# Routage compact pour les réseaux de genre borné

Cyril Gavoille* Nicolas Hanusse

ALGOTEL '99

**Résumé**

Pour les réseaux planaires à $n$ noeuds, nous montrons comment construire en temps linéaire, des tables de routage de plus court chemin qui requiert $8n+o(n)$ bits par noeud, et $O(\log^{2+\epsilon} n)$ bit-operations par noeud pour extraire la route, pour toute constante $\epsilon > 0$. Nous généralisons ce résultat pour tout réseau ayant un nombre borné d'intersections d'arêtes. Nous étendons aussi notre résultat pour tout réseau de genre borné par $\gamma$, en construisant des tables de routage de plus court chemin de taille $n \log(\gamma+1) + O(n)$ bits, et avec $O(\log^{2+\epsilon} n)$ bit-operations par noeud pour extraire la route. Ce résultat est obtenu par l'utilisation d'ensembles dominants, du codage compact de partitions non croisées et de la représentation en $k$ pages de graphe.

## 1 Introduction

Dans un réseau de communication point à point ou dans les ordinateurs parallèles, une fonction de routage est utilisée pour envoyer des messages entre processeurs. Quand les réseaux grossissent, il devient important de réduire la quantité de mémoire contenue dans chaque noeud pour l'opération de routage. En même temps, il est essentiel d'acheminer les messages avec les chemins les plus courts possibles.

Un *schéma de routage universel* est un algorithme qui génère une fonction de routage pour tout réseau donné. Un type de schéma de routage universel trivial est basé sur des schémas dans lesquels chaque noeud contient une *table de routage* complète qui spécifie un port de sortie pour toute destination. Même si ce schéma peut garantir un routage selon les plus courts chemins, chaque routeur doit stocker locallement $n \log d$ bits de mémoire, où $d$ est le degré du routeur (c.a.d. le nombre de ports de sortie) et $n$ est le nombre de noeuds du réseau. Cependant, ce schéma est impraticable quand on traite de grands réseaux.

Une *fonction de routage* $R$ est un algorithme distribué dont le rôle consiste à envoyer des messages entre les noeuds d'un réseau. Plus précisément, la fonction de routage $R$ utilise une paire de fonctions entières $(P, H)$ où $P$ est la *fonction port* et $H$ est la *fonction d'en-tête*. Pour chaque paire de noeuds distincts $u$ et $v$, $R$ produit un chemin $(u = u_0, u_1, \ldots, u_k = v)$, une séquence $h_0, h_1, \ldots, h_k$ d'en-tête, et une séquence $p_0, p_1, \ldots, p_k$ de numéro de ports. Un message avec l'en-tête $h_i$ arrivant à un noeud $u_i$ à travers le port d'entrée $p_i$ génère un nouvel

*LaBRI, Université Bordeaux I, 351, cours de la Libération, 33405 Talence Cedex, France. e-mail : {*gavoille,hanusse*}*@labri.u-bordeaux.fr*

en-tête $H(u_i, p_i, h_i) = h_{i+1}$, et est transmis à un port de sortie $P(u_i, p_i, h_i) = p_{i+1}$. Ainsi, pour chaque $i \in \{1, \ldots, k-1\}$, nous avons $H(u_i, p_i, h_i) = h_{i+1}$, $P(u_i, p_i, h_i) = p_{i+1}$ et chaque $p_{i+1}$ correspondant au numéro de port de sortie du lien $(u_i, u_{i+1})$. Sur chaque routeur, le port numéroté 0 correspond au lien spécial entre le routeur et son hôte. Nous complétons la description en imposant que $p_0 = 0$, $P(u_k, p_k, h_k) = 0$, $h_0 = v$, en fixant l'en-tête initial. Comme nous sommes intéressés par les schémas de routage distribués, nous exprimerons le protocole de délivrance comme une collection de fonctions de routage locales $R_u = (P_u, H_u)$, une pour chaque noeud $u$. Notre objectif est de trouver des schémas de routage de plus court chemin de telle manière que les fonctions de routage générées, $R_u$, peuvent être codées avec un espace mémoire minimum et que la route peut être extraite de manière très rapide.

