Groupage sur un chemin orienté

Séverine Petat [†] and Marie-Emilie Voge [‡]

Projet Mascotte, I3S-CNRS/INRIA/Université de Nice-Sophia Antipolis, 2004 Route des Lucioles - B.P. 93 - 06902 Sophia Antipolis Cedex, France

Nous présentons plusieurs approches, heuristiques, programmation linéaire mixte et en nombres entiers et décomposition de graphes, pour étudier un problème de groupage sur un chemin (orienté). On dispose d'un chemin et d'un ensemble quelconque de requêtes unitaires. L'objectif est l'installation d'un ensemble minimum de tubes (arcs joignant deux sommets du chemin) de capacité C, permettant d'acheminer toutes les requêtes. Ce problème est un problème de dimensionnement de réseaux qui reste NP-difficile malgré ses hypothèses restrictives.

Keywords: groupage, routage, network design, GMPLS, WDM

1 Introduction

Le groupage de trafic (ou grooming) est un principe qui consiste à agréger des flux dans des flux de débit supérieur. On retrouve cette idée en particulier dans les réseaux GMPLS [IET] et WDM [DR02, BCM03].

En termes de graphes, les entrées du problème de groupage sont modélisées par un graphe support G muni de capacités, un ensemble de requêtes R de débit donné entre des paires de sommets de G, un routage de ces demandes sur G et un graphe "virtuel", le graphe des tubes disponibles, muni de coûts fixes et de capacités. Un tube représente une connexion entre deux sommets de G, c'est à dire un chemin fixé entre ces deux sommets. Une requête peut entrer (respectivement sortir) dans un tube uniquement au sommet origine (respectivement destination) du tube.

Grouper les demandes dans des tubes équivaut à trouver un ensemble de tubes tel que le chemin emprunté par chaque requête se décompose en une succession de tubes. Le groupage doit respecter des contraintes de capacités à deux niveaux différents. D'une part, la somme des capacités des tubes passant sur une arête du graphe support ne doit pas dépasser la capacité de cette arête. D'autre part, la somme des débits des requêtes empruntant un tube doit être inférieure à la capacité de ce tube.

Plusieurs critères d'optimisation ont été étudiés, comme la minimisation du nombre d'ADM (Add Drop Multiplexer) dans [BCM03], la minimisation des capacités totales ou maximales utilisées sur le réseau support, etc. Ici nous voulons minimiser le nombre de tubes nécessaires avec un chemin pour graphe support. Ce critère est celui considéré dans [BDPS03] où le graphe support est un anneau.

Nos hypothèses permettent de classer le groupage dans les problèmes de dimensionnement de réseau (Network Design). La littérature sur le Network Design est abondante et les résolutions proposées sont souvent basées sur une approche polyédrale comme dans [BG96, KM01], cependant des algorithmes d'approximation et des heuristiques ont aussi été étudiés en particulier dans [GMM95].

Nous définissons le groupage sur le chemin précisement en section 2.1. Nous avons abordé ce problème sous différents angles, comme la recherche de briques élémentaires (analogue à [BDPS03]) qui ne sera pas développée ici, ou la programmation linéaire en nombres entiers, ou enfin en proposant des heuristiques (section 3.2) dont certains résultats expérimentaux sont présentés en section 4.

2 Problème du groupage sur un chemin

2.1 Définition du problème

Le problème de groupage, en général, requiert la donnée de trois éléments :

[†] recherche soutenue par le CRC CORSO avec France Telecom et le projet européen CRESCO
[‡] ACI Sécurité PRESTO avec CNRS

Un graphe support Ici, le graphe support considéré $G = (V,A)$ est un chemin orienté composé de n sommets numérotés de 0 à $(n-1)$, $V = \{0,1,2,...,(n-1)\}$ et des arcs $A = \{(i,i+1) \ \forall \ 0 \leq i < n-1\}$. Pour simplifier, on supposera que la capacité des arcs est infinie, c'est-à-dire qu'autant de tubes que nécessaire peuvent emprunter un arc donné.

Un ensemble de requêtes R R est un ensemble de paires ordonnées de sommets de G. Ici, les requêtes sont de la forme (i,j) avec $i < j$ et $i,j \in V$. Pour simplifier, nous considérons des requêtes simples et unitaires, c'est à dire qu'il existe au plus une demande de valeur 1 entre deux sommets donnés.

