faut remarquer que le nombre d'intervalles associés à un arc ne peut dépasser $n/2$. (S'il y a plus de $n/2$ intervalles, alors il existe des destinations consécutives qui peuvent être regroupées dans le même intervalle.) Utilisant la théorie des codes, nous avons proposé dans [GG98] la construction d'un graphe ayant une compacité d'au moins $n/12$ (et $n/8$ si n est une puissance de 2), prouvant ainsi que la compacité maximale est en $\Theta(n)$. Grâce à la complexité de Kolmogorov, cette borne inférieure a été améliorée à $n/4 - o(n)$ dans [Gav97], avant que dans [GP99] nous montrons que tout graphe a une compacité au plus[5] $n/4 + o(n)$, résolvant un problème plus général défini sur les matrices booléennes. Il faut noter que ces deux derniers résultats sont existentiels. Pour la borne supérieure nous donnons un algorithme "randomisé" trivial, mais il serait intéressant d'en trouver un déterministe. De plus, pour de nombreuses familles de graphes, les bornes sur la compacité maximale ne sont pas aussi fines (voir les questions ouvertes ci-après).

Enfin, dans [GP98] nous montrons entre autre qu'une fraction d'au moins $1 - 1/n$ de tous les graphes à n sommets ont une compacité au plus 2. En fait, pour cette même fraction de graphes, nous montrons qu'il est possible d'avoir un intervalle par arc excepté

5. Bien que présenté ici comme résultat asymptotique, nous donnons une borne précise valable pour tout $n \geqslant 1$ et valant $\alpha(n)$ défini ci-après dans l'équation (1.1) page 8.

sur $O(\log^3 n)$ arc par sommet où deux suffisent. Nous verrons un corollaire de ce résultat au paragraphe 2.2.4 page 21.

L'article [Gav00b] propose 14 questions ouvertes, dont certaines ont été résolues, voir les articles [FGM00, Man99, TL99]. Ci-dessous, on trouvera quelques unes d'entres elles encore non résolues.

Questions ouvertes

- Est-ce qu'une fraction de $1-o(1)$ de tous les graphes à n sommets sont de compacité 1 ?
- Quelle est la compacité maximale des graphes planaires ? Des graphes séries-parallèles ?

Pour les graphes planaires, nous savons qu'elle se situe asymptotiquement entre $\sqrt{3n/20}$ et $n/4$, (la borne inférieure reste en $\Omega(\sqrt{n})$ pour les graphes séries-parallèles de degré bornés [Gav96]). Les familles de graphes connues pour avoir une compacité bornée sont mentionnées dans [Gav00b].

Toujours concernant les graphes planaires, Chepoi et Rollin [CR00] ont montré que les quadrangulations du plan où tous les sommets internes ont un degré $\geqslant 4$ supportent un routage par intervalles de plus courts chemins avec 7 intervalles linéraires. Ils étendent le résultat avec un facteur d'étirement 2 pour des triangulations du plan où tous les sommets internes ont des degrés $\geqslant 6$.

Enfin, il faut noter que les problèmes de compacité sont ouverts pour certains graphes d'intersections comme les graphes triangulés, sans triple-astéroïde, graphes de permutation, graphes de cordes, *etc.* Seuls quelques résultats sont connus dans les graphes d'intervalles, et les k-arbres, cf. [FG98a, NN96, NS96].

1.4.2 Dilatation

La *dilatation* d'un routage par intervalles est la longueur de la plus longue route induite par ce routage. Trouver un compromis entre le nombre d'intervalles et la dilatation est un problème naturel au regard des résultats ci-dessus sur la compacité (lorsque les plus courts chemins sont requis).

Généralement, on fixe le nombre d'intervalles et l'on cherche le routage minimisant la dilatation. Par exemple, si un seul intervalle doit être utilisé, la technique de l'arbre couvrant montre que la dilatation ne dépassera pas $2D$, où D représente diamètre du graphe. (D'ailleurs, [TL97] montrent, que dans le pire des cas, on ne peut pas espérer faire mieux que $2D-3$ avec un seul intervalle.) La dilatation minimum varie donc entre D et $2D$.

Kráľovič, Ružička et Štefankovič dans [KRŠ97] ont montré que $O(\sqrt{n \log n})$ intervalles suffisent pour garantir dans tout graphe une dilatation de $\lceil 3D/2 \rceil$. Nous avons complété ce résultat en montrant dans [Gav00a] qu'une dilatation de $\lfloor 3D/2 \rfloor - 1$ nécessite au moins

$\Omega\left(\frac{n}{D\log(n/D)}\right)$ intervalles dans le pire des cas.

La technique présentée dans [KRŠ97] est intéressante car elle peut être généralisée. Elle consiste à choisir dans le graphe G un ensemble $X = \{x_1, \ldots, x_k\}$ de k sommets de sorte que tout sommet de G soit à une distance au plus $\lceil D/2 \rceil$ de X. Ils montrent qu'un ensemble X tel que $k = O(\sqrt{n \log n})$ existe toujours. Les sommets de G sont ensuite partitionnés en k régions $R_1, \ldots, R_k$ de sorte que chaque R_i corresponde à l'ensemble des sommets les plus proches de x_i de n'importe quel autre $x_j \in X$ (on casse l'égalité arbitrairement en s'assurant cependant que chaque R_i induise un sous-graphe connexe). De plus, en chaque x_i on enracine un arbre couvrant de plus courts chemins de G, T_i. Chaque région R_i est ensuite numérotée par un ensemble d'entiers consécutifs, $[a_i, b_i]$. Des sous-intervalles de $[a_i, b_i]$ sont affectés le long de l'arbre T_i (restreint aux sommets de R_i) similairement à la technique de l'arbre présentée précédemment, cf. [SK85]. À l'exception près que les intervalles cycliques modulo n sont remplacés par deux intervalles linéaires[6] de $[a_i, b_i]$. On complète le routage en associant l'intervalle $[a_j, b_j]$ à tout arc (u, v), $u \in R_i$, $i \neq j$, appartenant à l'arbre T_j et dirigé vers sa racine x_j.

On peut vérifier facilement que la route de $x \in R_i$ à $y \in R_j$ (avec éventuellement $i = j$) suit d'abord un plus court chemin vers x_j (sans forcément l'atteindre) avant d'atteindre y par un plus court chemin le long de T_j. La longueur de la route est bornée par $D + \lceil D/2 \rceil = \lceil 3D/2 \rceil$, en se souvenant que x est à distance au plus de D de x_j et que, par construction, x_j est à une distance au plus $\lceil D/2 \rceil$ de y_j.

Le nombre d'intervalles associés à un arc est au plus $k + 1$ (au plus 2 pour l'arbre à l'intérieur de R_i, et 1, au pire, pour chacune des $k-1$ autres régions). En fait, utilisant notre résultat général [GP99] appliqué aux régions que l'on peut réordonner entre elles, on peut décroître ce nombre et montrer qu'il est borné par $k/2 + 1$ et même $\alpha(k) + 1 = k/4 + o(k)$ si $k/\log n \to +\infty$, avec :

$$\alpha(n) \ = \ \frac{n}{4} + \frac{1}{4}\sqrt{2n\ln(3n^2)} \ . \tag{1.1}$$

Nous allons retrouver cette construction dans les paragraphes suivants.

Question ouverte

- Prouver ou infirmer que tout graphe planaire de diamètre D supporte un routage par intervalles de dilatation $\lceil (1+\epsilon)D \rceil$ avec $O(1/\epsilon)$ intervalles par arc, pour tout constante $\epsilon > 0$.

6. Concernant les intervalles, linéaire s'oppose à cyclique.

1.4.3 Ensemble r-dominant et dilatation

Un ensemble r-dominant pour un graphe G est un sous-ensemble de sommets X tel que tout sommet de G est à distance au plus r de X. On note $\gamma_r(G)$ la cardinalité du plus petit ensemble r-dominant de G.

Il est connu que tout graphe de diamètre D vérifie $\gamma_{\lceil D/2 \rceil}(G) = O(\sqrt{n \log n})$ (cas qui se ramène à un ensemble 1-dominant dans un graphe de diamètre deux). Une simple généralisation du schéma présenté au paragraphe 1.4.2 montre que si $\gamma_r(G) = k$, alors G possède un routage par intervalles de dilatation $D + r$ avec au plus $\alpha(k) + 1$ intervalles.

Bien que le résultat soit intéressant pour tout r, il semble que la valeur $r = \lceil D/2 \rceil$ joue le rôle d'une valeur seuil, spécialement pour les graphes planaires. En effet, pour $r = \lceil D/2 \rceil - 1$, $\gamma_r(G)$ n'est pas borné, même pour un arbre. (Par exemple, $\gamma_r(G) = \Theta(n/r)$ pour une subdivision du graphe $K_{1,\Theta(n/r)}$.) Les travaux que nous avons initialisés avec Raspaud et Sopéna [GRS00] montrent que pour les graphes planaires-extérieurs et les graphes séries-parallèles, $\gamma_r(G)$ est borné par 2, respectivement par 3, pour $r = \lceil D/2 \rceil$. Chepoi et Vaxes [CV00] ont montré que toute triangulation du plan dont tous les sommets internes ont un degré $\geqslant 6$ vérifie $\gamma_r(G) \leqslant 2$. À l'heure actuelle, nous avons seulement des cas extrêmes où $\gamma_r(G) = 4$. Voir la figure 1.2. Pour tout graphe planaire de diamètre $D = 2$, MacGillivray et Seyffarth [MS96] ont montré que $\gamma_1(G) \leqslant 3$, borne pouvant être atteinte, et que $\gamma_1(G) \leqslant 10$ pour $D = 3$, donc $\gamma_2(G) \leqslant 10$.

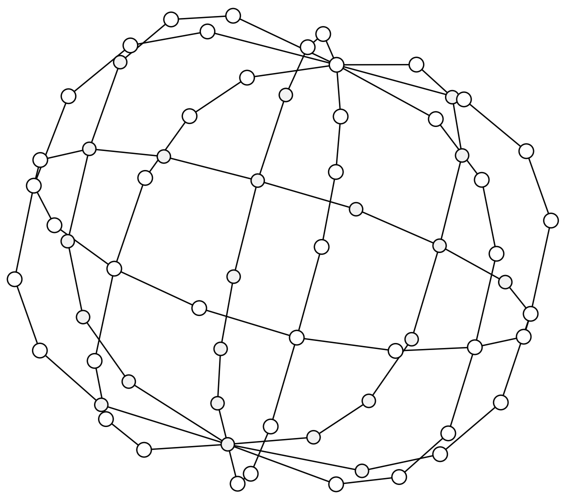


FIG. 1.2 – *Un graphe planaire G de diamètre 8 (plongé sur la sphère) tel que $\gamma_4(G) = 4$.*

Question ouverte

- Prouver ou infirmer que $\gamma_r(G)$ est constant pour tout graphe planaire G de diamètre au plus $2r$. Généraliser aux graphes de genre supérieur.

L'affirmative permettrai de répondre positivement à la question ouverte de la page 8 pour $\epsilon = 1/2$.

Dans une première étape, il serait intéressant de montrer que $\gamma_r(G)$ est borné par une fonction du diamètre. Sans conjecturer que le contre-exemple de la figure 1.2 est optimal, remarquons néanmoins que si $\gamma_r(G) \leqslant 4$ pour tout G planaire, alors, par l'équation (1.1), le nombre d'intervalles pour une dilatation $\lceil 3D/2 \rceil$ serait au plus $4/2 + 1 = 3$. Pour de plus amples détails sur ce problème, voir le document accessible sur le Web à l'adresse : http://dept-info.labri.u-bordeaux.fr/graphes/open_problem.html.

1.4.4 Facteur d'étirement

Il est parfois plus intéressant d'exprimer l'efficacité d'un routage par le rapport des longueurs de route sur la distance, plutôt que la longueur maximale des routes en fonction du diamètre. Certes, la méthode de l'arbre couvrant de [SK85] garantit une dilatation au plus deux fois le diamètre, mais appliquée au cycle, elle produira une route de longueur $n-1$ pour un couple de sommets qui sont pourtant voisins. On pourrait chercher des routages préservant autant que possible la localité du graphe[7].

Étant donné un routage par intervalles R sur un graphe G et un couple de sommets (u,v), on note $\rho_R(u,v)$ la longueur de la route de u à v induite par R, et $d_G(u,v)$ la distance de u à v dans G. Ces notations s'étendent aux graphes pondérés et/ou orientés.

Le *facteur d'étirement* et le *facteur d'étirement moyen* d'un routage R sur G sont respectivement les valeurs définies par :

$$s \;=\; \max_{u \neq v} \frac{\rho_R(u,v)}{d_G(u,v)} \qquad \text{et} \qquad \overline{s} \;=\; \frac{1}{n(n-1)} \sum_{u \neq v} \frac{\rho_R(u,v)}{d_G(u,v)} \;.$$

L'exemple du cycle démontrant l'inefficacité de la méthode des arbres couvrants est trompeur. En effet, dans ce cas, il est facile de se convaincre que le facteur d'étirement moyen est bien meilleur que $n-1$. Il est borné par $1.625 + o(1)$. Dans [GL99], nous avons montré que pour tout graphe de n sommets, il est possible de choisir un arbre couvrant, de sorte que le facteur d'étirement moyen induit par le routage par intervalles sur cet arbre est borné par $\nu(n) \leqslant \exp\left(O\left(\sqrt{\log n \cdot \log\log n}\right)\right)$. Ce résultat se déduit grandement d'un résultat d'Alon, Karp, Peleg et West [AKPW95] sur la théorie des jeux. Ils conjecturent que $\nu(n) = O(\log n)$.

Le lecteur aura remarqué que l'optimisation de la dilatation présentée au paragraphe 1.4.2 ne respecte pas la localité, c'est-à-dire n'a pas forcément un facteur d'étirement borné.

Nous avons corrigé dans [EGP98] ce défaut, en construisant, pour tout graphe, un routage avec au plus $O(\sqrt{n \log n})$ intervalles par arcs, une dilatation au plus $\lceil 3D/2 \rceil$ et un

7. Dans le cas du cycle, on peut bien évidemment réaliser un routage de plus courts chemins avec 1 intervalle par arc.

facteur d'étirement au plus 5. De plus le facteur d'étirement moyen n'est que de 3. Nous présentons un résultat similaire dans le cas des graphes pondérés (la dilatation garantie n'est plus que $2D$, D étant le diamètre pondéré du graphe, c'est-à-dire le coût de la distance la plus grande dans le graphe).

Sans entrer dans les détails, nous bornons le facteur d'étirement moyen en montrant que pour toute paire $\{u,v\}$:

$$\frac{\rho_R(u,v)}{d_G(u,v)} + \frac{\rho_R(v,u)}{d_G(v,u)} \leqslant 6 . \tag{1.2}$$

Ainsi le « facteur d'étirement sur un aller-retour » que nous avons introduit semble pertinent. Cette nouvelle mesure sera d'ailleurs réutilisée activement par Cowen et Wagner [CW00] dans le cadre du routage compact dans les graphes orientés.

Question ouverte

- Étendre le résultat précédent en montrant que pour tout entier $k > 0$, tout graphe supporte un routage par intervalles avec $O(k \cdot n^{1/k} \log^{O(1)} n)$ intervalles par arc et avec un facteur d'étirement borné par une fonction dépendante que de k (éventuellement une exponentielle en k).

1.4.5 Diffusion et routage par intervalles

Ce problème a été proposé par David Peleg en 1997, et résulte du constat suivant : l'établissement d'un routage par intervalles ou d'un routage compact est difficile (en terme de calculs distribués par exemple), surtout si les plus courts chemins sont requis. Ainsi, n'est-il pas possible d'utiliser ce calcul et les structures de données générées (intervalles et numéros des sommets) pour réaliser d'autres tâches distribuées?

La diffusion consiste à émettre un message depuis un sommet source vers tous les autres. La diffusion suit le schéma d'un algorithme distribué, et, suivant le modèle de communications sous-jacent du réseau, plusieurs mesures de complexité peuvent être définies. Le temps de diffusion est le temps maximum mis par l'algorithme pour informer tous les sommets du graphe, maximisé sur toutes les sources possibles. Cette complexité est utilisée dans le cas de réseaux synchrones où l'on suppose l'existence d'une horloge globale permettant simultanément à tous les sommets, et à chaque top d'horloge, d'émettre ou de recevoir un message avec leurs voisins. Ce temps dépend aussi du modèle de communication : modèle 1-port si un sommet ne peut communiquer qu'avec 1 seul de ses voisins à la fois, Δ-port s'il peut communiquer avec tous ses voisins simultanément. On distingue aussi le modèle *half* et *full*-duplex. (Se reporter à [dR94] pour une description plus précise de ces modèles.) Dans le cas asynchrone, on remplace le temps de diffusion par le nombre total de messages échangés. En effet, tous les messages transmis entre voisins peuvent être déphasés dans le temps, et donc, dans le pire des cas, un algorithme de diffusion échangeant N messages peut prendre un temps N.

La complexité en nombre de messages varie entre $\Omega(n)$ et $O(m)$, où m représente le nombre d'arêtes. En effet, d'une part la réception du message par tous les sommets différents de la source nécessite $n-1$ messages, et, d'autre part, la diffusion peut toujours être réalisée en imposant à tous les sommets de transmettre le message reçu à tous leurs voisins. Cette complexité dépend de la quantité d'informations mise à disposition de l'algorithme en chaque nœud, mais aussi de la taille maximale des en-têtes. Pour plusieurs auteurs, la quantité d'informations est mesurée par le rayon r tel que chaque sommet connaît son voisinage dans le graphe jusqu'à une distance r, pour $r \in \{0, 1, \ldots, n\}$. Se référer aux articles [ACK+98, AGPV90, DKKP99] qui présentant, en fonction de r, la complexité en nombre de messages.

Dans notre approche, nous supposons que chaque sommet connaît son numéro et les intervalles associés à ses arcs incidents pour un routage par intervalles R fixé. La relation entre le routage par intervalles et la diffusion avait déjà été étudiée par van Leeuwen et Tan [vLT87]. Ils avaient prouvé que la diffusion (par la construction préliminaire d'un arbre couvrant de plus courts chemins) ainsi que l'élection d'un leader pouvaient être résolues avec une complexité de messages en $O(n)$ dans le cycle et en $O(n+m)$ dans un graphe quelconque. Sans connaissance aucune (sans numérotation des sommets et des arêtes), les bornes sont $\Theta(n \log n)$ dans le cycle et $\Theta(m + n \log n)$ pour un graphe quelconque [GHS83]. Eilam, Peleg, Tan et Zaks [EPTZ97] ont montré que la famille des graphes supportant un routage par intervalles avec un intervalle par arc et représentant tous les plus courts chemins (les intervalles pouvant donc ne plus être deux à deux disjoints), a une complexité de messages optimale de $n-1$ pour la diffusion. Malheureusement, cette famille de graphes est très réduite puisqu'il s'agit essentiellement d'un chemin où chaque sommet peut être une clique. Le résultat sera étendu par de la Torre, Narayanan et Peleg [dlTNP98] à tous les routages par intervalles ayant la propriété SSR-TREE (pour chaque nœud x l'ensemble des routes d'origine x doit former un arbre couvrant de plus courts chemins). Malheureusement, tous les routages par intervalles de plus courts chemins n'ont pas forcément cette propriété.

Finalement, dans [FGM00], nous montrons que tout graphe de compacité 1 a une complexité de messages au plus $\Theta(n)$ pour la diffusion. Nous montrons aussi que ces graphes ont un algorithme distribué de construction d'arbres couvrants de plus courts chemins et d'élection de leader d'une complexité de messages en $\Theta(n)$, améliorant donc le résultat précédent en $O(m+n)$ de van Leeuwen et Tan [vLT87], tous ces résultats supposant la libre utilisation de la numéroation des sommets et des intervalles.