Dans ce papier, nous considérons uniquement les schémas de tables de routage, c'est-à-dire les schémas qui génèrent des fonctions de routage qui ne changent pas l'en-tête selon la route, qui ne dépendent que de la destination. Ceci entraîne que $h_0 = h_1 = ... = h_k = v_0$ (où $v_0$ est le noeud destination), et que la fonction $P$ ne dépend pas du port d'entrée $p_i$. Les noms des noeuds appartiennent à l'ensemble $\{1, \ldots, n\}$, et les numéros de port d'un noeud $u$ sont dans $\{1, \ldots, d\}$ où $d$ est le degré de $u$. Le modèle général est déja été utilisé dans [PU89] et [FG97] pour prouver des bornes inférieures sur la quantité de mémoire nécessaire au routage.

La topologie sous-jacente du réseau est représentée par un graphe. Les graphes considérés sont simples, non orientés et connexes. Pour diminuer la taille des tables de routage, nous autorisons de renuméroter les noeuds, en gardant l'ensemble des adresses dans l'ensemble $\{1, \ldots, n\}$.

Pour éviter l'implémentation de trop grandes tables de routage, il est possible de coder dans chaque noeud la carte complète du réseau. Ceci permet d'avoir une structure de données assez petite si le graphe sous-jacent peut être décrit avec un faible nombre de bits. Cependant, cette *approche centralisée* possède l'inconvénient suivant : nous devons déterminer la route avec un algorithme de plus court chemin qui prend au moins un temps linéaire par rapport au nombre de noeuds. Malgré les hypothèses standards des systèmes distribués qui supposent que le calcul local est négligeable devant le coût de communication, en pratique ce n'est pas nécessairement vrai, précisément pour l'approche centralisée. Etant donné la topoplogie du graphe, nous avons besoin d'un temps $\Omega(m)$ pour calculer le plus court chemin, où $m$ est le nombre d'arêtes du réseau. Fondamentalement, un tel algorithme de routage calcule le processus de routage en temps $\Omega(n)$, où $n$ est le nombre de noeuds du réseau, alors que la taille de l'entrée (l'adresse du noeud) est de taille $O(\log n)$ bits. Donc, la complexité en temps d'une telle approche est exponentielle.

L'algorithme de plus court chemin n'est pas nécessairement la faiblesse de l'approche centralisée. Il existe un deuxième inconvénient pour les réseaux qui utilisent une représentation compacte de leur topologie. Par exemple, dans [KK96], il est montré que tout graphe de Cayley (une large classe de réseaux d'interconnexion incluant : l'hypercube, les anneaux, les tores, shuffle-exchange, butterfly, ...) qui définit un routage de plus court chemin ne nécessitant que $O(\log^3 n)$ bits seulement. Ce résultat est basé sur la représentation des groupes abéliens, qui peut être fait de manière compacte car les formules énumératives de tels groupes montrent que peu de graphes à $n$ sommets sont des graphes de Cayley de groupes abéliens. Cependant, dans [KK96], aucune borne de temps n'est donné pour extraire la route. Coder un groupe peut être fait avec un faible nombre de bits, alors que le temps pour trouver le plus court chemin, c.a.d. la plus petite décomposition d'un élément en fonction de ses générateurs, est assez difficile sans la construction du graphe dans son entier. Plus générallement, il est montré dans [FG98] qu'il existe des fonctions de routage qui peuvent être codées en $O(\log n)$ bits à chaque routeur (signifiant que la complexité de Kolmogorov des fonctions de routage est $O(\log n)$ bits seulement) telles que toute implémentation de telles fonctions en moins de $n - o(n)$ bits requiert un temps et un espace pour calculer la fonction plus grand que toute pile de puissance de 2 de taille constante, c.a.d.

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

Une approche plus pratique consisterait à stocker une vue partielle de la topologie du réseau, et calculerait la route en temps polynomial par rapport à la taille de l'entrée, c.a.d. en temps $\log^{O(1)} n$.

Concernant les schémas de routage distribués, il est montré dans [GP96] que $\Omega(n \log d)$ bits sont nécessaires, alors que $n \log d$ sont suffisant pour l'utilisation de tables de routage standards. Plus précisément, cette borne inférieure optimale signifie qu'il existe des graphes à $n$ noeuds, de degré maximum $d$, pour tout $3 \leqslant d \leqslant \epsilon n$ et toute constante $\epsilon < 1$, tels que pour toute numérotation des ports et des noeuds, et quelque soit la fonction de routage de plus court chemin, la taille de toute implémentation de cette fonction est de taille au moins $\Omega(n \log d)$ bits. De plus, il est montré que cette borne est valable sur $\Theta(n)$ noeuds du graphe.