Un ensemble de tubes disponibles T Tous les tubes de la forme (i,j) avec $i < j$ et $i,j \in V$ sont autorisés en autant d'exemplaires que nécessaire. En revanche, leur capacité C, ou facteur de groupage, est uniforme et fait partie des données. Au plus C requêtes peuvent donc emprunter le même tube. Le coût d'un tube est unitaire et fixe, il ne dépend pas du nombre de requêtes qui l'utilisent. En d'autres termes le coût d'un ensemble de tubes est son cardinal.

On souhaite installer un ensemble de tubes de cardinal minimum, sur le graphe support, constituant un groupage réalisable des demandes. Pour illustrer nos propos, la figure 1 montre deux groupages différents pour un graphe des requêtes donné, dans le cas où le graphe support est un chemin.

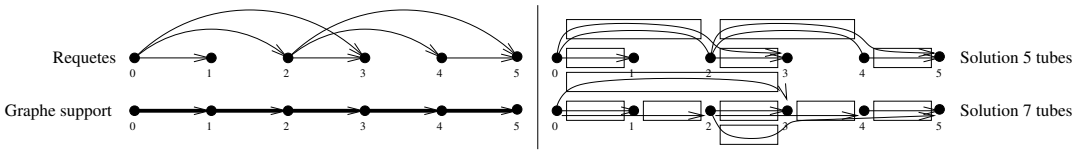


FIG. 1: Illustration du problème de groupage sur un chemin pour $C = 2$

Le groupage sur le chemin et sur l'anneau diffèrent uniquement par leurs graphes support. Les résultats de [BDPS03], comme la NP-complétude et les bornes sur le nombre de tubes optimal, n'utilisent pas les propriétés du graphe support et sont donc valables pour le chemin. En l'occurence, le nombre de tubes optimal est majoré par le nombre de requêtes R et minoré par $\frac{2R}{C+1}$.

2.2 Programmation linéaire en nombres entiers

Le groupage sur le chemin s'interprète comme un problème de dimensionnement de réseau consistant, d'une façon générale, à trouver un ensemble d'arcs de coût minimum, nos tubes, entre des sommets connus pour router des requêtes données.

Ainsi, pour le groupage sur le chemin, deux formulations classiques en programmation linéaire en nombres entiers existent. Bien que la formulation arcs-chemins comporte un nombre exponentiel de variables, elle est plus efficace que la formulation compacte sommets-arcs (1), grâce à la génération de colonnes.

Nous avons utilisé le modèle sommets-arcs suivant pour calculer le nombre de tubes optimal et le modèle arcs-chemins avec un ensemble restreint de chemins pour en obtenir rapidement une borne supérieure.

Soient $T = (V,A_T)$ le graphe des tubes disponibles, R l'ensemble des requêtes, x_a le nombre d'exemplaires du tube a utilisés et f_a^{st} le flot associé à la demande (st) circulant sur un exemplaire du tube a. $\delta^+(s)$ (resp. $\delta^-(s)$) est l'ensemble des tubes sortant (resp. entrant) du sommet s.

$$
\left\{
\begin{array}{lll}
Min & \displaystyle\sum_{a \in A_T} x_a & \\
t.q. & \displaystyle\sum_{a \in \delta^+(u)} f_a^{st} - \sum_{a \in \delta^-(u)} f_a^{st} = \left\{ \begin{array}{llll} 0 & si & u \neq s, \ u \neq t & \\ 1 & si & u = s & \\ -1 & si & u = t & \end{array} \right. & \forall u \in V, \ \forall (s,t) \in R \subseteq V^2 \\
 & \displaystyle\sum_{(st) \in R} f_a^{st} \leq C * x_a & \forall a \in A_T \\
 & f_a^{st} \in \{0,1\} & \forall (s,t) \in R, \ \forall a \in A_T \\
 & x_a \in IN & \forall a \in A_T .
\end{array}
\right. \quad (1)
$$

Nous avons formulé la conjecture suivante, qui permettrait de remplacer ce modèle en nombres entiers par un programme mixte (MIP) en relâchant l'intégrité du flot, et ainsi de le résoudre plus rapidement.

Conjecture 2.1 *Le nombre de tubes optimal est identique, que le flot soit réel ou entier.*

Aucun des tests effectués n'a invalidé la conjecture, cependant il existe une instance pour laquelle la solution du programme relaxé n'est pas réalisable pour le programme entier. Pour la rendre réalisable, permuter le flot entre les requêtes (les rerouter) ne suffit pas, il faut en effet modifier l'ensemble de tubes.