Le protocole de diffusion que nous proposons, appelé protocole *up/down*, est très simple. La source x envoie deux messages, l'un à destination de $x-1$, et l'autre à destination de $x+1$. Les en-têtes de ces messages sont sur $\log n + O(1)$ bits et sont composés du numéro de destination (appelé sommet cible) et d'un bit indiquant le type de message : *up* ou *down*. Lorsqu'un sommet y reçoit un message dont la cible est différente de y, il le retransmet grâce à son routage par intervalles. Sinon, la cible devient $y+1$ (dans le cas d'un message *up*) ou $y-1$ (pour un message *down*). L'en-tête changé, le message est retransmis vers la nouvelle cible, et ainsi de suite jusqu'à ce que les sommets étiquetés 1 et n soient atteints. D'autres protocoles sont possibles, et bien évidemment la difficulté réside dans l'analyse du nombre de messages échangés dans le pire des cas et non pas dans l'élaboration du protocole lui-même. La complexité du protocole *up/down* est $C_n = \sum_{i=1}^{n} d_G(i, i+1)$, où $d_G(x,y)$ est la distance entre les sommets étiquetés x et y. Nous montrons en fait que $C_n \leqslant 3n$ dans le cas où tous les intervalles sont linéaires, et $C_n \leqslant 9n$ dans le cas des intervalles cycliques. L'analyse consiste à étudier la séquence des étiquettes rencontrées entre les sommets cibles y et $y+1$ (où similairement y et $y-1$). Par exemple, dans le cas des intervalles linéaires, les étiquettes des sommets intermédiaires ne peuvent que décroître. De plus, si l'on exclut de cette séquence les blocs d'étiquettes consécutives, on constate que les sommets restants ne sont pas très éloignés en moyenne. On peut aussi interpréter notre résultat en disant que la distance moyenne entre y et $y+1$ est bornée par une constante.

Question ouverte

- Trouver une généralisation du protocole *up/down* à des routages par intervalles plus généraux : compacité > 1, facteur d'étirement > 1.

CHAPITRE 2 Routage compact

Le routage est un calcul distribué particulier, et le compromis entre la puissance (ou complexité) du calcul et la taille de la connaissance allouée est un thème central en théorie du calcul distribué. Le routage compact l'illustre donc parfaitement [1]. Rappelons qu'il s'agit étant donné un graphe, de fournir dans chaque sommet un algorithme de routage aussi efficace que possible. La complexité du calcul se mesure par l'efficacité du routage (longueur des routes), et la taille de la connaissance par la taille des structures de données utilisées.

Dans le problème du routage compact, on retrouve bien sûr toutes les composantes rencontrées dans le chapitre précédent. Cependant, de nouvelles composantes vont apparaître.

2.1 Retour sur les définitions

Les résultats de complexité que nous allons donner dépendent fortement du modèle de routage (renumérotation possible des adresses des sommets, taille des adresses, taille des en-têtes, modification des en-têtes autorisée, *etc*). Nous allons définir plus précisément toutes ces notions.

Considérons un graphe G. Chaque sommet u de G possède un nom, un *identifiant* noté $\mathrm{ID}(u)$. Sans être restrictif, on suppose que cet identifiant est un entier. Dans la plupart des cas nous confondrons u et $\mathrm{ID}(u)$. Pour le routage proprement dit, les sommets possèdent une *adresse*, étiquette notée $\mathcal{L}(u)$ supposée être un entier, potentiellement différente de $\mathrm{ID}(u)$.

Une *fonction de routage* R sur G est un algorithme distribué dont le rôle est de délivrer des messages entre les sommets de G. L'algorithme construit un chemin de la source à la destination en choisissant en chaque sommet intermédiaire le prochain lien à suivre. Plus précisément, R est composée d'un couple de fonctions entières (P, H) où P est la *fonction de port* et H la *fonction d'en-tête*. Pour deux sommets distincts [2] u et v, R fourni un chemin

1. Les « Best Student Paper Awards » des éditions 1996 et 2000 du symposium international *Principles of Distributed Computing (PODC)* concernent le routage compact, cf. [GP96b, HP00].

2. On ne considère jamais le routage de u à lui-même dans la mesure où l'on suppose toujours que le processeur hôte connecté à son routeur possède assez d'informations et de puissance de calcul pour

ou *route* $u = u_0, u_1, \ldots, u_r = v$, une suite $h_0, h_1, \ldots, h_r$ d'en-têtes, une suite $p_0, p_1, \ldots, p_r$ de ports de sortie, et une suite $q_0, q_1, \ldots, q_r$ de ports d'entrée. La longueur de la route, notée $\rho_R(u, v)$, est le coût du chemin de u à v si G est pondéré, sinon, $\rho_R(u, v) = r$. Les numéros de ports correspondent à l'identification des liens connectés à un sommet. C'est un nom local, si bien qu'un lien connectant x à y peut avoir un nom différent en x (port de sortie) et en y (port d'entrée). On suppose que les numéros de ports sont des entiers uniques de l'ensemble $\{1, \ldots, d\}$, où d est le nombre de ports correspondant au degré du sommet considéré[3]. Un message avec un en-tête h_i arrivant au sommet u_i sur le port d'entrée q_i se voit attribuer un nouvel en-tête $H(u_i, q_i, h_i) = h_{i+1}$, puis est retransmis par le port de sortie $P(u_i, q_i, h_i) = p_i$. Donc, nous imposons que pour tout $i \in \{0, \ldots, r-1\}$, $H(u_i, q_i, h_i) = h_{i+1}$, $P(u_i, q_i, h_i) = p_i$ et que le lien (u_i, u_{i+1}) a comme numéro de port de sortie p_i en u_i, et q_{i+1} comme port d'entrée en u_{i+1} (voir la figure 2.1). Sur chaque routeur il existe un port spécial d'entrée et de sortie numéroté 0. Il assure la connexion entre le routeur et le processeur hôte associé au sommet u_i. Ceci nous permet de compléter la description en imposant les contraintes $q_0 = p_r = 0$, et $h_0 = \mathcal{L}(v)$, fixant ainsi l'en-tête initial qui est transmis au routeur par le processeur hôte[4] (figure 2.1).

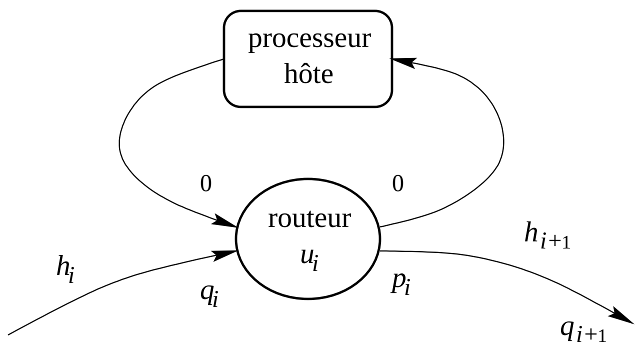


FIG. 2.1 – *Un modèle général de routeur.*

Une *méthode de routage* est un algorithme qui calcule pour un graphe G une fonction de routage R sur G. C'est donc la méthode de routage qui fixe les adresses, les numéros de ports et toutes les structures de données distribuées nécessaires à la fonction de routage. Une fonction, ou plus généralement une méthode, est dite *sommet-préservante* si le nom original des sommets n'est pas changé, c'est-à-dire si $\mathcal{L}(u) = \mathrm{ID}(u)$ pour tout sommet u. Une méthode est dite *universelle* si elle s'applique à tous les graphes de n sommets. Dans

éviter ce genre de communications. De plus, le cas $v = u$ pose des problèmes pour la définition du facteur d'étirement où $d_G(u, v)$ intervient en dénominateur.

3. Dans le cas des graphes orientés, il faut distinguer le degré sortant et entrant.

4. Cette convention qui peut paraître à priori superficielle permet de simplifier la modélisation d'une fonction de routage, et évite de définir comme dans [PU88] $R = (I, P, H)$ par un triplet de fonctions où I (pour fonction d'*initialisation*) est chargé de construire à partir de $\mathcal{L}(v)$ le premier en-tête h_1.

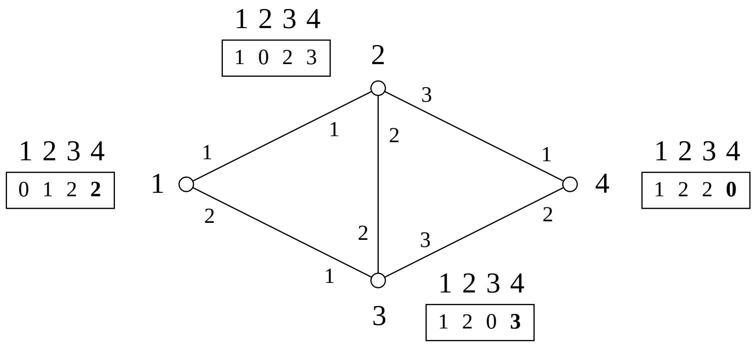
un cadre d'algorithmique distribuée, on note $R_u = (P_u, H_u)$ la restriction de R calculée en u, appelée aussi *fonction locale de routage*. Par commodité, on note $R_u(q,h) = (p,h')$ avec $p = P_u(q,h)$ et $h' = H_u(q,h)$ pour tout port d'entrée q et en-tête h.

Cette modélisation permet de pouvoir comparer précisément toutes les méthodes de routage connues.

2.1.1 Table de routage

Avant d'entrer plus en détails, considérons l'exemple suivant.

Soit $G = (V,E)$ le graphe définit par $V = \{1,2,3,4\}$, et $E = \{12,13,23,24,34\}$. Nous proposons sur G une fonction de routage qui est représentée par la table de la figure 2.2. Au cours du routage les en-têtes représentent la destination et ne varient pas. Le port de sortie (valeur indiquée dans la table), en chaque sommet, ne dépend que de l'adresse de destination. Cette table se lit de la manière suivante : un message de source le sommet 1 et de destination le sommet 4, utilisera le port de sortie indiqué dans la ligne 1 et la colonne 4 de la table. Ainsi, la route d'un message allant du sommet 1 au sommet 4, utilise le port de sortie 2, une fois au sommet 3 utilise le port de sortie 3, puis arrive au sommet 4 par le port d'entrée 2. Donc, dans cet exemple nous avons $R_1(0,4) = (2,4)$, $R_3(1,4) = (3,4)$ et $R_4(2,4) = (0,4)$. Les fonctions locales de routage R_1, R_2, R_3, R_4 (ligne de la table) sont reportées près de chaque sommet. Cette fonction de routage est sommet-préservante car $\mathcal{L}(u) = \mathrm{ID}(u)$ pour tout sommet u.



↱	1	2	3	4
1	0	1	2	**2**
2	1	0	2	3
3	1	2	0	**3**
4	1	2	2	**0**

FIG. 2.2 – *Table de routage.*

Une fonction de routage qui, en chaque sommet, ne dépend que de l'en-tête, et non du port d'entrée, est dite *oblivious* (= oubliante). Et, une fonction de routage *oblivious* qui dépend seulement de la destination, c'est-à-dire telle que $h_0 = h_1 = \cdots = h_r = \mathcal{L}(v)$, est une fonction de *routage direct* ou encore *table de routage*.

Toutes ces distinctions ont un impact sur la mise en œuvre des fonctions de routage. La modification des en-têtes par exemple, peut être coûteuse pour des réseaux de type « tout optique » impliquant des conversions optique-électrique. Les fonctions de routage direct peuvent être codées simplement au moyen d'une table, comme l'exemple précédent. Elles ont aussi la propriété d'être *sans boucle*, c'est-à-dire que la route traversée par un message ne peut pas comporter de cycle, puisque sinon le message bouclerait indéfiniment sans atteindre sa destination, contredisant la définition même d'une fonction de routage (existence d'un chemin entre toutes les paires de sommets).

2.1.2 Paramètres et complexités

La description des paramètres à optimiser va nous permettre de définir proprement les problèmes auxquels nous allons nous intéresser.

Déjà rencontré au chapitre précédent dans l'équation (1.2) page 10, le facteur d'étirement de R sur G est le maximum, sur tous les couples de sommets distincts (u, v), de $\rho_R(u, v)/d_G(u, v)$, alors que la dilatation est la longueur de la route la plus longue, c'est-à-dire $\max_{u \neq v} \rho_R(u, v)$. Une fonction de facteur d'étirement 1 est dite de plus courts chemins.

L'*espace mémoire* d'une fonction de routage R est la taille (en bits) du plus petit programme implémentant R (écrits dans un langage fixé, disons le langage C), comprenant les constantes et les structures de données (*memory requirements*). On distingue l'espace mémoire *local* de R en un sommet u, et l'espace mémoire *global* défini comme la somme de l'espace mémoire local sur tout les sommets du graphes. L'espace mémoire locale de R en u est donc la longueur en bits du plus petit programme[5] C implémentant R_u. L'espace mémoire peut dépendre de la taille des en-têtes et des adresses définies comme la longueur de leur écriture en binaire (exprimée en bits).

La *latence* est le temps maximum de calcul de la fonction (local) de routage R, maximisé sur tous les sommets du graphe. On utilise la complexité en temps dans le pire cas. Comme le temps dépend de la taille des structures de données utilisées, comme nous allons le voir ci-après, il est plus approprié de définir la complexité en temps d'une implémentation donnée de R.

Le problème du routage compact est donc de décrire des méthodes universelles de routage de sorte que les fonctions de routage générées soient les plus efficaces possibles (plus courts chemins ou presque les plus courts, complexité en temps la plus faible possible), avec une mise en œuvre la plus compacte possible (espace mémoire local/global, taille des en-têtes, des adresses).

5. Le choix du langage C est ici arbitraire. En fait l'espace mémoire ainsi défini à partir du C ne varie que par d'une constante additive pour un autre langage. Par exemple, il existe un programme de taille constante permettant de traduire du C en FORTRAN. Le lecteur pourra aussi se référer au « théorème d'invariance » de la complexité de Kolmogorov [LV97].

2.2 Tour d'horizon

2.2.1 Méthodes universelles

Il convient tout d'abord de remarquer que tout graphe possède une fonction de routage de plus courts chemins avec un espace mémoire local de taille[6] $O(n \log d)$, si d est le degré du sommet : ce sont les tables de routage. (Comme dans l'exemple de la figure 2.2, il suffit le lister en chaque sommet le port de sortie pour les $n-1$ destinations possibles). D'autre part, tout routage par intervalles est une fonction de routage direct et inversement. (On peut toujours coder un routage par intervalles par une table de routage en rajoutant le numéro de port de sortie associé à chaque intervalle). En fait, si un routage par intervalles utilise au plus k intervalles par arcs, alors l'espace mémoire local est borné par $O(kd \log n)$ et même $O(kd \log(n/k))$, cf. [Gav00b]. La méthode des tables de routage, ainsi que le routage par intervalles, sont donc des méthodes universelles.

Après cette constatation, une question vient naturellement à l'esprit : existe-t-il des méthodes universelles de routage plus compactes ? Plus compactes que $O(n \log d)$?

Dans [GP96b] nous répondons négativement à cette question en montrant que pour tout entier d, $3 \leqslant d \leqslant n/2$, toute méthode universelle de routage de plus courts chemins nécessite au moins $\Omega(n \log d)$ bits d'information simultanément sur $\Theta(n)$ sommets pour au moins un graphe à n sommets et de degré maximum d, en supposant que les adresses sont optimisées par la méthode et sont choisies dans l'ensemble $\{1, \ldots, n\}$. Le résultat reste valable si les adresses sont de taille $c \log n$ bits, pour toute constante $c \geqslant 1$, et qu'elle que soit la taille des en-têtes. Cette incompressibilité a des implications dans d'autres résultats, comme pour le routage par intervalles où l'on montre dans [GP96a] l'existence[7] de graphe de degré borné et de compacité $\Theta(n)$.

L'espace mémoire peut donc être réduit seulement si l'on accepte de relaxer au moins deux contraintes : les plus courts chemins et la taille des adresses.

2.2.2 Espace mémoire *vs.* facteur d'étirement

Il semble que le premier article traitant de la compression des tables de routage en choisissant des routes approchant les plus courts chemins soit dû à Kleinrock et Kamoun dans [KK77]. L'existence d'un compromis entre l'espace mémoire et le facteur d'étirement sera prouvée plus tard par Peleg et Upfal dans [PU88, PU89]. Pour tout entier k, ils montrent que toute méthode universelle de facteur d'étirement $\Theta(k)$ nécessite un espace mémoire global de $\Omega(n^{1+1/\Theta(k)})$ dans le pire des cas, et en construisent une qui nécessite seulement $O(n^{1+1/\Theta(k)})$ d'espace mémoire global. Le problème général semble donc résolu, mais nous allons préciser ces résultats puisque les constantes cachées dans les notations Θ

6. La fonction $\log n$ représente le logarithme de n en base deux.
7. Malheureusement on ne connaît pas de construction déterministe de tels contre-exemples.

font que ces bornes sont loin d'être optimales.

Bien que la borne inférieure de [PU88] sur l'espace mémoire global soit de $\Omega(n^{1+1/(2s+4)})$ pour un facteur d'étirement s quelconque, ils utilisent pour la borne supérieure une décomposition hiérarchique imposant un facteur d'étirement effectif de $s = 12k + 3$ pour un paramètre $k \geqslant 1$ entier, pour une mémoire globale de $O(k^3\, n^{1+1/k} \log n)$, des adresses de taille $O(\log^2 n)$ et des en-têtes de taille[8] $O(\log n)$. Par exemple, pour obtenir un espace mémoire global strictement meilleur que celui réalisé par les tables de routage, c'est-à-dire meilleur que $O(n^2 \log n)$, il faut choisir $k = 2$ pour avoir $O(n^{1.5} \log n)$ bits de mémoire mais l'étirement est alors $s = 27$. La borne inférieure pour $s = 27$ donne seulement $\Omega(n^{1+1/58})$.

À l'heure actuelle, les meilleures bornes inférieures indiquent que pour $s < 1.4$, l'espace mémoire global doit être au moins de $\Omega(n^2 \log n)$ [GP96b], et, pour $s < 3$, l'espace doit être au moins de $\Omega(n^2)$ [GG97].

Notons aussi que la méthode proposée dans [PU88] n'est pas sommet-préservante, suppose que les coûts sur les arêtes du graphe sont uniformes, et enfin ne permet pas de borner l'espace mémoire local (c'est-à-dire ne permet pas de borner l'espace local par une fonction plus petite que $O(n \log n)$). La méthode d'Awerbuch, Bar-Noy, Linial et Peleg présentée dans [ABNLP89] supprime tous ces défauts avec un espace mémoire local borné par $O(k\, n^{1/k} \log n)$ et un facteur d'étirement en $O(k^2\, 9^k)$. Le facteur d'étirement est réduit à $O(k^2)$ dans [AP90] moyennant un léger accroissement de l'espace mémoire local à $O(k\, n^{1/k} \log^2 n\, \log D)$ où D est le diamètre pondéré du graphe.

Un désavantage majeur de toutes ces méthodes est qu'elles sont plutôt complexes, et donc impraticables pour des réseaux haut-débits pour lesquels le temps de latence dans chaque routeur doit être très court. Ces méthodes sont basées sur des techniques générant des couvertures en régions de faible densité dans le graphe. Ces méthodes fixent $k = O(\log n)$ niveaux de couvertures en régions de rayon croissant. Intuitivement, les niveaux les plus hauts sont responsables du routage des destinations les plus éloignées. Un message est d'abord envoyé sur le niveau le plus bas, dans l'espoir que la destination soit proche. Si ce n'est pas le cas, alors la transmission échoue et le message revient vers l'émetteur. Cette étape est vue comme une « phase » par la fonction de routage. Une fois le message de retour reçu (avec une notification d'échec inscrite dans l'en-tête), l'émetteur essaye d'envoyer le message sur un niveau plus élevé, et ainsi de suite jusqu'à ce que le message soit délivré. Clairement, la mise en œuvre dans chaque sommet de telles procédures résulte d'une fonction de décision complexe, le numéro de la phase courante devant être écrit dans chaque en-tête. Comme la taille des réseaux augmente beaucoup, ce genre d'inconvénients deviennent cruciaux, puisque le principal goulot d'étranglement dans ces réseaux est souvent la décision dans les routeurs et non pas le temps de propagation entre les routeurs. Par conséquent, des fonctions de routage direct, ou disons plus simples à mettre en œuvre en *hardware*, sont préférables en pratique.