## 2 Graphes planaires et graphes de genre borné

Bien qu'il existe des asymptotiques du nombre de graphes dans le cas général [GP96], il n'existe pas de bornes fines dans le cas des graphes planaires. En utilisant une approche centralisée, $O(n)$ bits sont suffisants, car il est bien connu que le nombre de graphes planaires connexes non numérotés à $n$ noeuds est borné par $c^{n+o(n)}$, pour $c$ constant. Donc, nous pouvons théoriquement déterminer tout graphe planaire avec un index de taille $n \log c + o(n)$ bits . La meilleure formule énumérative pour les graphes planaires non numérotés à $n$ noeuds, nombre noté $\mathcal{U}(n)$, montre qu'il existe une constante $c$ telle que (voir [DVW96]) :

$$\lim_{n \to +\infty} \mathcal{U}(n)^{1/n} \; = \; c \; ,$$

où $3.24 < \log c < 6.25$. Donc, asymptotiquement, les tables de taille $n \log c < 6.25n$ bits sont suffisantes pour l'approche centralisée dans le cas des graphes planaires. Le problème est que la formule énumérative d'un ensemble d'objets ne donne de méthodes efficaces pour construire rapidement un objet. Le temps d'extraction de la route est à priori exponentiel (au moins la cardinalité de l'ensemble, $c^{n+o(n)}$). Une meilleure solution consiste à utiliser un codage compact et efficace des graphes planaires. Le codage de Jacobson [Jac89] permet les requêtes d'adjacence en ne parcourant que $O(\log n)$ bits, mais avec $36n$ bits de codage. Récemment [CGH$^+$98] a été amélioré le codage précédent à $4m/3 + 5n + o(n)$ bits, c.a.d. , $9n + o(n)$ bits pour les graphes planaires denses avec un temps $O(1)$ pour le test d'adjacence. Ce résultat implique des tables de routage de taille $O(n)$ bits par noeud avec un temps $\Omega(n)$ pour extraire la route (pour chaque source, on doit exécuter un algorithme de plus court chemin sur tout le graphe). On peut noter que dans le cas des graphes planaires, les chemins les plus courts à partir d'une source peuvent être calculés en temps $O(n)$ [KRRS94] au lieu de $O(n \log n)$ pour l'algorithme de Dijkstra. Dans le meilleur cas, ceci mène à des tables de routage compact de taille $9n + o(n)$ bits avec un temps $O(n)$ pour extraire la route. De plus, on remarque que le temps est toujours exponentiel par rapport à la taille de l'entrée car les adresses des destinations sont codées avec $\log n$ bits.

En utilisant spécifiquement une approche de routage compact, Frederickson et Janardan ont montré dans [FJ88] que tout graphe $p$-planaire de genre $\gamma$ peut utiliser des tables de routage de plus court chemin de *compacité* au plus $3p/2 + \gamma$. — Un graphe $p$-planaire est un graphe qui peut être dessiné de telle manière que les noeuds peuvent être couvert par au plus $p$ faces. Par exemple, les graphes planaires extèrieurs sont des graphes 1-planaire de genre 0 — Ceci signifie que les tables de routage satisfont la propriété que l'ensemble des adresses qu'utilise une arête orientée donnée consiste en au plus $\lfloor 3p/2 + \gamma \rfloor$ intervalles d'entiers consécutifs (modulo $n$). Les tables de routage de compacité $k$ sont aussi appelées *$k$-routage par intervalle*, et ont été introduit par van Leeuwen et Tan [vLT87]. Une telle representation implique des tables de routage pour les graphes $p$-planaires de taille $O(d(p + \gamma) \log n)$ bits par noeud de degré $d$ avec un temps $O(\log n)$ pour extraire la route (cf. [Gav98] pour une synthèse récente sur cette technique). Malheureusement, le paramètre $p$ d'un graphe peut être $\Theta(n)$, même pour les graphes de genre $\gamma = 0$. Finallement, aucun schéma de routage de plus court chemin n'est connu pour les graphes planaires utilisant moins de $n \log d$ bits en espace ($n \log n$ dans le plus mauvais cas) et un temps poly-logarithmique.