2.3 Heuristiques

Nous proposons deux heuristiques gloutonnes pour résoudre le groupage sur le chemin.

Heuristique 1 La i^{eme} itération de cette heuristique consiste à ajouter aux tubes choisis précédemment un ensemble de tubes permettant de router toutes les requêtes de longueur i : si les tubes existants ne permettent pas de router une requête, alors le tube le plus court possible[§] permettant de la router est ajouté.

Prenons l'exemple 1 avec $C = 2$. La première itération ajoute un tube de longueur 1 pour chaque requête de longueur 1, i.e. les tubes $(0,1)$[¶], $(2,3)$, $(4,5)$. Dans chacun de ces tubes il reste la place de router une autre requête. A la deuxième itération, les tubes placés ne suffisent pas à router les requêtes $(0,2)$ et $(2,4)$, il faut donc ajouter de nouveaux tubes. Pour router la requête $(0,2)$ on peut ajouter soit le tube $(0,2)$, soit le tube $(1,2)$, l'heuristique choisit $(1,2)$ car c'est le plus court des deux. De même on ajoute $(3,4)$ qui permet avec le tube $(2,3)$ déjà placé de router la requête $(2,4)$. Ainsi le tube $(0,1)$ est complet car il contient les requêtes $(0,1)$ et $(0,2)$. De même pour le tube $(2,3)$. A la troisième itération, on obient la solution en 7 tubes donnée à la figure 1.

En pratique cette heuristique donne de bons résultats, mais une requête peut emprunter beaucoup de petits tubes, comme la requête $(2,5)$ dans l'exemple qui emprunte les trois tubes $(2,3)$, $(3,4)$ et $(4,5)$. Or dans un réseau de télécommunication réel, le respect de la qualité de service impose souvent un nombre de sauts maximum pour une route. Le principe de la deuxième heuristique permet de remédier à cet inconvénient.

Heuristique 2 L'idée de la deuxième heuristique est de regrouper une longue requête (i, j) avec des requêtes formant un plus court chemin[‖] entre i et j dans le graphe des requêtes privé de (i, j). Un tube est ajouté lorsqu'entre deux sommets C requêtes ont pu être regroupées ou qu'aucun regroupement n'est possible. Dans l'exemple 1 avec $C = 2$, il n'existe qu'un chemin, $(0,2)$ et $(2,3)$, entre les sommets 0 et 3 pour la requête $(0,3)$. Entre 0 et 2 les requêtes $(0,2)$ et $(0,3)$ sont donc regroupées, le tube $(0,2)$ est ajouté, de même pour $(2,3)$.

Si pour une requête on ne trouve pas de chemin, l'heuristique 2 la décompose en deux requêtes telles qu'il existe un chemin, le plus long possible en nombre d'arcs, pour l'une des deux seulement.

Supposons que la requête $(1,5)$ existe dans l'exemple 1. Il n'y a pas de chemin entre 1 et 5 mais il en existe entre 2 et 5 ($(2,5)$ ou $(2,4)$ et $(4,5)$) et aussi entre 4 et 5. Deux décompositions sont possibles, soit $(1,2)$ et $(2,5)$ car il existe au moins un chemin de 2 à 5 et aucun de 1 à 2, soit $(1,4)$ et $(4,5)$. L'heuristique choisit la première car la requête $(2,5)$ issue de cette décomposition est plus longue que $(4,5)$.

L'heuristique 2 appliquée à l'exemple 1 donne la solution en 5 tubes.

3 Résultats

Nous avons comparé les méthodes de résolution évoquées précédemment, les heuristiques (H1, H2), les programmes en nombres entiers sommets-arcs (NAI, programme 1) et arcs-chemins (API) et le programme mixte sommets-arcs (NAF). Pour la formulation arcs-chemins nous n'avons généré qu'un ensemble restreint de chemins (variables du programme) par requête, le nombre de tubes fourni par le solveur Ilog Cplex est donc une borne supérieure.