8. Le lecteur ne doit pas être surpris par le fait d'avoir des en-têtes de tailles inférieures à celles des adresses étant donné que la source, à la réception de l'adresse de la destination émanant du processeur hôte sur le port 0, a la possibilité de modifier h_0, et de créer le premier en-tête h_1 de taille plus petite.

2.2.3 Routage direct et facteur d'étirement faible

Nous avons vu que les méthodes générales garantissent un espace mémoire réduit, mais pour des valeurs numériques élevées pour le facteur d'étirement (par exemple, $s = 27$ et des fonctions de routage plutôt complexes [PU88]). Pour les petites valeurs du facteur d'étirement, $s \in [2, 5]$, d'autres méthodes ont été proposées.

Dans ce cadre, Cowen dans [Cow99] a construit une méthode générant des fonctions de routage sans boucle pour les graphes pondérés qui nécessitent $O(n^{2/3} \log^{4/3} n)$ d'espace mémoire local pour un facteur d'étirement $s = 3$, et pour des adresses et en-têtes de taille $3 \log n$.

Nous avons proposé dans [EGP98] une méthode, basée sur le routage par intervalles, générant des fonctions de routage direct (donc sans boucle) avec un espace mémoire local borné par $O(\sqrt{n} \log^{3/2} n)$, un facteur d'étirement $s = 5$, un facteur d'étirement moyen $\overline{s} = 3$, et des adresses (et en-têtes) choisies dans l'ensemble $\{1, \ldots, n\}$. De plus la dilatation est bornée par $2D$, voir même $\lceil 1.5D \rceil$ dans le cas de graphes non pondérés. Cette méthode garantit une latence de $O(\log n)$ par sommet. Plus fondamentalement, nous montrons qu'il n'existe aucune méthode de routage générant des fonctions de routage sans boucle avec un espace mémoire local inférieur à $3.7\sqrt{n}$ et des adresses choisies dans l'ensemble $\{1, \ldots, n\}$, et ceci pour n'importe quelle valeur du facteur d'étirement [9]. Il en découle que les compromis espace mémoire/facteur d'étirement présentés précédemment (par exemple la méthode sommet-préservante de [AP90]) ne peuvent prétendre générer des fonctions de routage sans boucle, donc nécessitent de pouvoir changer les en-têtes.

Question ouverte

- Quel est le meilleur compromis possible entre l'espace mémoire local et le facteur d'étirement (ou facteur d'étirement moyen) pour les fonctions de routage direct utilisant des adresses choisies parmi l'ensemble $\{1, \ldots, m\}$ avec $m > n$?

2.2.4 Presque tous les graphes

Nous avons vu que dans le pire des cas, une fonction de routage de plus courts chemins nécessite $\Theta(n \log n)$ bits d'espace de mémoire local (cf. le résultat de [GP96b] en prenant $d = \Theta(n)$). Nous allons voir qu'en « moyenne » l'espace mémoire local est seulement de $O(n)$, et que de manière assez surprenante, les réseaux nécessitant un grand espace mémoire pour le routage de plus courts chemins, ne sont pas ceux qui possèdent la plus grande quantité d'informations [10].

9. Dans le paragraphe 2.3.3 page 28 nous donnons une idée de la preuve.

10. Cette famille de graphes est vague, mais elle peut être vue comme les graphes dont la matrice d'adjacence est incompressible au sens de la complexité de Kolmogorov.

Certains résultats en théorie des graphes sont valables pour « presque tous les graphes ». L'expression « presque tous » doit être prise au sens statistique du terme. On indique par là que la fraction des graphes à n sommets pour laquelle la propriété est satisfaite tend vers 1 lorsque n tend vers l'infini. Les outils pour établir ces résultats sont les probabilités, avec les graphes aléatoires et la famille[11] $\mathcal{G}_{n,p}$, ou encore la complexité de Kolmogorov [LV97] avec les graphes aléatoires de Kolmogorov. Précisons tout de suite que le deuxième outil varie avec le premier essentiellement dans la forme, cf. [BLV98].

Dans le cadre du routage par intervalles, Flammini, van Leeuwen et Marchetti-Spaccamela avaient déjà dans [FvLMS95] étudié la compacité (c'est-à-dire le nombre d'intervalles par arcs pour un routage de plus courts chemins, cf. le chapitre précédent) des graphes $\mathcal{G}_{n,p}$, pour des valeurs particulières de p. Plus généralement, Buhrman, Hoepman et Vitányi dans [BHV99] ont montré qu'une fraction d'au moins $1 - 1/n^3$ de tous les graphes possèdent des tables de routage de plus courts chemins de taille $3n+o(n)$ bits, sous l'hypothèse que les adresses des sommets sont choisies dans l'ensemble $\{1,\ldots,n\}$ et permutées aléatoirement, et que les numéros de ports peuvent être renumérotés. Si les adresses des sommets peuvent être renumérotées par un mot binaire de $c\log^2 n$ bits, alors les tables de routage de plus courts chemins peuvent être réduites à $c\log^2 n$ bits par sommets, où c est une constante. D'autres résultats pour des facteurs d'étirements $s > 1$ sont mentionnés.

Finalement, un corollaire de notre résultat [GP98] sur le routage par intervalles (cf. page 6) est qu'une fraction d'au moins $1 - 1/n$ de tous les graphes possèdent des tables de routage de plus courts chemins de taille $n + O(\log^4 n)$ bits par sommet, avec des adresses de sommets choisies parmi l'ensemble $\{1,\ldots,n\}$, améliorant le résultat de [BHV99].

Question ouverte

- La borne $n + o(n)$ est-elle la meilleure possible pour une fraction de $1 - o(1)$ de tous les graphes ?

L'établissement d'une borne inférieure n'est pas facile. Par exemple, on pourrait croire qu'un sommet doit au moins "connaître" ses voisins pour réaliser un routage de plus courts chemins, donc doit connaître $\log\binom{n}{d} \sim n$ bits si $d \approx n/2$ (sélection aléatoire de d voisins parmi n). Mais, à cause de la renumérotation possible des sommets et des numéros de ports, c'est totalement faux, en particulier pour les arbres où $O(\sqrt{n})$ bits sont suffisants, même pour $d = \omega(\sqrt{n})$ (voir le paragraphe 2.3.3 page 28). Notons aussi que, techniquement parlant, lorsque les sommets d'un graphe aléatoire $G \in \mathcal{G}_{n,1/2}$ ont été renumérotés, alors pour tout $x, y \in \{1,\ldots,n\}$ la probabilité d'avoir une arête entre les sommets numérotés x et y n'est plus $1/2$. Elle dépend de l'algorithme de renumérotation utilisé ; le graphe n'est plus totalement aléatoire.

11. Ce modèle définit la probabilité p d'avoir une arête entre deux sommets d'un graphe à n sommets, cf. [Bol75].

2.2.5 Graphes planaires et séparateurs

D'autres résultats sur le routage compact s'appliquent à des familles de graphes particulières.

Frederickson et Janardan ont proposé dans [FJ90] des méthodes de routage applicables à toute famille de graphes (connexes) qui peuvent être décomposés récursivement par un séparateur de cardinalité au plus c. Un séparateur est un ensemble de sommets dont la suppression déconnecte un graphe (connexe) en deux (ou plus) composantes connexes, de taille au plus $2/3$ la taille initiale du graphe. Plus généralement, le graphe est dit c-décomposable si pour toute pondération des sommets du graphe il existe un séparateur de taille c qui produit des composantes connexes de poids (qui est la somme des poids sur les sommets) au plus $2/3$ du poids initial. Cette définition implique une décomposition récursive du graphe en au plus $\log_{3/2}(n)$ niveaux hiérarchiques, cf. [FJ90]. Un séparateur assure que tout chemin entre deux sommets de composantes différentes doit passer par le séparateur, ce qui permet de concentrer l'information de routage sur les sommets du séparateur si celui-ci est de petite taille. Les graphes planaires-extérieurs et plus généralement les graphes séries-parallèles sont 2-décomposables, les graphes de largeur arborescente k sont $O(k)$-décomposables, et les graphes planaires sont $O(\sqrt{n})$-décomposables.

Plus précisément, ils ont proposé deux méthodes de routage. La première, applicable aux graphes pondérés, utilise un espace mémoire global de $O(c\,n\log^2 n)$ bits, un facteur d'étirement $s = 3$, et des adresses de taille $O(\log n)$. La seconde méthode, avec le même espace mémoire global, permet de réduire le facteur d'étirement à $s = 1+2/\alpha$, où $1 < \alpha \leqslant 2$ (donc $s < 3$) est la racine de l'équation $\alpha^{\lceil (c+1)/2 \rceil} - \alpha = 2$, pour un accroissement des adresses qui passent à une taille de $3.42\,c\log c\,\log n$ bits. Pour $c \in \{2,3\}$, $\alpha = 2$, et pour $c \in \{4,5\}$, $\alpha \leqslant 2.32$. Les en-têtes sont supposés modifiables par la fonction de routage, et le routage n'est pas forcément sans boucle. De plus l'espace mémoire local n'est pas borné.

Ces deux méthodes sont efficaces que si c est constant. Pour les graphes planaires, échappant à cette contrainte, $c = \Theta(\sqrt{n})$, ils proposent de meilleures méthodes [FJ89] (valables également pour les graphes planaires pondérés). La première garantit un espace mémoire global $O(n^{4/3}\log n)$ et un facteur d'étirement $s = 3$, pour des adresses et en-têtes de taille $O(\log n)$. La seconde, pour tout ϵ, $0 < \epsilon < 1/3$, produit un espace mémoire global de $O((1/\epsilon)n^{1+\epsilon}\log n)$ pour des adresses de taille $O((1/\epsilon)\log n)$, et un facteur d'étirement $s = 7$. Ces deux méthodes souffrent du même défaut que les précédentes : elles ne permettent pas de borner l'espace mémoire local (qui peut atteindre $O(n\log n)$ dans le pire des cas), et utilisent des en-têtes modifiables et des adresses de taille strictement plus grande que $\log n$.

2.2.6 Graphes en k pages et graphes de genre borné

Comme nous venons de le voir, la complexité de l'espace mémoire local des fonctions de routage de plus courts chemins dans les graphes planaires est un problème ouvert.

Nous avons abordé cette question dans l'article [GH99]. L'idée est qu'un graphe planaire, ou plus généralement un graphe de genre borné, possède de fortes propriétés géométriques. Par exemple, un graphe planaire dessiné sur une sphère possède un système de coordonnées longitude/latitude qui doit pouvoir être utilisé pour le routage. Comme nous allons le voir cette intuition se trouve vérifiée, mais notre résultat ne découle malheureusement pas d'un système de coordonnées sphériques généralisé. En effet, la planarité est une propriété difficilement utilisable mathématiquement, en particulier pour définir des métriques, ce qui explique peut-être la difficulté à résoudre les problèmes de routage compact et d'ensembles r-dominants sur les graphes planaires (cf. le paragraphe 1.4.3 page 8).

Il est facile de montrer que les graphes planaires possèdent des fonctions de routage de plus courts chemins d'espace mémoire local $O(n)$. Cela découle d'un argument simple de comptage. En effet, le nombre de graphes planaires non isomorphes est $2^{cn+o(n)}$ où $c \in [3.24, 6.24]$. Donc, $6.24n$ bits sont suffisants pour que chaque routeur ait une description complète du graphe, et donc sache trouver un plus court chemin. Il suffit d'indiquer au routeur l'indexe du graphe dans une énumération fixée de l'ensemble des graphes planaires, ce qui est calculable bien que très coûteux algorithmiquement. Cependant, même si le codage du graphe, par cet indexe, est optimal, c'est-à-dire de taille $cn + o(n)$ bits, et permet de tester l'adjacence en temps $O(1)$, il reste à calculer un algorithme de plus court chemin pour trouver le port de sortie. Dans tous les cas, une telle approche nécessite une latence de $\Omega(n)$ par sommet, temps de calcul qui doit se dérouler sur chaque sommet de la route ! Notez que ce temps représente, pour une adresse de taille $O(\log n)$, une complexité exponentielle en la taille de l'entrée pour la fonction locale de routage.

Notre résultat, valable pour les graphes planaires pondérés, utilise des tables de routage de plus courts chemins de taille $8n + o(n)$ telle que la latence dans chaque routeur est $O(\log^{1+\epsilon} n)$, pour toute constante $\epsilon > 0$. Pour prouver cela nous utilisons la représentation en k pages des graphes. Cette représentation consiste à ordonner les sommets sur un cercle de sorte que les arêtes qui sont des cordes puissent être coloriées en k couleurs et que deux arêtes de la même couleur ne se coupent pas. Les graphes planaires peuvent être représentés en au plus 4 pages [Yan86].

Les adresses des sommets sont fixées par la représentation en k pages du graphe. Les sommets sont numérotés consécutivement autour du cercle par les entiers de 1 à n. Chaque routeur u code un arbre couvrant de poids minimum enraciné en u, noté T_u, qu'il utilisera de la façon suivante pour router un message : on effectue un parcours de T_u depuis le sommet de destination v jusqu'à l'un des voisins de u. Ce voisin permettra de déterminer le port de sortie de u pour router vers v. On code T_u (dont les arêtes sont coloriées) par un système de parenthèses car les arêtes d'une couleur donnée ne se coupent pas. La taille de la structure de données ne dépasse pas $O(n)$ bits pour les graphes planaires, et $O(n \log k)$ dans le cas général des graphes en k pages.

Cependant, la structure de données doit être raffinée car le parcours dans T_u de v à u nécessite de trouver le père de v, son grand-père et ainsi de suite (chaque calcul d'ancêtre correspondant à trouver une parenthèse ouvrante ou fermente correspondante). De plus, le nombre de sommets de l'arbre à traverser peut être en ordre de n (proportionnel au nombre minimum d'arêtes d'un chemin reliant u à v). Pour obtenir un temps de latence polylogarithmique, nous utilisons la structure de données de [MR97] sur $o(n)$ bits qui permet de répondre en temps constant à des requêtes du type parenthèse ouvrante ou fermente correspondante. Enfin, pour réduire la longueur du parcours dans l'arbre, nous utilisons des tables représentant un ensemble r-dominant de T_u, noté X, et l'on fixe $r = \log^{1+\epsilon} n$, avec $\epsilon > 0$ une constante arbitraire. Ainsi, après un parcours de r sommets au plus depuis v, on rencontre nécessairement un sommet appartenant à X. Il suffit de coder in extenso le port de sortie pour tous les sommets de X et de stopper le parcours de l'arbre. Donc la latence pour u est borné par $O(\log^{1+\epsilon} n)$. Il est connu que l'on peut choisir X tel que $|X| \leqslant n/r$. Donc la taille de la table générée (celle codant in extenso tous les ports de sorties pour les sommets de X) est $O(|X| \cdot \log n) = o(n)$ bits.

Ainsi notre méthode produit des tables de routage d'espace mémoire local $O(n \log k)$ pour les graphes en k pages avec une latence $O(\log^{1+\epsilon} n)$. Par le résultat de Malitz [Mal88] qui montre que les graphes de genre g sont représentables sur $O(\sqrt{g})$ pages, nous étendons cette borne à $O(n \log g)$ bits dans le cas général. Les résultats restent valables même si les graphes sont "presque" de genre g, c'est-à-dire représentables sur une surface de genre g avec $o(n/\log n)$ croisements.

Questions ouvertes

- Quelle est la complexité exacte de l'espace mémoire local pour les tables de routage de plus courts chemins dans les graphes planaires?

La borne inférieure est de $\Omega(\sqrt{n})$ (cf. ci-dessous le paragraphe 2.3.3 concernant les arbres) et la borne supérieure est de $O(n)$.

- Plus généralement, quelle est la complexité exacte de l'espace mémoire local pour les tables de routage de plus courts chemins dans les graphes de genre g?

En effet, il n'est pas certain que la borne $O(n \log g)$ soit la meilleure possible. Une dépendance plus faible en g est possible étant donnée qu'aucune borne inférieure n'existe.

2.3 Quelques problèmes clés

Compresser c'est comprendre. Représenter des données par une structure de données compactes, c'est finalement déterminer un modèle de ces données. Ainsi, le routage compact peut se réduire (en première approximation) à la connaissance de la structure des plus courts chemins dans un graphe.

Nous donnons ci-dessous quelques problèmes clés liés au routage compact, et aux structures de données compactes.

2.3.1 Comment trouver l'intervalle ?

Supposons qu'un graphe possède une fonction de routage par intervalles R avec un intervalle linéaire par arc. La question est de savoir, quelle structure de données utiliser pour R de façon à minimiser la latence (le temps de calcul local de la fonction de routage). D'un point de vu pratique, cette question est évidemment importante, et nous allons voir que la réponse n'est pas si simple.

Considérons un sommet x de degré d et notons $[a_i, b_i]$ l'intervalle associé à l'arc (x, y_i), $i \in \{1, \ldots, d\}$, numéroté tel que $a_1 < a_2 < \cdots < a_d$. On souhaite donc calculer le plus rapidement possible, pour tout $y \in \{1, \ldots, n\}$, l'indice $i = R_x(y)$ tel que $y \in [a_i, b_i]$.

Une première solution consiste simplement à stocker les a_i dans une table T telle que $T[i] = a_i$. Le calcule de $R_x(y)$ consiste à chercher i tel que $a_i \leqslant y < a_{i+1}$, car les $[a_i, b_i]$ forment une partition de $[1, n]$. La latence de R_x est donc au plus $O(\log d)$ en utilisant une recherche dichotomique de y dans T. L'espace mémoire de R_x est $O(d \log n)$ bits. Cependant on peut faire plus compact et plus rapide (et nous allons voir comment) si $d > n/\log n$.

La deuxième méthode garantit un espace de $n+o(n)$ et une latence en $O(1)$. Elle consiste cette fois à représenter l'ensemble des a_i par une chaîne binaire, B, telle que $B[\alpha] = 1$ si $\alpha \in \{a_1, \ldots, a_d\}$ et $B[\alpha] = 0$ sinon. Il découle que $R_x(y) = i$ si le nombre de a_i inférieurs ou égaux à y est exactement i. C'est aussi le nombre de 1 dans B depuis le premier bit jusqu'à la position y incluse. L'espace mémoire est n (la chaîne B) et le temps est à priori $O(y)$, le temps de parcourir B et de compter le nombre de 1 jusqu'à la position y. Munro et Raman ont montré dans [MR97, Mun96] que ce problème précis pouvait être résolu en temps[12] $O(1)$ dans le pire des cas, grâce à une structure de données de taille $n + o(n)$ bits constructible en temps linéaire.

La solution idéale serait bien évidemment de représenter l'ensemble $\{a_1, \ldots, a_d\}$ par une structure de données de taille[13] $\log \binom{n}{d} = \Theta(d \log(n/d))$ pour une latence $O(1)$. Bien que Brodnik et Munro dans [BM99] présentent un codage quasi-optimal en $\log \binom{n}{d}$ bits des ensembles à d éléments choisis parmi $\{1, \ldots, n\}$ avec un temps constant pour le test de l'appartenance, la question du calcul du rang d'un élément de l'ensemble en temps constant et avec un nombre de bits optimal reste ouverte.

Question ouverte

- Peut-on trouver une structure de données garantissant $O(d \log(n/d))$ d'espace mémoire par sommet de degré d, et un temps de latence constant pour les graphes de compacité 1 ?

12. Le modèle de calcul est le modèle *word-RAM* où les instructions arithmétiques standards sur des entiers de taille $O(\log n)$ bits sont exécutées en temps constant.

13. Ceci serait un codage optimal puisqu'il y a $\binom{n}{d}$ ensembles possibles.

2.3.2 Latence cumulée

Nous venons de voir qu'il est difficile de produire une structure de données compactes pour une latence minimale, même dans le cas simple du routage par intervalles. Une alternative possible serait de considérer la latence *cumulée* sur une route de longueur L.