## 3 Resultats

Nous montrons comment router en temps poly-logarithmique en utilisant des tables de taille au plus $8n + o(n)$ bits par noeud. La même borne est valable pour les graphes non planaires qui sont planaires après avoir enlevé $o(n/\log n)$ arêtes. Nous proposons une optimisation avec des tables de taille $9n - 8d + o(n)$ bits pour les noeuds de degré $d$, quand $d > n/8$. De plus, le temps pour construire les tables est linéaire pour chaque noeud, ce qui est optimal. Notons que notre borne en espace est plus petite que tous les codages connus pour les graphes planaires qui supporte un temps *raisonnable* pour les requêtes d'adjacence[1] (cf. [CGH+98]). Cependant notre approche n'est pas centralisée. Voir Table 1 pour un résumé.

Nous généralisons notre résultat aux graphes de genre borné par $\gamma$, $\gamma \geqslant 0$, en construisant des tables de routage compact de taille $n \log(\gamma + 1) + O(n)$ bits avec un temps poly-logarithmique pour extraire la route la plus courte. Dans le plus mauvais cas, $\gamma = O(n^2)$ et la taille devient du même ordre que les tables de routage standards, c.a.d., $O(n \log n)$ bits. Cette nouvelle mesure sur les informations de routage montre clairement que le plongement et la représentation d'un graphe "aide" pour le routage.

## Références

[CGH+98] R. C.-N. Chuang, A. Garg, X. He, M.-Y. Kao, and H.-I. Lu, *Compact encodings*

[1]On peut restreindre notre attention aux codages qui supporte un temps poly-logarithmique pour les requêtes d'adjacence .

| Table de routage taille en bits | Temps pour extraire la route | Commentaires |
|---|---|---|
| $n \log n$ | $O(\log n)$ | tables de routage standards (en bit-operations) |
| $6.25n$ | $2^{n^{O(1)}}$ | borne supérieure théorique [Tut62] |
| $9n + o(n)$ | $O(n)$ | codage compact des graphes planaires non numérotés avec un temps $O(1)$ pour les requêtes d'adjacence [CGH$^+$98] et algorithme de plus court chemin [KRRS94] |
| $8n + o(n)$ | $O(\log^{2+\epsilon} n)$, $\epsilon > 0$ | tables de routage compact (en bit-operations) [**Ce papier**] |

TAB. 1: Tables de routage compact pour les graphes planaires de $n$ noeuds.


*of planar graphs via canonical orderings and multiple parentheses*, in ICALP'98, K. Guldstrand Larsen, S. Skyum, and G. Winskel, eds., vol. 1443 of Lecture Notes in Computer Science, Springer, July 1998, pp. 1–12.

[DVW96] A. Denise, M. Vasconcellos, and D. Welsh, *The random planar graph*, Congressus Numerantium, 113 (1996), pp. 61–79.

[FG97] P. Fraigniaud and C. Gavoille, *Universal routing schemes*, Journal of Distributed Computing, 10 (1997), pp. 65–78.

[FG98] P. Fraigniaud and C. Gavoille, *A theoretical model for routing complexity*, in SIROCCO'98, Carleton Scientific, July 1998, pp. 98–113.

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

[Gav98] C. Gavoille, *A survey on interval routing*, Theoretical Computer Science, (1998). To appear.

[GP96] C. Gavoille and S. Pérennès, *Memory requirement for routing in distributed networks*, in PODC'96, ACM PRESS, ed., May 1996, pp. 125–133.

[Jac89] G. Jacobson, *Space-efficient static trees and graphs*, in FOCS'89, IEEE Computer Society Press, October 1989, pp. 549–554.

[KK96] E. Kranakis and D. Krizanc, *Boolean routing on Cayley networks*, in SIROCCO'96, N. Santoro and P. Spirakis, eds., Carleton University Press, June 1996, pp. 119–124.

[KRRS94] P. Klein, S. Rao, M. Rauch, and S. Subramanian, *Faster shortest-path algorithms for planar graphs*, in STOC'94, 1994, pp. 27–37.

[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.

[Tut62] W. T. Tutte, *A census of planar triangulations*, Canadian Journal of Mathematics, 14 (1962), pp. 21–38.

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