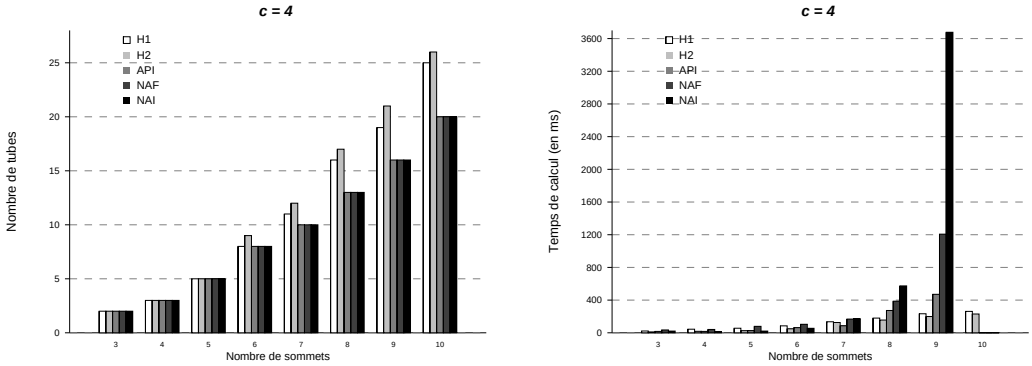
Nous avons effectué des tests pour différents facteurs de groupage sur des graphes de requêtes complets, i.e. pour tous sommets i, j du graphe support tels que $i < j$ la demande (ij) existe.

[§] Le plus court tube en nombre d'arcs.

[¶] La première coordonnée indique le sommet origine du tube et la seconde indique le sommet destination du tube.

[‖] Le plus court chemin en nombre de requêtes traversées.

Les graphiques suivants présentent le nombre de tubes obtenu par les différentes méthodes ainsi que les temps de calcul associés en millisecondes pour $C = 4$. Pour dix sommets les temps de calculs nécessaires aux résolutions des trois programmes mixtes dépassent l'heure et n'apparaissent pas sur le graphique.



Notons que la conjecture est vérifiée pour ces exemples, les valeurs obtenues par les programmes NAI et NAF sont identiques. Par contre, pour certains cas le calcul est plus rapide avec les variables de flots entières qu'avec les variables de flots réelles. Cependant la différence n'est pas significative et ne remet pas en cause l'intérêt de la relaxation NAF. Elle s'avère très intéressante en particulier pour un graphe à dix sommets et un facteur de groupage $C = 2$ puisque le temps de calcul passe de plus de 8h30 pour NAI à un peu plus de 55 minutes pour la relaxation NAF. Ces résultats montrent aussi que les heuristiques sont satisfaisantes car elles fournissent très rapidement des solutions proches de l'optimal.

4 Conclusion

Le problème de groupage que nous avons étudié, bien que fortement simplifié, reste difficile et long à résoudre pour des graphes dépassant dix sommets. D'où l'intérêt de nos heuristiques, qui fournissent très rapidement des solutions proches de l'optimal.

Beaucoup reste à faire sur ce problème, en particulier la démonstration de la conjecture et des tests supplémentaires des heuristiques sur des instances de grande taille. Le travail déjà effectué sur les briques élémentaires nous donne une infinité d'instances qui nous permettrons d'effectuer ces tests.

Enfin, le problème pour des graphes supports différents, comme les arbres reste à explorer.

Références

[BCM03] J.-C. Bermond, D. Coudert, and X. Munoz. Traffic grooming in unidirectional wdm ring networks : the all-to-all unitary case. In *ONDM*, pages 1135–1153, 2003.

[BDPS03] J.-C. Bermond, O. DeRivoyre, S. Pérennes, and M. Syska. Groupage par tubes. In *Conference ALGOTEL2003, Banyuls, May 2003*, pages 169–174, 2003.

[BG96] D. Bienstock and O. Günlück. Capacited network design-polyhedral structure and computation. *Informs journal on Computing*, 8, 1996.

[DR02] R. Dutta and G. N. Rouskas. Traffic grooming in wdm networks : past and future. *IEEE Networks*, 16(6) :46–56, november/december 2002.

[GMM95] M. Grötschel, C.L. Monma, and M.Stoer. *Handbooks in Operations Research and Management Science*, volume 7 :Network Models. Elsevier publisher, 1995.

[IET] IETF. *http ://www.ietf.org/internet-drafts/draft-ietf-ccamp-gmpls-architecture-07.txt*.

[KM01] H. Kérivin and A.R. Mahjoub. On survivable network polyhedra. *submitted to Discrete Mathematics*, 2001.