Le cas du routage de plus courts chemins dans le graphe de de Bruijn [PS85] est révélateur. Il consiste à calculer le plus grand préfixe de l'adresse de destination qui est un suffixe de l'adresse de la source[14]. Ce préfixe constitue le premier en-tête. À chaque sommet, le premier bit de l'en-tête courant est extrait : si c'est 0 le message est transmis vers le port 1, si c'est 1 il est transmis vers le port 2 (ce graphe est orienté et n'a que deux arcs sortants). Dans tous les cas, ce bit est détruit de l'en-tête après utilisation. Le message arrive à destination lorsque l'en-tête est vide. Donc la latence cumulée sur une route de longueur L est $O(L+\log n)$, le terme $\log n$ résultant du calcul du premier en-tête par l'algorithme de Boyer-Moore [Aho90]. La latence est constante, excepté pour le sommet source. Comme $L \leqslant \log n$, la latence cumulée ne dépasse jamais $O(\log n)$.

Le problème de trouver une structure de données compactes pour optimiser la latence cumulée a été soulevé par Frederickson dans [Fre93]. Pour les graphes planaires-extérieurs pondérés (de compacité 1), il donne une structure de données de taille $O(d \log n)$, basée sur les intervalles mais avec des tables auxiliaires, de sorte que la latence cumulée n'excède pas $O(L+\log n)$. (L représente le nombre de routeurs traversés.)

Exprimée dans une complexité légèrement différente (complexité en bit-opérations), nous avons présenté dans [FG98b] une méthode de routage garantissant une latence de $O(\log n)$ bit-opérations par sommet et une latence cumulée de $O(D + n^{1/k} \log n)$ bit-opérations, où D est le diamètre du graphe et $k \geqslant 2$ une constante arbitraire. Cette méthode est sommet-préservante et utilise des en-têtes de taille $O(\log n)$. Cependant, cette méthode ne garantit pas un routage selon les plus courts chemins, et l'espace mémoire local peut être $O(n \log n)$ comme les tables de routage. Ce résultat doit être bien sûr comparé à la méthode des tables de routage, qui, dans ce modèle, génèrent une latence de $O(\log n)$ bit-opérations par sommet et une latence cumulée de $O(D \log n)$ bit-opérations dans le pire des cas. En effet, dans ce modèle, la lecture d'un bit de l'en-tête coûte un temps 1. Et la lecture d'une donnée sur ℓ bits dans une table de taille t coûte $O(\ell + \log t)$, donc $O(\log n)$ bit-opérations pour la lecture d'un numéro de port dans une table de routage. Notre approche dans [FG98b] consiste donc à éviter le plus possible d'utiliser des tables de routage trop grandes, et de prendre le moins possible des décisions de routage. Autant que possible, nous utilisons un routage sur $k = O(1)$ niveaux : la décision se prend uniquement en fonction du niveau courant d'un message, l'en-tête comprenant le niveau et l'adresse de destination. Si le niveau vaut $i \in \{1, \dots, k\}$ le message est routé vers le port de sortie p_i, dépendant que de i, que l'on a numéroté pour être dans l'ensemble $\{1, \dots, k\}$. Cette décision ne coûtera que $O(\log i + \log k)$ bit-opérations, ce qui est constant puisque $k = O(1)$.

14. Les sommet de ce graphe sont les $n = 2^k$ mots binaires de longueur k. Le sommet $x_k x_{k-1} \dots x_2 x_1$ est relié aux deux sommets $x_{k-1} \dots x_2 x_1 \alpha$, $\alpha \in \{0, 1\}$.

(Notre modèle suppose que l'en-tête et le corps du message peuvent être copiés et transmis sur le port de sortie en temps constant, cf. [FG98b].)

Question ouverte

- Existe-t-il une méthode de routage garantissant une latence cumulée maximum de $O(D + \log n)$ bit-opérations ?

2.3.3 Routage dans un arbre

Nous l'avons vu au chapitre précédent : le routage dans les arbres, et donc le routage par intervalles qui lui est associé, est un problème fondamental pour toutes les applications qui en découlent (cf. le paragraphe 1.3). On aurait envie d'ajouter : « ... et ce problème est bien compris ». En est-on si sûr ? On peut dire, en effet, que le routage par intervalles garantit pour tout arbre un espace mémoire global $O(n \log n)$ ce qui est très faible, $O(\log n)$ par sommet en moyenne. De plus il est très simple à mettre en œuvre, la latence est $O(\log n)$. Il y a toutefois au moins deux critiques :

1. L'espace mémoire local n'est pas borné par $O(\log n)$ lorsque le degré est élevé.
2. On pourrait espérer une latence $O(1)$ avec une structure de données adéquate.

Dans le paragraphe 2.3.1 nous avons vu que dans le cas d'un seul intervalle par arc, on pouvait espérer un espace mémoire local $O(d \log(n/d))$, et qu'avec un espace de $n + o(n)$ bits on pouvait garantir une latence $O(1)$. L'espace $O(d \log(n/d))$ est-il une borne optimale pour les arbres ?

La réponse est non, on peut faire mieux. Dans l'article [Gav00b], nous montrons que lorsque $d > \sqrt{n}/\log n$ il est possible d'utiliser une structure de données de taille $O(\sqrt{n})$ au plus. Et dans le cas général, nous montrons qu'il n'est pas possible de faire mieux !

Explicitons la borne inférieure. Prenons une méthode de routage sur les arbres à n sommets qui produise des fonctions de routage direct (donc sans boucle et forcément de plus courts chemins dans le cas des arbres) avec des adresses optimisées dans l'ensemble $\{1, \ldots, n\}$ (tables de routage). Fixons un arbre T, un sommet x de T, et notons R_x la fonction de routage de x. Sachant n et l'adresse de x, il est possible de tester R_x sur toutes les adresses distinctes de x et de compter combien de destinations utilisent chaque port de sortie. Sans autre hypothèse, nous obtenons une séquence $S = (n_1, \ldots, n_d)$ d'entiers où n_i est le poids du sous-arbre enraciné au i-ème fils de x, c'est-à-dire le nombre de descendants du i-ème fils. En effet, tous doivent être router vers le même port. En ordonnant les entiers n_i de manière croissante, nous avons $1 \leqslant n_1 \leqslant \cdots \leqslant n_d$ et $\sum_{i=1}^{d} n_i = n - 1$. C'est la définition d'une partition de l'entier $n - 1$ en d parts. Notons que pour chaque partition P de $n - 1$ il existe un arbre et avec un sommet où ses fils ont un poids valant exactement chacune des parts de P (par exemple un arbre de hauteur deux). On déduit qu'il existe un arbre et un sommet z où l'espace mémoire local est au moins de $\log(p_{n-1}) - O(\log n)$ bits, p_{n-1} représentant le nombre de partitions de $n-1$ et le terme $O(\log n)$ représentant la quantité d'informations maximale contenue dans n et l'adresse du sommet z. On conclut par le fait que[15]

$$p_n \sim 2^{c\sqrt{n}+o(\sqrt{n})} \quad (c \approx 3.7, \text{ formule de Ramanujan}).$$

La borne supérieure découle de la preuve précédente. En effet, il suffit de numéroter les sommets par un parcours en profondeur-d'abord modifié de sorte que les sommets de poids le plus fort soient numérotés en priorité (le poids d'un sommet étant le nombre de ses descendants). L'arbre de la figure 1.1 page 3 montre une telle numérotation des sommets. On numérote les ports de sortie d'un sommet x suivant le poids décroissant de ses fils (le port connectant au père, s'il existe, est fixé au degré de x). Outre l'adresse de x, il suffit de coder la séquence des poids des d fils[16] de x, $S = (n_1, \ldots, n_d)$, pour calculer la fonction de routage qui est

$$R_x(y) = \begin{cases} i & \text{si} \quad y \in \left] x + \sum_{j=1}^{i-1} n_j,\ x + \sum_{j=1}^{i} n_j \right] \\ \\ d+1 & \text{si} \quad y < x \ \text{ ou } \ y > x + \sum_{j=1}^{d} n_j \end{cases} \tag{2.1}$$

Cette fonction est calculable connaissant S, x et d. Le codage de S peut théoriquement être réalisé en codant l'indexe de S sur $c\sqrt{n}$ bits dans une énumération fixée de l'ensemble des partitions de $n-1$.

Bien que ceci représente un codage optimal en taille, la question de la latence n'est pas abordée. (Par exemple, comment extraire de S un n_j en un temps raisonnable, si S est compressée sur $O(\sqrt{n})$ bits?)

Questions ouvertes

• Trouver une structure de données de taille $O(\sqrt{n})$ permettant de coder S efficacement et de répondre en temps $O(1)$ à l'équation (2.1).

• Plus généralement, on peut aussi se demander si une méthode utilisant des adresses un peu plus grandes, disons sur $c \log n$ bits avec $c = O(1)$, peut diminuer significativement l'espace mémoire local.

Notons que la preuve de la borne inférieure présentée ci-dessus ne s'applique pas si les adresses des destinations sont sur $\log(2n)$ bits.

15. Pour être plus précis, $c = \left(\pi\sqrt{2/3}\right) / \ln 2 = 3.7006565593...$

16. Donc ici d vaut soit le degré de x si x est racine, soit le degré moins 1.

2.3.4 Routage compact et latence exponentielle

En fait, dans [FG98b] nous avons abordé la question du temps brutalement en s'interrogeant ainsi : peut-il se produire qu'une fonction de routage soit compacte mais inutilisable à cause de sa latence exponentielle ?

La réponse est oui. Dans une généralisation du résultat présenté dans [FG98b] nous montrons qu'il existe des graphes à n sommets possédant des fonctions de routage de plus courts chemins d'espace mémoire locale $O(\log n)$, mais pour lesquels toute fonction de routage de plus courts chemins implémentée avec moins de $cn \log n$ bits d'espace mémoire locale, pour une constante $c > 0$, et des adresses de taille $O(\log n)$, a une latence supérieure à n'importe quelle pile d'exponentielles de taille constante, c'est-à-dire :

$$2^{2^{\cdot^{\cdot^{\cdot^{2^n}}}}}$$

La preuve est basée en partie sur un résultat classique de la complexité de Kolmogorov [LV97], et à ce titre ne montre que l'existence de tels graphes. Il serait souhaitable de savoir si ce genre de résultat peut se produire pour des familles plus concrètes.

2.3.5 Routage dans les graphes de Cayley

Les familles de graphes à forte structure sont de très bons candidats pour le routage compact, puisqu'à priori peu de mémoire est nécessaire pour coder la structure du graphe. Seulement, la structure du graphe codée telle quelle peut ne pas donner de moyens efficaces pour le problème de la recherche d'un plus court chemin.

Les graphes de Cayley font partie de cette catégorie. Les sommets sont les éléments d'un groupe Γ et les arcs[17] sont définies par la donnée d'un ensemble générateur $S \subset \Gamma$: x est relié à y s'il existe un élément de $s \in S$ permettant de passer d'un sommet à l'autre par l'opération du groupe, c'est-à-dire $y = x \cdot s$ en notation multiplicative.

Pour fixer les idées, prenons $\Gamma = \mathbb{Z}_n$ (le groupe additif modulo n) et $S = \{\pm 1, \pm c_1, \pm c_2\}$ avec $c_1, c_2 \in \mathbb{Z}_n \setminus \{0, 1\}$ (on rend symétrique le graphe en prenant des générateurs opposés). Le graphe de Cayley induit par (Γ, S) peut être décrit simplement en codant les entiers c_1 et c_2. Donc le problème du routage compact peut être résolu avec seulement $O(\log n)$ bits d'espace mémoire local (en effet, comme vu précédemment, il est toujours possible de construire le graphe et d'appliquer un algorithme en largeur d'abord pour trouver le port de sortie). Le temps de latence d'une telle solution est $\Omega(n)$ par sommet, alors qu'on pourrait espérer un temps poly-logarithmique, c'est-à-dire linéaire en la taille de la donnée pour la fonction de routage (l'entrée étant l'adresse de la destination). Ce problème peut revenir à la décomposition minimale d'un élément en somme de générateurs. Malheureusement c'est

17. Les graphes de Cayley sont définis comme orientés. Cependant si $s^{-1} \in S$ pour tout $s \in S$, on peut les considérer comme non orientés.

un problème reconnu difficile. En fait, pour $S = \{\pm c_1, \pm c_2\}$ le problème peut être résolu en temps $\log^{O(1)} n$, mais il reste ouvert pour des ensembles S de cardinalité supérieure.

Question ouverte

• Quelle est la latence des fonctions de routage de plus courts chemins sur les graphes de Cayley de degrés k définis sur un groupe Abélien, si l'espace mémoire local alloué est $O(k \log n)$?

L'espace alloué est suffisant pour pouvoir stocker entièrement la description du graphe, puisqu'il suffit de décrire le groupe et l'ensemble générateur : il y a $O(k)$ éléments dans S, chacun pouvant être représenté par un entier de $\{1, \ldots, n\}$ ($O(k \log n)$ bits), et, pour chaque n il y a au plus $n^{O(1)}$ groupes Abéliens non isomorphes, donc chacun représentable sur $O(\log n)$ bits. (En effet, un groupe Abélien Γ est isomorphe à un produit de groupes cycliques, $\Gamma \simeq \prod_{i=1}^{r} \mathbb{Z}_{n_i}$ avec $n = \prod_{i=1}^{r} n_i$, ce qui implique un nombre au plus polynômial de suites non ordonnées $(n_1, \ldots, n_r)$ possibles.)

CHAPITRE 3 Étiquetage de distance

La plupart des représentations standards de réseaux sont globales par nature. Elles imposent aux utilisateurs d'avoir un accès sur la structure du graphe sous-jacent dans l'ensemble, pour extraire de l'information, même si la partie de l'information visée est très locale et concerne seulement quelques sommets.

3.1 Introduction

La notion de schéma d'*étiquetage d'adjacence*, introduite par Breuer et Folkman [Bre66, BF67], suggère une représentation des graphes plus localisée qu'elle ne l'est habituellement. L'idée est d'étiqueter les sommets de façon que l'on puisse tester l'adjacence entre sommets à partir de leur étiquettes, sans utiliser aucune autre source d'information. Spécifiquement, ils montrent dans [BF67] qu'il est possible d'étiqueter tout graphe à n sommets et de degré maximum Δ par des mots binaires de $2n\Delta$ bits, de sorte que deux sommets sont adjacents si et seulement si leurs étiquettes sont à une distance de Hamming[1] au plus $4\Delta - 4$.

Bien évidemment, des étiquettes de taille illimitée peuvent servir à coder toute l'information désirée (par exemple on pourrait coder dans l'étiquette du sommet i la i-ème ligne de la matrice d'adjacence du graphe). Cependant, pour que le schéma d'étiquetage soit intéressant, il faut se limiter à des étiquettes de taille relativement courte (disons, de taille poly-logarithmique en la taille du graphe), et à une certaine tolérance dans l'efficacité pour extraire l'information des étiquettes (disons, également en temps poly-logarithmique). La faisabilité de tels schémas d'étiquetage efficaces a été investie il y a plus d'une décade par Kannan, Naor et Rudich [KNR88]. En particulier, ils ont donné un schéma très élégant pour les arbres avec des étiquettes de taille $2\log n$ (bits). Ce schéma s'étend naturellement, avec des étiquettes de taille $O(\log n)$, à de nombreuses familles de graphes, telles que les graphes d'arboricité bornée (incluant les graphes de degré ou de genre borné, comme les graphes planaires par exemple), divers graphes d'intersections (incluant les graphes d'intervalles), et les graphes c-décomposables ou de largeur arborescente bornée (cf. le pa-

1. La distance de Hamming entre deux mots binaires de même longueur représente le nombre de bits dont ils diffèrent.

ragraphe 2.2.5 page 23). Leur conjecture sur la *représentation implicite*, qui semble être tombée en désuétude, énonce que toute famille héréditaire de graphes à n sommets et qui comporte $2^{O(n \log n)}$ membres[2], possède un schéma d'étiquetage d'adjacence avec des étiquettes de taille $O(\log n)$.

Cette idée naturelle a été ravivée récemment par l'observation que l'on peut non seulement calculer l'adjacence, mais encore qu'il serait possible de capturer une information de distance par des schémas semblables. Cela mène à la notion d'*étiquetage de distance*: schémas d'étiquetage des sommets des graphes ayant la capacité de déterminer efficacement la distance entre deux sommets en connaissant leurs étiquettes seulement.

3.2 Applications

Nous donnons deux applications de l'étiquetage de distance. On pourrait en rajouter d'autres. Par exemple, la compression des matrices de distances pour le calcul intensif. En particulier, en géométrie algorithmique, dans le contexte des graphes définis sur des espaces topologiques spécifiques et réguliers (par exemple, les graphes Euclidiens de faibles dimensions).

3.2.1 Établissement d'une connexion

Revenons un instant sur le problème du routage. Trouver un algorithme ou des tables de routage nécessite un pré-calcul sur tout le réseau, effectué de manière distribuée ou non. Dans de nombreux cas concrets, cette liberté n'est pas donnée. Par exemple, parce que le réseau est de taille trop importante, ou que son évolution est dynamique et donc imprévisible.

Pour établir une connexion entre les sommets x et y, en l'absence de table de routage, on peut donc être amené à construire de manière dynamique un chemin, puis, une fois établi et réservé, à l'utiliser de manière intensive. On souhaite de plus que ce chemin soit le plus court possible pour des raisons d'efficacité (pour l'utilisateur) et d'optimisation de la bande passante (pour le gestionnaire du réseau), quitte à perdre un peu de temps pour l'établissement de ce chemin.

Une méthode standard pour trouver un chemin entre x et y sans table de routage est de diffuser un message depuis x vers l'ensemble des sommets du réseau afin d'atteindre y le plus rapidement possible, puis qu'y détermine cette connexion en remontant jusqu'à x par un des chemins possibles par l'intermédiaire d'accusé de réception. Pour le gestionnaire, il y a un compromis à réaliser entre d'une part, le temps et la bande passante utilisés pour l'établissement de cette connexion, et d'autre part le gain réalisé en offrant le chemin le

2. Ici un membre est un graphe étiqueté de la famille dont l'ensemble des sommets est $V = \{1, \ldots, n\}$. Ainsi, la famille des arbres à n sommets possèdent n^{n-2} membres.

plus court. En utilisant cette méthode, il est clair que l'établissement de la connexion va coûter $\Theta(m)$ messages, où m est le nombre de d'arêtes du graphe. En effet, la diffusion depuis x ne peut pas être arrêtée une fois qu'y est atteint. Des messages vont parcourir tous les liens du réseau, et ceci même si y est proche de x. Il y a donc un gaspillage évident de la bande passante.

Un gain peut être réalisé si x connaît à priori la distance le séparant de y. Il suffit d'ajouter un compteur dans l'en-tête du message établissant la connexion, initialisé à $d_G(x, y)$, puis de le décrémenter à chaque réception par les sommets intermédiaires, et de stopper la diffusion lorsque le compteur vaut 0. Le nombre de messages est donc limité au nombre de liens dans la boule de rayon $d_G(x, y)$ centrée en x.

Ainsi, si la distance $d_G(x, y)$ peut être calculée (ou même approximée) en x seulement avec la donnée de l'étiquette y, on peut limiter le coût de la connexion.

3.2.2 Routage sans mémoire

Le routage sans mémoire (*memory-free routing*) est une variante du routage compact. Il s'agit d'étiqueter les sommets de sorte qu'un message puisse être acheminé entre deux sommets seulement par leurs étiquettes (et les étiquettes des sommets intermédiaires le long de la route), et sans aucune autre information stockée dans les sommets.

Un étiquetage de distance peut servir à router un message suivant un plus court chemin. Il suffit de collecter les étiquettes de ses voisins. Après avoir calculé la distance entre la source et la destination, la source calcule la distance à la destination depuis chacun de ses voisins, en sélectionne un pour qui la distance a diminué, et lui transmet le message.

En fait, nous le reverrons au chapitre 4 concernant les schémas d'étiquetage généralisés, toute fonction de routage peut être transformée en routage sans mémoire, en accolant à l'adresse du sommet u toutes les informations nécessaires concernant sa fonction locale de routage, R_u (cf. page 48). Le problème est donc très proche de celui du routage compact à l'exception que l'information à stocker s'est déplacée du sommet vers l'adresse du sommet. Par exemple, dans [Pel99], Peleg montre que les arbres ont un routage sans mémoire (de plus courts chemin) avec des adresses de taille $O(\log^2 n)$, mais il n'est pas démontrée que cette borne est la meilleure possible.

3.3 Définitions

Définissons la notion d'étiquetage de distance plus précisément. Étant donné un graphe G (supposé non orienté[3] et connexe) et deux sommets u et v, notons $d_G(u, v)$ la distance entre u et v dans G. Un *étiquetage sommet* est une fonction entière à valeurs non nulles L qui associe une étiquette $L(u, G)$ à chaque sommet u de G.

3. Les définitions s'étendent cependant sans difficultés particulières aux graphes orientés.

Un *décodeur de distance* est une fonction à valeur entière, f, chargée du calcul de la distance : étant donné deux étiquettes λ_1 et λ_2 (sans connaître de quel graphe elles sont issues), elle retourne $f(\lambda_1, \lambda_2)$. La paire $\langle L, f \rangle$ forme un *étiquetage de distance* pour G si $f(L(u,G), L(v,G)) = d_G(u,v)$ pour toute paire distincte[4] $u, v \in V(G)$. L'étiquetage de distance est (s,r)-approximé si

$$d_G(u,v) \leqslant f(L(u,G), L(v,G)) \leqslant s \cdot d_G(u,v) + r$$

pour toute paire distincte $u, v \in V(G)$. Plus généralement, $\langle L, f \rangle$ est un *schéma d'étiquetage de distance* (s,r)-approximé pour une famille de graphe $\mathcal{G}$ si c'est un étiquetage de distance (s,r)-approximé pour tout $G \in \mathcal{G}$. Soit $|L(u,G)|$ la taille (en bits) de l'étiquette $L(u,G)$ écrite en binaire associé au sommet u de $G \in \mathcal{G}$, et notons

$$\begin{aligned} \ell_{\langle L,f \rangle}(\mathcal{G}) &= \max_{G \in \mathcal{G}} \max_{u \in V(G)} |L(u,G)| \\ \ell_{(s,r)}(\mathcal{G}) &= \min \left\{ \ell_{\langle L,f \rangle}(\mathcal{G}) \right\} \end{aligned}$$

où le minimum est pris sur tous les schéma d'étiquetage de distance (s,r)-approximé $\langle L, f \rangle$ pour $\mathcal{G}$.

Outre la taille des étiquettes, on s'intéresse au temps de calcul du décodeur de distance (complexité en temps dans le pire des cas). Nous supposerons, comme aux chapitres précédents, un modèle *word-RAM* où les instructions arithmétiques standards sur des entiers de taille $O(\log n)$ bits sont exécutées en temps constant.

3.4 Bornes générales sur la taille des étiquettes

Dans cette section on notera $\mathcal{G}_n$ la famille de tous les graphes (connexes) à n sommets.

3.4.1 Bornes supérieures

La *matrice de distances* d'un graphe G est une matrice telle que $M[i][j] = d_G(i,j)$ pour tout $i, j \in V(G)$. Un schéma d'étiquetage de distance trivial consiste à poser, pour tout sommet $i \in V(G)$, $L(i,G) = \langle i, M[i] \rangle$, où $M[i]$ est la i-ème ligne de la matrice de distances de G. Le décodeur défini par $f(\langle i, M[i] \rangle, \langle j, M[j] \rangle) = M[i][j]$, le j-ème élément du vecteur ligne $M[i]$, peut être calculé en temps constant. La taille des étiquettes est $n \log n + O(\log n)$ dans le pire des cas (il faut coder i ainsi que n distances, chacune variant dans l'ensemble $\{0, \ldots, n-1\}$). Donc, $\ell_{(1,0)}(\mathcal{G}_n) \leqslant n \log n + O(\log n)$.

Graham et Pollak ont proposé dans [GP72] un schéma général pour la famille des graphes à n sommets. Ce schéma étiquète chacun des sommets par un mot de q_n symboles

4. On peut éliminer de la définition la condition $u \neq v$. Cependant, dans la suite nous supposerons que l'on ne s'intéresse pas à calculer la distance d'un sommet à lui-même, de façon à pouvoir obtenir des longueurs d'étiquettes plus petites que $\log n$.

pris sur l'alphabet $\{0,1,*\}$ de façon à ce que la distance entre deux sommets corresponde à la distance de Hamming entre deux mots (la distance entre * et les symboles 0 ou 1 est nulle). Référencé comme *The Squashed Cube Conjecture*, Winkler dans [Win83] a prouvé que $q_n \leqslant n-1$ pour tout n, impliquant un schéma d'étiquetage de distance avec des étiquettes de taille $n\log_3(2) \approx 1.58n$ bits. D'où $\ell_{(1,0)}(\mathcal{G}_n) \leqslant 1.58n$. Cependant le temps de calcul de la distance est en $\Theta(n)$.

Avec une approche très différente, nous avons dans [GPPR01] proposé un schéma avec des étiquettes de taille $O(n)$, mais avec un temps de décodage de seulement $O(\log\log n)$. Nous en donnons l'idée principale. C'est un schéma récursif basé sur une collection d'ensembles 2^i-dominants de cardinalité au plus $n/2^i$, pour $i \in \{0,\ldots,\lceil\log\log n\rceil\}$. On associe à chaque sommet x et chaque ensemble 2^i-dominant S_i son *pivot* $p_i(x)$, un sommet de S_i atteignant $d_G(x,S_i)$. La distance entre x et y peut être calculée par $d_G(x,y) = d_G(p_i(x),p_i(y)) + \delta(y)$, où $\delta(y) \in [-2^{i+1}, 2^{i+1}]$. Et pour calculer la distance entre les pivots, on peut appliquer le schéma récursivement avec le niveau $i+1$. En initialisant $i=0$, il suffit de stocker en x toutes les corrections de distance, une pour chaque $y \in V(G)$. Chaque correction $\delta(y)$ peut être codée sur seulement $\lceil\log(2\cdot 2^{i+1}+1)\rceil \leqslant i+3$ bits (la longueur de l'intervalle pour $\delta(y)$ est $2^{i+2}+1$). Cela fait un total de $(i+3)n = 3n$ bits pour le premier niveau, et $(i+3)|S_{i-1}|$ de manière générale pour le niveau $i>0$. On stocke récursivement le schéma pour les sommets de l'ensemble 2^{i+1}-dominant qui est de taille $|S_{i+1}| \leqslant |S_i|/2$. On montre que la série donnant la taille des étiquettes (du type $n\sum_{i\geqslant 0} O(i/2^i)$) est bornée par $11n + O(\log n \cdot \log\log n)$.

Enfin, Peleg dans [Pel99] a construit, pour tout $k \geqslant 1$, un schéma d'étiquetage de distance $(8k,0)$-approximé utilisant des étiquettes de taille au plus $O(k\, n^{1/k}\log n\, \log D)$, où D est le diamètre (éventuellement pondéré) du graphe. Il en découle que $\ell_{(2\log n,0)}(\mathcal{G}_n) = O(\log^3 n)$, résultat qui peut aussi se déduire (et même être légèrement amélioré) des travaux de [Bou85, LLR95, Bar98] sur le plongement de métriques générales dans des espaces Euclidiens de faible dimension.

3.4.2 Bornes inférieures : une technique générale

Dans ce paragraphe nous montrons une technique générale pour borner inférieurement $\ell_{(s,0)}(\mathcal{F})$, pour une famille de graphes $\mathcal{F}$.

Soit $\mathcal{F}$ une famille de graphes et A un ensemble de sommets tels que, pour tout $G \in \mathcal{F}$, $A \subseteq V(G)$. Soit $k>1$ un réel. La famille de graphe $\mathcal{F}$ est dite une *(A,k)-famille* si pour tout $G,H \in \mathcal{F}$, il existe deux sommets $x,y \in A$ tels que soit $d_G(x,y) \geqslant k\cdot d_H(x,y)$, soit $d_H(x,y) \geqslant k\cdot d_G(x,y)$. La famille $\mathcal{F}$ est dite une *A-famille* s'il existe un réel $k>1$ telle que $\mathcal{F}$ est une (A,k)-famille. En particulier, toute famille de graphes finis et définie sur l'ensemble de sommets $V_n = \{1,\ldots,n\}$, est une $(V_n, 1+1/n)$-famille.

Dans [GPPR01] nous montrons le résultat suivant, découlant d'un simple argument de comptage. Pour toute (A,k)-famille $\mathcal{F}$, $k>1$, et pour tout réel $s<k$,

$$\ell_{(s,0)}(\mathcal{F}) \;\geqslant\; \frac{1}{|A|}\cdot\log|\mathcal{F}|\;. \tag{3.1}$$

3.4.3 Applications

Les applications de l'équation (3.1) sont multiples. En voici quelques exemples.

Considérant la sous-famille de $\mathcal{G}_n$, $\mathcal{F}_n$, des graphes à n sommets et de diamètre 2, nous obtenons directement que, pour tout $s<2$, $\ell_{(s,0)}(\mathcal{G}_n) \geqslant n/2 - O(1)$. En effet, $\mathcal{F}_n$ est une $(V_n,2)$-famille, et il est facile de se convaincre que $|\mathcal{F}_n| \geqslant \frac{1}{2}\cdot 2^{\binom{n}{2}}$.

En considérant la famille des graphes biparties à n sommets dont la plus petite partition comprend k sommets, $\mathcal{B}_k$, on montre de manière similaire que : pour tout $s<3$, et pour tout k tel que $2\log n \leqslant k \leqslant n/2$, $\ell_{(s,0)}(\mathcal{B}_k) \geqslant k(n-k)/n - 2\log n$. (Il suffit de compter le nombre de graphes de cette famille qui sont de diamètre 3.) Pour $k=n/2$, on déduit que $\ell_{(s,0)}(\mathcal{G}_n) \geqslant n/4 - O(\log n)$ pour tout $s<3$ étendant la borne précédente sur s.

Les graphes de $\mathcal{B}_k$ sont de largeur arborescente bornée par k. Donc pour la famille $\mathcal{T}_k$ des graphes de largeur arborescente au plus k, nous avons, pour tout $s<3$, $\ell_{(s,0)}(\mathcal{T}_k) = \Omega(k)$.

Bien que les graphes planaires soient de largeur arborescente $O(\sqrt{n})$ la borne inférieure précédente en $\Omega(\sqrt{n})$ ne s'applique pas : en effet les graphes de $\mathcal{B}_{\sqrt{n}}$ ne sont pas planaires. Nous avons donc construit une famille plus spécifique composée uniquement de graphes planaires. Plus précisément, nous donnons dans [GPPR01] une A-famille $\mathcal{Q}_n$ de graphes planaires à n sommets, avec $|A| = O(n^{1/3})$ et $\log|\mathcal{Q}_n| = \Omega(n^{2/3})$. Elle est composée des sous-graphes du graphe représenté sur la figure 3.1 où chaque arête $(x_{i,j}, y_{i,j})$ peut être présente ou supprimée. Par l'ajout de sommets de degré deux (non représenté sur la figure), on force le plus court chemin de u_i à v_i à passer par $x_{i,j}$ et $y_{i,j}$. L'ensemble $A=\{u_1,\ldots,u_k,v_1,\ldots,v_k\}$. Par l'équation (3.1) on déduit que $\ell_{(1,0)}(\mathcal{Q}_n) = \Omega(n^{1/3})$.

Pour la famille $\mathcal{D}_n$ des graphes de degré borné, nous montrons, avec la même technique, que $\ell_{(1,0)}(\mathcal{D}_n) = \Omega(\sqrt{n})$, laissant ouverte la question de savoir si cette borne est fine.

Soit G_0 un graphe de degré minimum δ et de maille au plus g, et comprenant un nombre minimal de sommets. En utilisant un résultat de la théorie des graphes extrêmes [Bol78, pp. 108]), et en considérant cette fois la famille des sous-graphes connexes de G_0, nous montrons encore grâce à l'équation (3.1), que pour tout k tel que $2 \leqslant k \leqslant o(\log n)$, et pour tout $s<k$, $\ell_{(s,0)}(\mathcal{G}_n) \geqslant \Omega(n^{1/(k-\rho)})$, où $\rho = 1 + (k \bmod 2)$. Ainsi, par le résultat de Peleg [Pel99] mentionné au paragraphe 3.4.1, nous avons :

$$\forall\, s = o(\log n), \quad \ell_{(s,0)}(\mathcal{G}_n) \;=\; n^{1/\Theta(s)}\;.$$

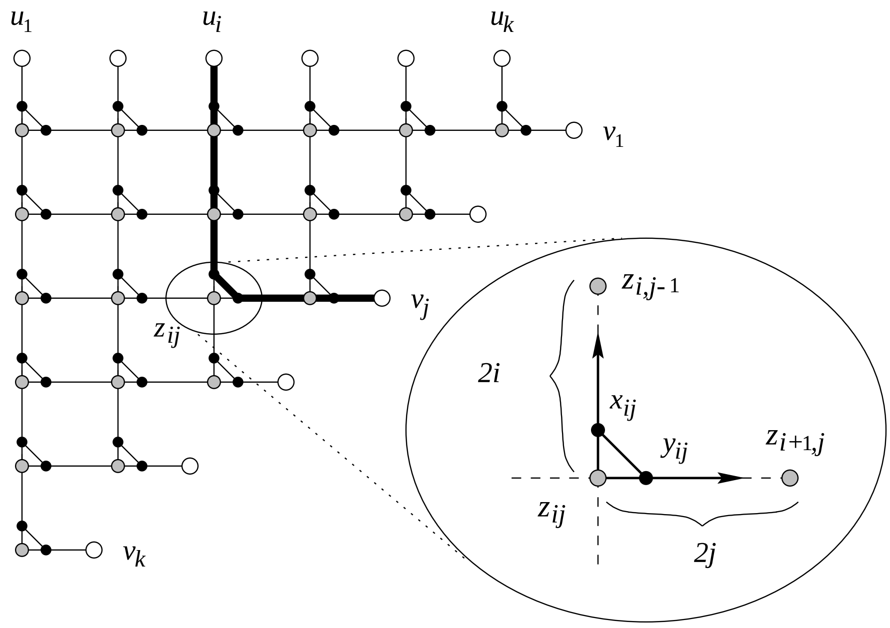


FIG. 3.1 – *Une A-famille de graphes planaires, $\mathcal{Q}_n$.*

Question ouverte

• Quelle est la complexité de $\ell_{(1,0)}(\mathcal{D}_n)$, les graphes de degré borné? (La complexité se située entre $\Omega(\sqrt{n})$ et $O(n)$.)

3.4.4 Temps exponentiel du calcul de la distance

Nous avons vu au chapitre 2 page 30, qu'une table de routage mise en œuvre par une structure de données trop compacte pouvait produire une latence aussi grande que n'importe quelle pile d'exponentielles de taille constante.

Le même phénomène se produit dans le cadre de l'étiquetage de distance. Nous montrons dans [GPPR01] le résultat général suivant qui découle de l'équation (3.1). Définissons d'abord, $\xi_h(n)$ la fonction "pile d'exponentielles de hauteur h" définie récursivement par $\xi_h(n) = 2^{\xi_{h-1}(n)}$ et $\xi_0(n) = n$. Par exemple, $\xi_2(n) = 2^{2^n}$.

Soit $\mathcal{F}_n$ une (A, k)-famille de graphes à n sommets reconnaissable en temps $\tau(n)$, c'est-à-dire telle que la question "G appartient-t-il à $\mathcal{F}_n$" puisse être décidée en temps au plus $\tau(n)$. On suppose que $A \subseteq \{1, \ldots, n\}$ est un ensemble dont la fonction caractéristique peut être décrite par un nombre constant de bits sachant n. (Cette hypothèse est technique. On peut passer outre moyennant quelques modifications de l'énoncé simplifié présenté ci-dessous. Pour plus de détails, se reporter à [GPPR01].) Supposons qu'il existe une constante h_0 vérifiant $\xi_{h_0}(|\mathcal{F}_n|) > n$. Alors, pour une infinité de n, il existe un graphe $G \in \mathcal{F}_n$ avec la propriété suivante :

1. Il existe un schéma d'étiquetage de distance $(1,0)$-approximé sur $\mathcal{F}_n$ générant des étiquettes de taille au plus $3\log n + O(\log\log n)$ pour le graphe G.
2. Pour tout étiquetage de distance $(s,0)$-approximé sur $\mathcal{F}_n$, $s < k$, générant des étiquettes de taille au plus $(\log|\mathcal{F}_n|)/|A| - O(\log\log|\mathcal{F}_n|)$ pour G, le temps de calcul pour décoder la distance sur G est plus grand que $\xi_h(n) - \tau(n)$, pour toute constante h.

Nous allons voir une seule application de ce résultat général : les graphes planaires. Le même résultat s'applique pour les graphes de diamètre deux, les graphes de degré borné, et les graphes de largeur arborescente bornée.

Soit $\mathcal{Q}_n$ la A-famille décrite au paragraphe 3.4.3 (voir figure 3.1). On peut fixer $k = 1 + 1/n$ pour obtenir une (A,k)-famille. Comme, $|\mathcal{Q}_n| = 2^{\Omega(n^{2/3})}$, on a clairement pour $h_0 = 0$, $\xi_{h_0}(|\mathcal{Q}_n|) > n$. De plus, la famille est reconnaissable en temps $\tau(n) = O(n)$. Sachant n, A est fixé à l'ensemble des u_i et des v_i, les seuls sommets de degré 1. Donc en appliquant le résultat précédent, il existe des graphes planaires (membres de la famille $\mathcal{Q}_n$) qui supportent un schéma avec des étiquettes de taille $3\log n + o(\log n)$ et qui nécessitent un temps $2^{2^n} - O(n)$ (et même plus) pour décoder la distance, et ceci, même si les étiquettes sont de taille $c\,n^{1/3} - O(\log n)$ pour une certaine constante $c > 0$.

3.5 Arbres et graphes de largeur arborescente bornée

Les notions de décomposition en arbre et de largeur arborescente sont relatives aux travaux de Roberston et Seymour sur les mineurs de graphes. Se référer à [RS86] pour plus de précisions. Un *arbre de décomposition* de G est un arbre T tel que :

1. $\bigcup_{X\in V(T)} X = V(G)$;
2. pour tout $\{u,v\} \in E(G)$, il existe $X \in V(T)$ tel que $u,v \in X$;
3. pour tout $X,Y,Z \in V(T)$, si Y est sur le chemin de X à Z dans T, alors $X \cap Z \subseteq Y$.

En reprenant les notations précédentes, on note $\mathcal{T}_k$ la famille des graphes de largeur arborescente au plus k.

3.5.1 Bornes supérieures : un schéma général

Notons tout d'abord que les notions de largeur arborescente et de séparateur récursif sont équivalentes, cf. [Bod98]. Un graphe de largeur arborescente k a nécessairement un séparateur récursif de taille k (et est donc k-décomposable d'après [FJ90]), et inversement. Il en découle un certain nombre de résultats tous basés sur un schéma d'étiquetage que nous allons expliciter. Ce schéma est un affinement de celui présenté par Peleg dans [Pel99].

Tout d'abord, on remarque que tout arbre à n sommets possède un sommet, appelé dans la suite *centroïde*, dont la suppression décompose l'arbre en composantes connexes, chacune de taille au plus $n/2$.

Considérons un graphe G de largeur arborescente au plus k, et soit T un arbre de décomposition de largeur au plus k tel que $|V(T)| \leqslant n$. (On montre qu'il est toujours possible de construire un tel arbre.) Soit U un sommet de T, $U \subseteq V(G)$. On associe à U une séquence d'entiers construits comme suit. Soit S_1 le premier centroïde de T. Chaque composante connexe F de $T \setminus \{S_1\}$ est numérotée par un entier unique $c(F) \in \{1, \ldots, \deg_T(S_1)\}$ tel que $c(A) \leqslant c(B)$ si $|V(A)| \geqslant |V(B)|$ pour toutes composantes A, B de $T \setminus \{S_1\}$. Soit T_1 la composante connexe de $T \setminus \{S_1\}$ contenant U, et soit $c_1 = c(T_1)$.

On applique cette décomposition de T récursivement à T_i, $i \geqslant 1$, afin d'obtenir S_{i+1}, T_{i+1} et c_{i+1} tel que $U \in V(T_{i+1})$ et $c_{i+1} = c(T_{i+1})$, jusqu'à obtenir $U = S_{i+1}$. Soit h l'indice tel que $U = S_{h+1}$. On note $C_U = (c_1, \ldots, c_h)$. Chaque $S_i \in V(T)$ est un ensemble de $k_i \leqslant k+1$ sommets du graphe noté $S_i = \left\{s^i_1, \ldots, s^i_{k_i}\right\}$. Remarquons que $h + 1 \leqslant \log n$.

Maintenant, pour tout sommet $x \in U$, on étiquète[5] x dans G par la paire $L(x, G) = \langle C_U, \Lambda_x \rangle$, où $\Lambda_x = (M_1(x), \ldots, M_h(x))$, $M_i(x)$ étant la liste des distances de x à tous les sommets du centroïde S_i de U, c'est-à-dire $M_i(x) = (d_G(x, s^i_1), \ldots, d_G(x, s^i_{k_i}))$.

La distance de $x \in U$ à $x' \in U'$ dans G peut-être calculée de la manière suivante :

1. Calculer le plus petit indice i_0 tel que les suites C_U et $C_{U'}$ diffèrent à la $(i_0 + 1)$-ème position. Ainsi S_{i_0} est le plus petit ancêtre commun dans T entre U et U'.
2. Extraire $M_{i_0}(x)$ et $M_{i_0}(x')$ du champ Λ_x et $\Lambda_{x'}$ de $L(x, G)$ et $L(x', G)$ respectivement.
3. La distance $d_G(x, x') = \min_z \{d_G(x, z) + d_G(z, x')\}$ où le minimum est pris pour tout $z \in S_{i_0}$. (Toutes ces distances sont ordonnées et stockées dans les suites $M_{i_0}(x)$ et $M_{i_0}(x')$.) En effet, l'ensemble de sommets S_{i_0} déconnecte G, donc tout plus court chemin entre x et x' passe par un sommet $z \in S_{i_0}$.

On peut montrer que la suite C_U peut être représentée sur $4 \log n$ bits, parce que la composante T_i numérotée c_i vérifie $|V(T_i)| \leqslant |V(T_{i-1})|/c_i$, donc $\sum_{i=1}^h \log c_i = \log(\prod_{i=1}^h c_i) \leqslant \log n$. La suite Λ_x peut être codée sur $(\sum_{i=1}^h k_i) \log n \leqslant O(k \log^2 n)$. En fait, de manière générale, pour $k = k(n)$ fonction monotone non décroissante de n, on a :

$$\ell_{(1,0)}(\mathcal{T}_k) \;=\; O(K(n) \cdot \log n) \tag{3.2}$$

où $K(n) = \sum_{i=1}^{\log n} k(n/2^i)$. Notons que $K(n) \leqslant k(n) \log n$, mais que pour $k(n) \geqslant n^\epsilon$, $\epsilon > 0$ constante, $K(n) = O(k(n))$. D'après la borne inférieure en $\Omega(k(n))$ précédente, résultante de la famille $\mathcal{B}_{k(n)}$, l'équation (3.2) est donc au plus éloignée d'un facteur $O(\log^2 n)$ de la valeur optimale.

5. Comme x peut apparaître plusieurs fois dans différents sommets de T, on prend soin d'étiqueter chaque sommet qu'une seule fois.

3.5.2 Applications du schéma général

Les applications de l'équation (3.2) sont nombreuses, en commençant par les graphes planaires qui ont une largeur arborescente en $O(\sqrt{n})$ [LT79, Bod98]. Ils supportent donc un schéma avec des étiquettes de taille $O(\sqrt{n}\log n)$. Plus généralement, les graphes de genre g ont une largeur arborescente $O(\sqrt{gn})$ [GHT84], et les graphes qui n'ont pas K_h comme mineur ont une largeur arborescente $O(\sqrt{h^3 n})$ [AST90] et aussi $O(h\sqrt{n\log n})$ [PRS94] qui est conjecturée pour être $O(h\sqrt{n})$. Rappelons que les arbres, les graphes planaires-extérieurs et les graphes séries-parallèles sont de largeur arborescente $O(1)$, donc, par l'équation (3.2), supportent tous un étiquetage de distance avec des étiquettes de taille $O(\log^2 n)$.

Question ouverte

• Soit $\mathcal{P}_n$ la famille des graphes planaires à n sommets. Quelle est la complexité de $\ell_{(1,0}(\mathcal{P}_n)$?

D'après ce qui précède, la borne est située entre $\Omega(n^{1/3})$ et $O(\sqrt{n}\log n)$. Dans [GPPR01] nous proposons un schéma $(3,0)$-approximé pour les graphes planaires montrant que $\ell_{(3,0)}(\mathcal{P}_n) = O(n^{1/3}\log n)$.

3.5.3 Approximations et temps de calcul du schéma général

la taille des étiquettes donnée par l'équation (3.2) peut être diminuée si l'on s'autorise une approximation avec un facteur $s > 1$. En effet, dans [GKK+00], nous montrons :

$$\ell_{(1+1/\log n,0)}(\mathcal{T}_k) \;=\; O(K(n)\cdot\log\log n) \tag{3.3}$$

avec les même notations qu'au paragraphe 3.5.1. En effet, il suffit de coder différemment les distances dans les listes $M_i(x)$. Chacune de ces listes comporte k distances que l'on peut coder avec $k\log n$ bits. Cependant, si l'on code seulement le bit de poids fort de chacune des distances, on ne peut retenir que $\log\log n$ bits d'informations par distance alors que l'erreur commise sur cette distance n'est que $s = 2$. Dans [GKK+00], nous expliquons comment avec $2\log\log n$ bits par distance on peut faire diminuer cette erreur à $s = 1 + 1/\log n$. Notons que cette distance peut correspondre à une distance dans un graphe pondéré. Si bien que l'équation (3.3) peut se réécrire en fonction du diamètre pondéré D du graphe en :

$$\ell_{(1+1/\log D,0)}(\mathcal{T}_k) \;=\; O(K(n)\cdot\log\log D) \tag{3.4}$$

Donc les arbres pondérés de diamètre D supportent un schéma $(1+1/\log D, 0)$-approximé avec des étiquettes de taille $O(\log n\cdot\log\log D)$.

Concernant le temps de calcul de la distance, le schéma du paragraphe 3.5.1 nécessite à priori un temps de $O(h)$ pour trouver i_0, puis d'un temps $O(k)$ pour trouver la distance, soit un total de $O(k + \log n)$. Cependant nous remarquons que le calcul de i_0 nécessite uniquement un calcul sur les suites C_U et $C_{U'}$ qui sont codées sur au plus $4 \log n$ bits. Nous avons montré dans [GR00] qu'en pré-calculant une table auxiliaire de taille $O(\log n)$ que l'on rajoute aux étiquettes, on peut calculer i_0 en temps $O(1)$, la division entière sur les entiers de taille $O(\log n)$ étant autorisée[6]. Donc le temps de calcul est $O(k)$. Il en découle, pour les arbres et plus généralement pour k borné, que le schéma sur $\mathcal{T}_k$ avec des étiquettes de taille $O(\log^2 n)$ (pour une $(1,0)$-approximation) ou de taille $O(\log n \cdot \log\log n)$ (pour une $(1 + 1/\log n, 0)$-approximation) nécessite un temps constant pour décoder la distance.

3.5.4 Borne inférieure

Malheureusement, l'équation (3.1) page 38 ne donne pas de résultat intéressant pour montrer une borne inférieure concernant les graphes de largeur arborescente bornée. Surtout dans le cas des arbres ($k = 1$). En effet, le nombre d'arbres est $n^{n-2} = 2^{\Theta(n \log n)}$ (formule de Cayley), qui donne une borne inférieure de $\Omega(\log n)$, alors que la borne supérieure est $O(\log^2 n)$ (équation (3.2)). Nous avons développé une nouvelle technique de borne inférieure spécifique aux cas des arbres, mais qui s'applique à toute la famille des graphes de largeur arborescente bornée étant donné que, dans les deux cas, les tailles des étiquettes ont la même complexité (elles ne diffèrent que par une constante multiplicative).

Dans [GPPR01], nous prouvons que la complexité réelle est en $\Theta(\log^2 n)$. L'idée principale est de considérer une famille particulière d'arbres (des arbres binaires), notée $\mathcal{H}$, et de calculer la cardinalité de l'ensemble des étiquettes utilisées pour étiqueter tous les arbres de $\mathcal{H}$. On considère donc l'ensemble $Z = \{L(u,T) \mid u \in V(T),\, T \in \mathcal{H}\}$ pour un étiquetage L arbitraire. Afin de montrer que Z est de cardinalité assez grande, on montre qu'une étiquette $\ell \in Z$ ne peut être utilisée que par les sommets d'un nombre limité d'arbres différents de $\mathcal{H}$. Plus précisément on montre que, pour un étiquetage L quelconque,

$$|Z| \;\geqslant\; \left(\sqrt{n/3}\right)^{\frac{1}{2}\log\left(\sqrt{n/3}\right)} .$$

Il en découle qu'il existe au moins un arbre $T_0 \in \mathcal{H}$ contenant un sommet où L affecte une étiquette de taille au moins $\log |Z| \geqslant \frac{1}{8} \log^2(n/3)$.

En modifiant et en étendant cette preuve, nous avons montré dans [GKK$^+$00] que pour $s < 1 + 6/\log n$, la taille minimale des étiquettes pour un schéma $(s, 0)$-approximé est au moins $\Omega(\log n \cdot \log\log n)$ pour la famille des arbres binaires, borne optimale d'après l'équation (3.3). (Dans le cas des arbres, $K(n) = \log n$.)

Nous avons aussi trouvé, en adaptant la preuve précédente des arbres, une borne inférieure de $\Omega(\log^2 n)$ sur la taille des étiquettes des schémas d'étiquetage de distance pour la famille des graphes planaires-extérieurs de largeur-chemin au plus 2. (Cette famille de graphes est composée de cycles agencés en chemin de sorte que deux cycles consécutifs ne

6. Sinon on peut faire le calcul en temps $O(\log\log n)$.

partagent qu'une seule arête.) Cette borne est optimale car les graphes planaires-extérieurs sont de largeur arborescente bornée.

Pour résumer, nous avons montrer que $\ell_{(1,0)}(\mathcal{T}_k) = \Theta(\log^2 n)$, et que $\ell_{(1+1/\log n,0)}(\mathcal{T}_k) = \Theta(\log n \cdot \log\log n)$ pour tout $k = O(1)$. pour compléter ces résultats, notons qu'il est facile de voir que ces bornes ne varient pas en ordre de grandeur si l'on considère des $(1, r)$-approximations ou des $(1 + 1/\log n, r)$-approximations avec $r = O(1)$. Pour s'en convaincre, il suffit de considérer des subdivisions uniformes de toutes les arêtes d'un cas extrême, cf. [GKK+00].

Question ouverte

- Existe-t-il un schéma d'étiquetage de distance pour les arbres avec des étiquettes de taille moyenne en $o(\log^2 n)$? (moyenne calculée sur tous les sommets du même arbre).

En effet, notre borne inférieure présentée brièvement ci-dessus, ne permet pas de conclure que $\Theta(n)$ sommets nécessitent des étiquettes de taille $\Omega(\log^2 n)$.

3.6 Cordalité et graphes d'intersections

Un graphe est *k-cordal* s'il ne contient pas de cycle sans corde de longueur plus grande que k. La *cordalité* d'un graphe G étant le plus petit k tel que G soit k-cordal.

Le famille des graphes k-cordal pour k constant contient de nombreuses familles de graphes bien connues. Les arbres sont 2-cordaux et les graphes *triangulés* sont exactement les graphes 3-cordaux. Certains graphes d'intersections entrent dans cette catégorie. Les graphes d'intervalles, par exemple, sont 3-cordaux, les graphes de permutations sont 4-cordaux.

3.6.1 Borne inférieure

Nous montrons dans [GKK+00] que les graphes k-cordaux n'ont pas de schéma d'étiquetage de distance avec des étiquettes courtes. Plus précisément, nous montrons que pour tout k, la famille des graphes k-cordaux, notée $\mathcal{C}_k$, vérifie $\ell_{(s,0)}(\mathcal{C}_k) \geqslant n/4 - O(1)$, pour tout $s < 2$.

En fait on se contente de montrer ce résultat pour la famille $\mathcal{C}_3$ (des triangulés) puisque $\mathcal{C}_k \subset \mathcal{C}_{k+1}$. La preuve découle encore de l'application de la borne inférieure générale résumée par l'équation (3.1) page 38. Considérons la famille $\mathcal{S}$ de tous les graphes à n sommets composés d'un clique C de $\lceil n/2 \rceil$ sommets, et d'un ensemble indépendant I de $\lfloor n/2 \rfloor$ sommets, tel que chaque sommet de I soit connecté à au moins un sommet de C. Comme tout cycle de longueur 4 ou plus dans un graphe quelconque de $\mathcal{S}$ possède au moins deux sommets dans C (et donc une corde), on a bien que $\mathcal{S} \subset \mathcal{C}_3$. De plus $|\mathcal{S}| = 2^{|I|\cdot(|C|-1)} \geqslant$

$2^{n^2/4-O(n)}$. Il est aussi facile de vérifier que deux graphes $G, H \in \mathcal{S}$ possèdent toujours deux sommets u, v tels que $d_G(u,v) = 1$ alors que $d_H(u,v) \geqslant 2$, ou le contraire. Donc $\mathcal{S}$ est une $(V_n, 2)$-famille, $V_n = \{1,\ldots,n\}$. On conclut directement par l'équation (3.1).

Les graphes sans *triple-astéroïde* sont des graphes qui ne possèdent pas trois sommets x, y, z tels qu'entre chaque paire de ces sommets il existe un chemin évitant le voisinage du troisième. Les graphes sans triple-astéroïde sont 5-cordaux et contiennent les graphes d'intervalles, les graphes de permutations et de co-comparabilité [Gol80].

Comme pour les graphes k-cordaux, il est facile de construire, une $(V_n, 2)$-famille $\mathcal{S}'$ de graphes sans triple-astéroïde, et de montrer par l'équation (3.1)q)ue pour tout $s < 2$, $\ell_{(s,0)}(\mathcal{S}') \geqslant n/4 - O(1)$. (La famille est similaire à $\mathcal{S}$ exceptée que l'indépendant I est remplacé par une clique, rendant impossible l'existence de triple-astéroïde).

3.6.2 Un schéma $(1, r)$-approximé

Nous avons vu que les graphes triangulés, les graphes sans triple-astéroïde, et plus généralement, les graphes k-cordaux ne peuvent avoir un schéma d'étiquetage de distance exact avec des étiquettes courtes. Qu'en est-il des schémas approximés ?

Il est connu que tout graphe triangulé G possède une décomposition suivant un arbre T (décomposition définie au paragraphe 3.5 page 40) telle que tout sommet de T induit une clique dans G. Nous avons étendu ce résultat aux graphes k-cordaux. Nous montrons que tout graphe G de cordalité k admet une décomposition en arbre, T, telle que tout $U \in V(T)$ vérifie $d_G(U) \leqslant k/2$, où $d_G(U) = \max_{x,y \in U} d_G(x,y)$. (Donc on retrouve la propriété précédente des graphes triangulés, $d_G(U) = 1$, en prenant $k = 3$.) De plus cette décomposition est constructible en temps polynômial.

Ainsi, en modifiant le schéma général utilisé dans le paragraphe 3.5 pour les arbres, nous obtenons que $\ell_{(1,\lfloor k/2 \rfloor)}(\mathcal{C}_k) = O(\log^2 n)$. (On peut le modifier de la façon suivante : au lieu de coder la liste $M_i(x)$ de toutes les distances de x au séparateur S_i, on ne stocke qu'une seule distance, $d_G(x, S_i)$, la distance entre x et le plus proche sommet de S_i.) L'erreur qui est faite est au plus le diamètre du séparateur S_i, donc une valeur additive $r = \lfloor k/2 \rfloor$. De plus le temps de calcul est constant en utilisant les idées présentées au paragraphe 3.5.3.

3.6.3 Graphes possédant une paire dominante

Katz, Katz[7] et Peleg ont montré dans [KKP00] que les graphes d'intervalles (qui sont 3-cordaux), et les graphes de permutations (qui sont 4-cordaux) possèdent un schéma d'étiquetage de distance (exact) avec des étiquettes de taille $O(\log^2 n)$. Pour ces deux familles spécifiques, notre résultat du paragraphe 3.6.2 est moins bon puisqu'avec des étiquettes de taille aussi en $O(\log^2 n)$, il donne une $(1, 1)$-approximation pour les graphes d'intervalles,

7. Mari et femme.

et une $(1,2)$-approximation pour des graphes de permutations. En utilisant le concept de "paire dominante" nous avons réduit la taille des étiquettes à $O(\log n)$ pour ces deux familles.

Plus précisément, un graphe G a une *paire dominante* s'il existe une paire de sommets, x, y, telle que pour tout chemin P de x à y, et tout sommet z, $d_G(z, V(P)) \leqslant 1$, c'est-à-dire P domine tous les sommets de G. Puisque les graphes sans triple-astéroïde ont une paire dominante [COS97], les graphes d'intervalles, les graphes de permutations, les graphes de tolérance bornée [Gol80], et les graphes de co-comparabilité, possèdent alors tous une paire dominante.

Pour les graphes 3-cordaux ayant une paire dominante nous obtenons un schéma $(1,1)$-approximé avec des étiquettes de taille $\log D + O(1)$ (D étant le diamètre du graphe). Avec des étiquettes de même taille, nous obtenons pour tous les graphes ayant une paire dominante un schéma $(1,2)$-approximé. Dans tous les cas, la distance peut être approximée en temps constant.

Donc pour la famille $\mathcal{I}_n$ des graphes d'intervalles à n sommets nous obtenons $\ell_{(1,1)}(\mathcal{I}_n) = O(\log n)$. Pour la famille $\mathcal{A}_n$ des graphes sans triple-astéroïde, $\ell_{(1,2)}(\mathcal{A}_n) = O(\log n)$.

Question ouverte

- Quelle est la complexité de la taille des étiquettes pour un schéma d'étiquetage de distance pour les graphes d'intervalles? Pour les graphes de permutations?

CHAPITRE 4

Généralisation et perspectives

4.1 Généralisation

Nous avons vu au travers des chapitres précédents que les problèmes du routage compact [1] et de l'étiquetage de distance pouvaient être étudiés selon trois grandes classes de paramètres à optimiser :

Paramètres	ROUTAGE COMPACT	ÉTIQUETAGE DE DISTANCE
Espace mémoire local	espace mémoire	taille des étiquettes
Qualité des solutions	facteur d'étirement, dilatation	(s, r)-approximations
Temps de calcul local	latence	temps de calcul

Une telle similarité entre les notions de "routage compact" et d'"étiquetage de distance" révèle que ces deux problèmes pourraient n'en former qu'un s'ils étaient repensés dans un cadre plus général.

4.1.1 Schéma d'étiquetage généralisé

On peut présenter et définir un schéma d'étiquetage généralisé de la façon suivante.

Soit $\mathcal{F}$ une famille de graphes, et soit $P(S, G)$ une fonction définie pour toute suite S de sommets de tout graphe $G \in \mathcal{F}$. Un *étiquetage sommet* sur $\mathcal{F}$ est une fonction L qui pour tout $G \in \mathcal{F}$ et tout sommet $u \in V(G)$ associe l'étiquette $L(u, G)$. Pour toute suite $S = (s_1, \ldots, s_k)$ de sommets d'un graphe G, on note $L(S, G)$ la suite d'étiquettes $(L(s_1, G), \ldots, L(s_k, G))$. Un *décodeur* est une fonction f qui, étant donnée une suite d'étiquettes Λ, retourne la valeur $f(\Lambda)$.

Un *P-schéma d'étiquetage* pour $\mathcal{F}$, ou encore un *schéma d'étiquetage généralisé* pour $\mathcal{F}$, est une paire $\langle L, f \rangle$ où L est un étiquetage sur $\mathcal{F}$ et f un décodeur tels que pour tout

1. Je range le routage par intervalles dans la catégorie "routage compact" car, comme nous l'avons vu au chapitre 2, le routage par intervalles est une méthode universelle de routage équivalente aux tables de routage.

$G \in \mathcal{F}$ et pour toute suite S de sommets de G, $f(L(S,G)) = P(S,G)$. On dit que P est d'arité k si les suites de sommets et d'étiquettes considérées ont k éléments.

Les schémas d'étiquetage de distance et les schémas d'étiquetage d'adjacence relèvent de ce même principe. En effet, si l'on pose la fonction d'arité 2, $\text{DIST}(x,y,G)$ retournant la distance entre tout couple de sommets d'un même graphe, nous avons bien que tout schéma d'étiquetage de distance est un DIST-schéma d'étiquetage. De même, en posant la fonction $\text{ADJ}(x,y,G) = 1$ si x est adjacent à y, et $\text{ADJ}(x,y,G) = 0$ si non, tout schéma d'étiquetage d'adjacence se trouve être un ADJ-schéma d'étiquetage.

Le problème du routage compact peut aussi s'exprimer en terme de schéma d'étiquetage généralisé. Il suffit de regrouper sous la même étiquette l'adresse du sommet et toutes les informations nécessaires concernant sa fonction locale de routage (cf. le routage sans mémoire rencontré page 35). Rappelons qu'une fonction de routage R sur un graphe G est une collection de fonctions locales de routage, $R = \{R_u \mid u \in V(G)\}$, chacune d'entres elles vérifiant : $R_u(\mathcal{L}(v)) = p$, où $\mathcal{L}(v)$ représente l'adresse de la destination et p le port de sortie en u par lequel le message doit être transmis pour atteindre v. (Nous nous limitons dans ce paragraphe aux fonctions de routage qui suivent le modèle des tables de routage. L'en-tête est non modifiable et comporte uniquement l'adresse de destination, cf. page 17. Donc ici, R_u est la table de routage du sommet u). Formellement, la paire $\langle L, f\rangle$ sera un schéma d'étiquetage "de routage" si L est de la forme $L(u,G) = \langle \mathcal{L}(u), R_u\rangle$ (la concaténation de l'adresse du sommet u et de la table de routage de u), et si f est définie par $f(L(u,G), L(v,G)) = R_u(\mathcal{L}(v))$.

4.1.2 Paramètres de complexité

On peut définir plusieurs paramètres à optimiser pour un P-schéma d'étiquetage $\langle L, f\rangle$. Nous proposons les trois suivants :

La ***taille des étiquettes***, qui est, pour une famille de graphes finis, $\mathcal{F}$, définie par la quantité :

$$\ell(P,\mathcal{F}) \;=\; \max_{G\in\mathcal{F}} \max_{u\in V(G)} |L(u,G)| \tag{4.1}$$

où $|L(u,G)|$ représente la longueur de $L(u,G)$ écrit en binaire (bits). Le chapitre 3 concerne donc le calcul de $\ell(\text{DIST},\mathcal{F})$ pour différentes familles de graphes $\mathcal{F}$, alors que les travaux de Kannan, Naor et Rudich [KNR88] sur la représentation implicite des graphes concernent le calcul de $\ell(\text{ADJ},\mathcal{F})$. Pour le routage compact, précisons que l'on sépare les étiquettes en deux parties donnant deux mesures de complexité : la taille des adresses d'une part (champs $\mathcal{L}(u)$), et la taille des tables de routage (champs R_u) d'autre part. Généralement, on fixe la taille des adresses à une certaine valeur suivant le modèle que l'on se fixe ($\log n$, $c\log n$, ou $O(\log^2 n)$ bits), puis on cherche à minimiser la taille de R_u. Dans le cas du routage sans mémoire (cf. page 35), on cherche à minimiser la somme des deux.

La ***qualité de la solution générée***, mesurant l'écart entre $f(L(S,G))$ et $P(S,G)$. Il est difficile de donner une définition précise tout en restant général, étant donné que le paramètre à optimiser dépend grandement de la fonction P (il faut pouvoir définir une distance pour P). Si $P(S, G)$ est un scalaire, on pourrait définir des (s, r)-approximations, comme pour le cas de l'étiquetage de distance (cf. page 36). Pour le routage compact, il s'agira plutôt de la longueur des routes générées (donc dépend d'une suite d'appels au décodeur f).

Le ***temps de calcul local***, qui peut être la complexité en temps (dans le pire des cas) du calcul de $f(L(S, G))$. On pourrait aussi définir le temps de calcul moyen, qui serait la moyenne sur toutes les suites S d'arité donnée sur le graphe. D'autres mesures sont possibles.

4.2 Perspectives

4.2.1 Vers des méta-théorèmes

On peut remarquer que beaucoup d'arguments proposés pour borner les paramètres de temps et d'espace pour le routage compact et l'étiquetage de distance sont relativement similaires. (Argument de comptage, complexité de Kolmogorov pour le temps de calcul exponentiel des structures de données trop compactes.) On pourrait donc s'intéresser aux P-schémas d'étiquetage pour des classes de fonctions P, afin d'établir des sortes de « méta-théorèmes ».

Donnons un exemple de tels théorèmes : l'indécidabilité de trouver un P-schéma d'étiquetage minimal. Trouver un P-schéma d'étiquetage consiste à fournir, pour chaque graphe d'une famille donnée, un ensemble d'étiquettes et surtout un décodeur. Trouver un décodeur calculable par une Machine de Turing et minimisant certains paramètres de taille de codage, est un problème difficile.

Pour illustrer cette difficulté, considérons le problème du routage compact dans le modèle des tables de routage de plus courts chemins. Les adresses sont choisies optimalement dans l'ensemble $\{1, \ldots, n\}$. Montrons qu'il est indécidable de calculer la taille de la plus petite table de routage d'un graphe G (connexe) à n sommets. Nous avons montré dans [GP96b] qu'il existe un graphe G_0 pour lequel la taille de la plus petite table de routage de plus courts chemins est au moins $c_0\, n \log n$ bits par sommet, pour une certaine constante $c_0 > 0$. Soit $B(n)$ une borne inférieure calculable[2] sur $c_0\, n \log n$. Considérons la fonction Graphe(i) qui retourne le i-ème graphe à n sommets. Notons que Graphe(i) est calculable. Supposons qu'il existe un algorithme $T_n(G)$ qui pour tout graphe G à n sommets retourne la taille de la plus petite table de routage de plus courts chemins pour G.

2. En analysant plus en détails la preuve de [GP96b], on peut montrer que $c_0 > 10^{-2}$. On peut donc choisir $B(n) = 10^{-2}\, n \log n$ par exemple, fonction de n calculable.

Considérons alors le programme suivant, noté Q_n :

1. $i := 0$
2. Faire

   $i := i + 1$
   $G :=$ Graphe(i)

   tant que $T_n(G) < B(n)$
3. Retourner i.

Par construction, Q_n retourne l'indice d'un graphe G^* dont la plus petite table de routage est au moins $B(n)$. Ce programme termine car il fait appel à des sous-fonctions elles-mêmes calculables, et de plus G^* existe par définition de $B(n)$ (notamment le programme s'arrête pour $G = G_0$). Notons que l'indice i_n tel que $G^* =$ Graphe(i_n) est calculable connaissant n. Remarquons aussi que le programme Q_n permet de construire une table de routage pour G^* particulièrement compacte. En effet, chaque sommet x mémorise n, une copie du programme Q_n, un algorithme D de plus courts chemins de type Dijkstra, ainsi qu'un entier identifiant x dans G^* (G^* est un graphe étiqueté avec comme ensemble de sommets $V = \{1, \ldots, n\}$, donc x peut être représenté de manière unique par $\lceil \log n \rceil$ bits d'information). La recherche d'un plus court chemin de x à y est réalisée en calculant l'entier i_n à partir de Q_n, puis G^* en lançant Graphe(i_n), et enfin en calculant par D un plus court chemin entre x et y. Tout ceci représente seulement $2 \log n + O(1)$ bits d'information[3], alors que toute table de routage sur G^* doit être au moins de $B(n) \gg 2 \log n + O(1)$: contradiction. Le programme $T_n(G)$ ne peut exister. Bien évidemment, la même contradiction apparaît si $T_n(G)$ est une approximation à un facteur multiplicatif $\Theta(n)$ près, puisque $B(n) = \Theta(n \log n)$.

Les mêmes types d'arguments pouvant être appliqués pour l'étiquetage de distance, on peut donc espérer pouvoir montrer sans trop d'efforts, et pour de nombreuses fonctions P, que trouver un P-schéma d'étiquetage $\langle L_0, f_0 \rangle$ minimisant la taille des étiquettes est indécidable (dans le sens où il n'y a pas de Machine de Turing permettant de le faire).

De la même manière, on doit pouvoir montrer qu'il y a toujours des graphes (même pour des graphes appartenant à des familles particulières) pour lesquels il est possible d'associer des étiquettes courtes, mais pour lesquels tout P-schéma associant des étiquettes même un peu plus grandes, produit des temps de calculs de complexité très grande (pile d'exponentielles).

Enfin, que dire de la taille des étiquettes si la fonction P est un prédicat, d'une certaine arité, exprimé par une formule logique d'une classe particulière (formule du premier ordre,

3. Il est clair que, connaissant n, les algorithmes et fonctions invoqués (comme Q_n, T_n, B, et D) sont des programmes de taille constante. Donc seuls n et x coûtent $\lceil \log n \rceil$ bits d'information.

monadique du second ordre, ...).

4.2.2 Vers de nouvelles applications

Je pense cependant qu'il est plus bénéfique, dans un futur proche, de se tourner vers les applications potentielles des P-schémas d'étiquetage. En effet, certaines fonctions P sont particulièrement pertinentes dans le calcul distribué.

Nous avons déjà vu que les DIST-schémas d'étiquetage (étiquetages de distance) peuvent servir à l'établissement d'une connexion à travers un réseau (cf. page 34). Les ADJ-schémas d'étiquetage (étiquetages d'adjacence) ont des applications en algorithmique des graphes. Par exemple, de nombreux problèmes algorithmiques sur les graphes d'intersections se trouvent simplifiés si le graphe est donné dans son modèle d'intersections (liste des intervalles dans le cas des graphes d'intervalles, ...). Étiqueter les sommets d'un graphe de la sorte revient à proposer un modèle pour ce graphe. Il s'agirait de trouver un compromis entre le pré-calcul des étiquettes et le calcul de la solution proprement dite. Ce paradigme a déjà été proposé dans [Rao99, LLR95, Spi00], où, pour approximer des problèmes NP-difficiles, on plonge la métrique du graphe dans un espace Euclidien de faible dimension en limitant la distorsion entre les métriques.

On pourrait proposer d'autres P-schémas d'étiquetage. Par exemple, Peleg dans un article récent [Pel00], étudie plusieurs problèmes d'étiquetage sur les graphes. Entre autres, il propose la fonction LCA(u,v) définie sur les arbres, comme le plus petit ancêtre commun de u et v, étant entendu que chaque sommet possède une identité propre non modifiable et fixée à l'avance sur $O(\log n)$ bits. Il propose un LCA-schéma d'étiquetage avec des étiquettes de taille $O(\log^2 n)$, et montre que c'est la meilleure borne possible.

On pourrait aussi considérer la fonction SP(x,y,z) valant 1 si z est situé sur un plus court chemin entre x et y, et valant 0 si non. Un SP-schéma d'étiquetage a des applications immédiates pour le routage compact. En effet, dans les tables de routage compactes utilisées par les méthodes de routage hiérarchiques (cf. paragraphes 2.2 page 19), on remplace souvent la liste des itinéraires pour tout un ensemble de sommets, par une seule route bien choisie. (Par exemple, pour router de x vers toute une composante donnée, on choisit de mémoriser la route vers un seul sommet du séparateur de cette composante. Généralement, on choisit le sommet le plus proche de x.) Cette méthode est efficace (en terme de longueur des routes et de facteur d'étirement) pour certains de ces sommets, mais apparaît moins bonne pour les autres. En pouvant tester des sommets potentiels intermédiaires, on pourrait, par un faible accroissement de la taille des adresses, améliorer la qualité des routes en diminuant le facteur d'étirement, sans augmenter significativement la taille des tables. (Dans le cas des séparateurs, on pourrait simplement stocker les adresses des sommets du séparateur, permettant ainsi de sélectionner le bon candidat grâce à la fonction SP.)

Une autre fonction, ayant des applications en terme d'optimisation du trafic dans les réseaux communiquants, est NBSP(x,y) qui donne le nombre de plus courts chemins arêtes-disjoints entre x et y, ou le poids d'une coupe minimale séparant x et y. Savoir calculer cette fonction avec des étiquettes courtes peut permettre d'optimiser les complexités (en temps ou en nombre de messages) de certains protocoles distribués (cf. [Pel00] pour une plus ample discussion).

4.2.3 Conclusion

Les schémas d'étiquetage permettent donc de mieux comprendre la contrainte de localité des structures de données dans les problèmes de graphes, et ils ont de nombreuses applications pour l'algorithmique distribuée.

Tous les problèmes que nous avons abordés consistent à optimiser des requêtes en lignes sur un graphe statique. Par exemple, étant donné x et y dans un arbre T, calculer le plus petit ancêtre commun $\textsc{lca}(x,y)$. La grande différence, par rapport aux problèmes "classiques" des requêtes en lignes, est que le calcul de la solution doit être effectué de manière locale. Se pose alors le problème de la distribution de la structure de données définissant T. Explicitons au moyen du même exemple : les requêtes $\textsc{lca}(x,y)$ peuvent être trivialement résolues en temps $O(n)$, si T est donné. Harel et Tarjan ont même montré dans [HT84] que ces requêtes pouvaient être satisfaites en temps $O(1)$ après un pré-traitement en temps et espace $O(n)$. Mais que se passe-t-il si nous n'avons pas accès à T dans son ensemble ? (Par exemple, on a accès seulement à une partie de l'information de T détenue par x et y.) Peut-on encore répondre à toutes les requêtes si, en quelques sortes, T est "distribué" sur tous les sommets ? C'est à cette problématique que nous comptons apporter des réponses.

Il semble que le problème LCA ne puisse être équitablement distribué. En effet, l'arbre T représente une information théorique d'au plus $O(n \log n)$ bits (la donnée de sa liste d'adjacences), alors que tout LCA-schéma d'étiquetage nécessite des étiquettes de taille $\Theta(\log^2 n)$, cf. [Pel00]. Nous arrivons tout naturellement à la notion de redondance de l'information contenue dans les étiquettes : on pourrait espérer des étiquettes de taille $O(\log n)$ (l'information maximale de l'arbre divisée par le nombre de sommets), et pourtant il en faut de taille $\Theta(\log^2 n)$.

Plus formellement, définissons pour toute fonction P et toute famille $\mathcal{F}_n$ de graphes à n sommets, la *redondance* de P pour $\mathcal{F}_n$ par :

$$\operatorname{red}(P,\mathcal{F}_n) \;=\; n \cdot \frac{\ell(P,\mathcal{F}_n)}{\log|\mathcal{F}_n|} \; . \tag{4.2}$$

Le terme $\log|\mathcal{F}_n|$ mesure l'information maximale d'un graphe de $\mathcal{F}_n$, alors que $\ell(P,\mathcal{F}_n)$ mesure la taille optimale d'un P-schéma d'étiquetage sur $\mathcal{F}_n$ (cf. l'équation (4.1) page 48). En effet, comme nous l'avons déjà mentionné, des étiquettes de taille $\log|\mathcal{F}_n|$ suffisent pour calculer P, étant donné que l'étiquette contient toute l'information du graphe. D'un autre coté, le mieux que l'on puisse espérer[4] est la distribution idéale des étiquettes, à savoir des étiquettes de taille $(\log|\mathcal{F}|)/n$.

4. Toute cette discussion sous-entend que la fonction P considérée est "normale". Il doit être possible de créer de manière artificielle des problèmes pour lesquels il n'est pas nécessaire d'avoir une information totale

D'autres définitions pour la redondance auraient pu être proposées. Par exemple, le rapport entre la somme, sur tous les sommets du graphe, des étiquettes et $\log|\mathcal{F}_n|$. On pourrait aussi choisir $\max_{G\in\mathcal{F}_n} C(G)$ comme dénominateur, où $C(G)$ représente la complexité de Kolmogorov du graphe G. Ce ne sont que des variantes la même notion, le coût de la répartition d'une structure données pour résoudre un problème de graphes.

La redondance définie par l'équation (4.2) varie entre $O(1)$ et $O(n)$, et peut permettre de classer les fonctions P entres elles, mais aussi les familles de graphes. Considérons les exemples suivants :

Soit $\mathcal{T}_n$ la famille des arbres à n sommets. Rappelons que $|\mathcal{T}_n| = n^{n-2}$, donc $\log|\mathcal{T}_n| = (n-2)\log n$. D'après ce qui précède et le chapitre 3, nous avons

$$\mathrm{red}(\textsc{lca}, \mathcal{T}_n) = \Theta(\log n) \quad \text{et} \quad \mathrm{red}(\textsc{dist}, \mathcal{T}_n) = \Theta(\log n) \ .$$

Cependant, pour l'adjacence,

$$\mathrm{red}(\textsc{adj}, \mathcal{T}_n) \ \leqslant \ 2 + o(1)$$

car nous avons vu que des étiquettes de tailles $2\log n$ suffisent toujours (cf. page 33).

Inversement, les familles de graphes s'ordonnent naturellement pour chaque problème P en fonction de leur redondance. Ainsi, page 36, nous avons vu que $\ell(\textsc{dist}, \mathcal{G}_n) \leqslant 1.58n$, où $\mathcal{G}_n$ représente la famille de tous les graphes à n sommets (il y en a $2^{\binom{n}{2}}$), donc :

$$\mathrm{red}(\textsc{dist}, \mathcal{G}_n) \leqslant 3.16 \quad \text{alors que} \quad \mathrm{red}(\textsc{dist}, \mathcal{T}_n) = \Theta(\log n) \ .$$

En quelque sorte, pour les requêtes de type DIST, il est plus difficile de distribuer et d'équilibrer la structure de données d'un arbre que celle d'un graphe général.

Enfin, la conjecture de la représentation implicite des graphes (rencontrée au chapitre 3 page 34) qui stipule que toute famille héréditaire de graphes à n sommets comportant $2^{O(n\log n)}$ membres possède un schéma d'étiquetage d'adjacence avec des étiquettes de taille $O(\log n)$, peut être résumée par l'équation :

$$\mathcal{F}_n \text{ héréditaire et } |\mathcal{F}_n| = 2^{O(n\log n)} \quad \Rightarrow \quad \mathrm{red}(\textsc{adj}, \mathcal{F}_n) \ = \ O(1) \ ?$$

D'ailleurs, on pourrait se demander si, pour des familles de plus grande cardinalité, on n'a pas de manière plus générale :

$$\mathcal{F}_n \text{ héréditaire} \quad \Rightarrow \quad \mathrm{red}(\textsc{adj}, \mathcal{F}_n) \ = \ O(1) \ ?$$

Je pense avoir résumé, par les notions d'étiquetage généralisé et de redondance, les problèmes fondamentaux inhérents aux structures de données compactes et distribuées.

$\log|\mathcal{F}_n|$ bits, et donc pour lesquels des étiquettes plus courtes que $(\log|\mathcal{F}_n|)/n$ bits sont envisageables. Ce n'est pas le cas des fonctions P que nous avons considérées jusqu'à présent.

Bibliographie

[AAPS87] Y. Afek, B. Awerbuch, S. A. Plotkin, and M. Saks, *Local management of a global resource in a communication network*, in 28^{th} Annual Symposium on Foundations of Computer Science (FOCS), IEEE Computer Society Press, October 1987, pp. 347–357.

[ABNLP89] B. Awerbuch, A. Bar-Noy, N. Linial, and D. Peleg, *Compact distributed data structures for adaptive routing*, CWI Quarterly, 2 (1989), pp. 277–305.

[ABNLP90] B. Awerbuch, A. Bar-Noy, N. Linial, and D. Peleg, *Improved routing strategies with succinct tables*, Journal of Algorithms, 11 (1990), pp. 307–341.

[ACK+98] B. Awerbuch, I. Cidon, S. Kutten, Y. Mansour, and D. Peleg, *Optimal broadcast with partial knowledge*, SIAM Journal on Computing, 28 (1998), pp. 511–524.

[AGPV90] B. Awerbuch, O. Goldreich, D. Peleg, and R. Vainish, *A trade-off between information and communication in broadcast protocols*, Journal of the ACM, 37 (1990), pp. 238–256.

[AGR89] Y. Afek, E. Gafni, and M. Ricklin, *Upper and lower bounds for routing schemes in dynamic networks*, in 30^{th} Annual Symposium on Foundations of Computer Science (FOCS), IEEE Computer Society Press, October 1989, pp. 370–375.

[Aho90] A. V. Aho, *Algorithms for finding patterns in strings*, in Handbook of Theoretical Computer Science, J. van Leeuwen, ed., vol. A, North-Holland, 1990, pp. 255–300.

[AKPW95] N. Alon, R. M. Karp, D. Peleg, and D. West, *A graph-theoretic game and its application to the k-serveur problem*, SIAM Journal on Computing, 24 (1995), pp. 78–100.

[AP90] B. Awerbuch and D. Peleg, *Sparse partitions*, in 31^{th} Symposium on Foundations of Computer Science (FOCS), IEEE Computer Society Press, 1990, pp. 503–513.

[AST90] N. Alon, P. D. Seymour, and R. Thomas, *A separator theorem for graphs with an excluded minor and its applications*, in 22^{nd} Symposium on Theory of Computing (STOC), May 1990, pp. 293–299.

[Bar98] Y. Bartal, *On approximating arbitrary metrics by tree metrics*, in 30^{th} Annual ACM Symposium on Theory of Computing (STOC), May 1998, pp. 161–168.

[BF67] M. A. Breuer and J. Folkman, *An unexpected result on coding the vertices of a graph*, Journal of Mathematical Analysis and Applications, 20 (1967), pp. 583–600.

[BFG$^+$00] L. Barrière, P. Fraigniaud, C. Gavoille, B. Mans, and J. M. Robson, *On recognizing cayley graphs*, in 8^{th} Annual European Symposium on Algorithms (ESA), vol. 1879 of Lectures Notes in Computer Science, Springer, September 2000, pp. 76–87.

[BHV99] H. Buhrman, J.-H. Hoepman, and P. Vitányi, *Space-efficient routing tables for almost all networks and the incompressibility method*, SIAM Journal on Computing, 28 (1999), pp. 1414–1432.

[BLTT97] H. L. Bodlaender, J. v. Leeuwen, R. B. Tan, and D. Thilikos, *On interval routing schemes and treewidth*, Information and Computation, 139 (1997), pp. 92–109.

[BLV98] H. Buhrman, M. Li, and P. Vitányi, *Kolmogorov random graphs and the incompressibility method*, in IEEE Conference on Compression and Complexity of Sequences 1997, IEEE Comp. Soc. Press, 1998, pp. 78–96.

[BM99] A. Brodnik and J. I. Munro, *Membership in constant time and almost-minimum space*, SIAM Journal on Computing, 28 (1999), pp. 1627–1640.

[Bod98] H. L. Bodlaender, *A partial k-arboretum of graphs with bounded treewidth*, Theoretical Computer Science, 209 (1998), pp. 1–45.

[Bol75] B. Bollobás, *Random Graphs*, Academic Press, New York, 1975.

[Bol78] B. Bollobás, *Extremal Graph Theory*, Academic Press, New York, 1978.

[Bou85] J. Bourgain, *On Lipschitz embeddings of finite metrics spaces in Hilbert spaces*, Israel J. Mathematics, (1985), pp. 46–52.

[Bre66] M. A. Breuer, *Coding the vertexes of a graph*, IEEE Trans. on Information Theory, IT-12 (1966), pp. 148–153.

[CFG99] J. Cohen, P. Fraigniaud, and C. Gavoille, *Recognizing bipartite incident-graphs of circulant digraphs*, in 25^{th} International Workshop, Graph - Theoretic Concepts in Computer Science (WG), P. Widmayer, G. Neyer, and S. Eidenbenz, eds., vol. 1665 of Lecture Notes in Computer Science, Springer, June 1999, pp. 215–227.

[COS97] D. G. Corneil, S. Olariu, and L. Stewart, *Asteroidal triple-free graphs*, SIAM Journal on Discrete Mathematics, 10 (1997), pp. 399–430.

[Cow99] L. J. Cowen, *Compact routing with minimum stretch*, in 10^{th} Symposium on Discrete Algorithms (SODA), ACM-SIAM, 1999, pp. 255–260.

[CR00] V. Chepoi and A. Rollin, *Interval routing in some planar networks.* Manuscript, 2000.

[CV00] V. Chepoi and Y. Vaxes, *On covering bridged plane triangulations with balls*. Manuscript submitted, 2000.

[CW00] L. J. Cowen and C. Wagner, *Compact roundtrip routing in directed networks*, in 19^{th} Annual ACM Symposium on Principles of Distributed Computing (PODC), ACM PRESS, July 2000, pp. 51–59.

[DKKP99] K. Diks, E. Kranakis, D. Krizanc, and A. Pelc, *The impact of knowledge on broadcasting time in radio networks*, in 7^{th} Annual European Symposium on Algorithms (ESA), J. Nešetřil, ed., vol. 1643 of Lectures Notes in Computer Science, Springer, July 1999, pp. 41–52.

[dlTNP98] P. de la Torre, L. Narayanan, and D. Peleg, *Thy neighbor's interval is greener: A proposal for exploiting interval routing schemes*, in 5^{th} International Colloquium on Structural Information & Communication Complexity (SIROCCO), L. Gargano and D. Peleg, eds., Carleton Scientific, 1998.

[dR94] J. de Rumeur, *Communications dans les réseaux de processeurs*, Études et Recherches en Informatique (ERI), MASSON, 1994.

[EGP98] T. Eilam, C. Gavoille, and D. Peleg, *Compact routing schemes with low stretch factor*, in 17^{th} Annual ACM Symposium on Principles of Distributed Computing (PODC), ACM PRESS, August 1998, pp. 11–20.

[EMZ97] T. Eilam, S. Moran, and S. Zaks, *The complexity of the characterization of networks supporting shortest-path interval routing*, in 4^{th} International Colloquium on Structural Information & Communication Complexity (SIROCCO), D. Krizanc and P. Widmayer, eds., Carleton Scientific, July 1997, pp. 99–111.

[EPTZ97] T. Eilam, D. Peleg, R. B. Tan, and S. Zaks, *Broadcast in linear messages in IRS representing all shortest paths*. Manuscript, 1997.

[FG98a] P. Fraigniaud and C. Gavoille, *Interval routing schemes*, Algorithmica, 21 (1998), pp. 155–182.

[FG98b] P. Fraigniaud and C. Gavoille, *A theoretical model for routing complexity*, in 5^{th} International Colloquium on Structural Information & Communication Complexity (SIROCCO), L. Gargano and D. Peleg, eds., Carleton Scientific, July 1998, pp. 98–113.

[FGM00] P. Fraigniaud, C. Gavoille, and B. Mans, *Interval routing schemes allow broadcasting with linear message-complexity*, Research Report LRI-1241, LRI, Université Paris-Sud, 91405 Orsay cedex, France, January 2000.

[FJ86] G. N. Frederickson and R. Janardan, *Separator-based strategies for efficient message routing*, in 27^{th} Symposium on Foundations of Computer Science (FOCS), IEEE Computer Society Press, May 1986, pp. 428–437.

[FJ88] G. N. Frederickson and R. Janardan, *Designing networks with compact routing tables*, Algorithmica, 3 (1988), pp. 171–190.

[FJ89] G. N. Frederickson and R. Janardan, *Efficient message routing in planar networks*, SIAM Journal on Computing, 18 (1989), pp. 843–857.

[FJ90] G. N. Fredrickson and R. Janardan, *Space-efficient message routing in c-decomposable networks*, SIAM Journal on Computing, 19 (1990), pp. 164–181.

[Fla97] M. Flammini, *On the hardness of devising interval routing schemes*, Parallel Processing Letters, 7 (1997), pp. 39–47.

[Fre93] G. N. Frederickson, *Searching among intervals and compact routing tables*, in 20^{th} International Colloquium on Automata, Languages and Programming (ICALP), A. Lingas, R. Karisson, and S. Carlsson, eds., vol. 700 of Lecture Notes in Computer Science, Springer-Verlag, July 1993, pp. 28–39.

[FvLMS95] M. Flammini, J. van Leeuwen, and A. Marchetti-Spaccamela, *The complexity of interval routing on random graphs*, in 20^{th} International Symposium on Mathematical Foundations of Computer Sciences (MFCS), J. Wiederman and P. Hájek, eds., vol. 969 of Lecture Notes in Computer Science, Springer-Verlag, August 1995, pp. 37–49.

[Gav96] C. Gavoille, *Lower bounds for interval routing on bounded degree networks*, Research Report RR-1144-96, LaBRI, University of Bordeaux, 351, cours de la Libération, 33405 Talence Cedex, France, October 1996. Submitted for publication.

[Gav97] C. Gavoille, *On the dilation of interval routing*, in 22^{nd} International Symposium on Mathematical Foundations of Computer Science (MFCS), I. Prívara and P. Ružička, eds., vol. 1300 of Lecture Notes in Computer Science, Springer, August 1997, pp. 259–268.

[Gav00a] C. Gavoille, *On the dilation of interval routing*, The Computer Journal, 43 (2000), pp. 1–7.

[Gav00b] C. Gavoille, *A survey on interval routing*, Theoretical Computer Science, 245 (2000), pp. 217–253.

[GG97] C. Gavoille and M. Gengler, *Space-efficiency of routing schemes of stretch factor three*, in 4^{th} International Colloquium on Structural Information & Communication Complexity (SIROCCO), D. Krizanc and P. Widmayer, eds., Carleton Scientific, July 1997, pp. 162–175.

[GG98] C. Gavoille and E. Guévremont, *Worst case bounds for shortest path interval routing*, Journal of Algorithms, 27 (1998), pp. 1–25.

[GH99] C. Gavoille and N. Hanusse, *Compact routing tables for graphs of bounded genus*, in 26^{th} International Colloquium on Automata, Languages and Programming (ICALP), J. Wiedermann, P. van Emde Boas, and M. Nielsen, eds., vol. 1644 of Lecture Notes in Computer Science, Springer, July 1999, pp. 351–360.

[GHS83] G. Gallager, P. A. Humblet, and P. M. Spira, *A distributed algorithm for minimal spanning tree*, ACM Trans. Programming Languages Systems, 30 (1983), pp. 66–77.

[GHT84] J. R. Gilbert, J. P. Hutchinson, and R. E. Tarjan, *A separation theorem for graphs of bounded genus*, Journal of Algorithms, 5 (1984), pp. 391–407.

[GKK+00] C. Gavoille, M. Katz, N. A. Katz, C. Paul, and D. Peleg, *Approximate distance labeling schemes*, Research Report RR-1250-00, LaBRI, University of Bordeaux, 351, cours de la Libération, 33405 Talence Cedex, France, December 2000.

[GL99] C. Gavoille and W. Laurencine, *Facteur d'étirement moyen et tables de routage de compacité 1*, stage de DEA, Université Bordeaux I, LaBRI, 351, cours de la Libération, 33405 Talence cedex, France, June 1999.

[Gol80] M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, Harcourt Brace Jovanovich, Academic Press ed., 1980.

[GP72] R. L. Graham and H. O. Pollak, *On embedding graphs in squashed cubes*, Lecture Notes in Mathematics, 303 (1972), pp. 99–110.

[GP96a] C. Gavoille and S. Pérennès, *Lower bounds for interval routing on 3-regular networks*, in 3^{rd} International Colloquium on Structural Information & Communication Complexity (SIROCCO), N. Santoro and P. Spirakis, eds., Carleton University Press, June 1996, pp. 88–103.

[GP96b] C. Gavoille and S. Pérennès, *Memory requirement for routing in distributed networks*, in 15^{th} Annual ACM Symposium on Principles of Distributed Computing (PODC), ACM PRESS, May 1996, pp. 125–133.

[GP98] C. Gavoille and D. Peleg, *The compactness of interval routing for almost all graphs*, in 12^{th} International Symposium on Distributed Computing (DISC), S. Kutten, ed., vol. 1499 of Lecture Notes in Computer Science, Springer, September 1998, pp. 161–174.

[GP99] C. Gavoille and D. Peleg, *The compactness of interval routing*, SIAM Journal on Discrete Mathematics, 12 (1999), pp. 459–473.

[GPPR01] C. Gavoille, D. Peleg, S. Pérennes, and R. Raz, *Distance labeling in graphs*, in 12^{th} Symposium on Discrete Algorithms (SODA), ACM-SIAM, January 2001, pp. 210–219.

[GR00] C. Gavoille and J. M. Robson, *On computing the least significant bit.* Manuscript, 2000.

[GRS00] C. Gavoille, A. Raspaud, and E. Sopena, *k-dominating set in planar graphs of diameter at most $2k$.* Manuscript in preparation, 2000.

[HP00] Y. Hassin and D. Peleg, *Sparse communication networks and efficient routing in the plane*, in 19^{th} Annual ACM Symposium on Principles of Distributed Computing (PODC), ACM PRESS, July 2000, pp. 41–59.

[HT84] D. Harel and R. E. Tarjan, *Fast algorithms for finding nearest common ancestors*, SIAM Journal on Computing, 13 (1984), pp. 338–355.

[KK77] L. Kleinrock and F. Kamoun, *Hierarchical routing for large networks; performance evaluation and optimization*, Computer Networks, 1 (1977), pp. 155–174.

[KKP00] M. Katz, N. A. Katz, and D. Peleg, *Distance labeling schemes for well-separated graph classes*, in 17^{th} Annual Symposium on Theoretical Aspects of

Computer Science (STACS), H. Reichel and S. Tison, eds., vol. 1770 of Lecture Notes in Computer Science, Springer Verlag, February 2000, pp. 516–528.

[KNR88] S. Kannan, M. Naor, and S. Rudich, *Implicit representation of graphs*, in 20^{th} Annual ACM Symposium on Theory of Computing (STOC), Chicago, IL, May 1988, pp. 334–343.

[KRŠ97] R. Kráľovič, P. Ružička, and D. Štefankovič, *The complexity of shortest path and dilation bounded interval routing*, in 3^{rd} International Euro-Par Conference, C. Lengaur, M. Griebl, and S. Gorlatch, eds., vol. 1300 of Lectures Notes in Computer Science, Springer, August 1997, pp. 258–265.

[LLR95] N. Linial, E. London, and Y. Rabinovich, *The geometry of graphs and some of its algorithmic applications*, Combinatorica, 15 (1995), pp. 215–245.

[LT79] R. J. Lipton and R. E. Tarjan, *A separator theorem for planar graphs*, SIAM Journal on Applied Mathematics, 36 (1979), pp. 177–189.

[LV97] M. Li and P. M. B. Vitányi, *An Introduction to Kolmogorov Complexity and its Applications (second edition)*, Springer-Verlag, 1997.

[Mal88] S. M. Malitz, *Genus g graphs have pagenumber $O(\sqrt{g})$*, in 29^{th} Symposium on Foundations of Computer Science (FOCS), IEEE Computer Society Press, October 1988, pp. 458–468.

[Man99] B. Mans, *On the interval routing of chordal rings of degree* 4, in International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN), IEEE, Zomaya, Hsu, Ibarra, Horiguchi, Nassimi, and Palis, eds., June 1999, pp. 16–21.

[Mil86] G. L. Miller, *Finding small simple cycle separators for* 2*-connected planar graphs*, Journal of Computer and System Sciences, (1986), pp. 265–279.

[MR97] J. I. Munro and V. Raman, *Succinct representation of balanced parentheses, static trees and planar graphs*, in 38^{th} Symposium on Foundations of Computer Science (FOCS), IEEE Computer Society Press, October 1997, pp. 118–126.

[MS96] G. MacGillivray and K. Seyffarth, *Domination numbers of planar graphs*, Journal of Graph Theory, 22 (1996), pp. 213–229.

[Mun96] J. I. Munro, *Tables*, in 16^{th} FST&TCS, vol. 1180 of Lectures Notes in Computer Science, Springer-Verlag, 1996, pp. 37–42.

[NN96] L. Narayanan and N. Nishimura, *Interval routing on k-trees*, in 3^{rd} International Colloquium on Structural Information & Communication Complexity (SIROCCO), N. Santoro and P. Spirakis, eds., Carleton University Press, June 1996, pp. 104–118.

[NS96] L. Narayanan and S. Shende, *Characterizations of networks supporting shortest-path interval labeling schemes*, in 3^{rd} International Colloquium on Structural Information & Communication Complexity (SIROCCO), N. Santoro and P. Spirakis, eds., Carleton University Press, June 1996, pp. 73–87.

[Pel99] D. Peleg, *Proximity-preserving labeling schemes and their applications*, in 25^{th} International Workshop, Graph - Theoretic Concepts in Computer

Science (WG), P. Widmayer, G. Neyer, and S. Eidenbenz, eds., vol. 1665 of Lecture Notes in Computer Science, Springer, June 1999, pp. 30–41.

[Pel00] D. Peleg, *Informative labeling schemes for graphs*, in 25^{th} International Symposium on Mathematical Foundations of Computer Science (MFCS), vol. 1893 of Lecture Notes in Computer Science, Springer, August 2000, pp. 579–588.

[PRS94] S. A. Plotkin, S. Rao, and W. D. Smith, *Shallow excluded minors and improved graph decomposition*, in 5^{th} Symposium on Discrete Algorithms (SODA), ACM-SIAM, 1994, pp. 462–470.

[PS85] D. K. Pradhan and M. R. Samathan, *The de Bruijn multiprocessor network: A versatile sorting network*, in IEEE Proceedings of the 12^{th} International Symposium on Computer Architecture, June 1985, pp. 360–367.

[PU88] D. Peleg and E. Upfal, *A tradeoff between space and efficiency for routing tables*, in 20^{th} Annual ACM Symposium on Theory of Computing (STOC), Chicago, IL, May 1988, pp. 43–52.

[PU89] D. Peleg and E. Upfal, *A trade-off between space and efficiency for routing tables*, Journal of the ACM, 36 (1989), pp. 510–530.

[Rao99] S. Rao, *Small distorbtion and volume preserving embeddings for planar and Euclidian metrics*, in Computational Geometry, 1999.

[RS86] N. Robertson and P. D. Seymour, *Graph minors. II. Algorithmic aspects of tree-width*, Journal of Algorithms, 7 (1986), pp. 309–322.

[SK85] N. Santoro and R. Khatib, *Labelling and implicit routing in networks*, The Computer Journal, 28 (1985), pp. 5–8.

[Spi00] J. Spinrad, *Efficient Graph Representations*, 2000. In preparation.

[TL97] S. S. H. Tse and F. C. M. Lau, *An optimal lower bound for interval routing in general networks*, in 4^{th} International Colloquium on Structural Information & Communication Complexity (SIROCCO), D. Krizanc and P. Widmayer, eds., Carleton Scientific, July 1997, pp. 112–124.

[TL99] S. S. H. Tse and F. C. M. Lau, *On the space requirement of interval routing*, IEEE Transactions on Computers, 48 (1999), pp. 752–757.

[vLT87] J. van Leeuwen and R. B. Tan, *Interval routing*, The Computer Journal, 30 (1987), pp. 298–307.

[Win83] P. Winkler, *Proof of the squashed cube conjecture*, Combinatorica, 3 (1983), pp. 135–139.

[Yan86] M. Yannakakis, *Four pages are necessary and sufficient for planar graphs*, in 18^{th} Annual ACM Symposium on Theory of Computing (STOC), 1986, pp. 104–108.