Une approche en programmation par contraintes pour le dimensionnement de réseaux

Diego Olivier Fernandez Pons [1,2]

[1] Université Pierre et Marie Curie - Paris VI, 5 place Jussieu 75005 Paris
[2] ILOG S.A, 9 rue de Verdun 94253 Gentilly Cedex
diego-olivier.fernandez-pons@cicrp.jussieu.fr dofpons@ilog.fr

Résumé

Nous présentons une méthode de programmation par contraintes pour résoudre un problème de dimensionnement de réseaux avec routage des demandes par des chemins. Cette approche a été choisie car la contrainte de monoroutage semblait perturber considérablement les méthodes basées sur la programmation linéaire en nombres entiers. Nous avons été dès lors conduits à chercher des relaxations sous forme de problèmes de rangement (*packing*) et à étudier leur interaction avec la structure de graphe et d'union de chemins du problème initial.

Abstract

We present a constraint programming method to solve a network design problem with communication routing by a single path. This technique was chosen because the unsplittable-flow constraint was perturbing linear programming methods. We relax the network design problem into a packing problem and study its interaction with the graph structure and the integer multicommodity-flow problem.

1 Introduction

Les projets RNRT Rococo et RNTL Fado, en collaboration avec France Télécom et le laboratoire PRiSM de l'université de Versailles Saint-Quentin se proposaient d'étudier un problème de dimensionnement de réseaux dont les particularités résidaient en ce que les coûts sont des fonctions en escalier, le routage des demandes doit se faire par des chemins et non des flots, enfin, des contraintes supplémentaires portant sur la topologie du réseau ou sur les routages des demandes doivent pouvoir être intégrées au problème (*unsplittable multi-commodity network flow with step increasing cost functions and side-constraints*). Le temps de résolution imparti est de 10 minutes. En outre, les méthodes de résolution robustes lors de l'ajout de contraintes inconnues doivent être préférées à des méthodes qui par trop spécialisées verraient leur performances se dégrader considérablement.

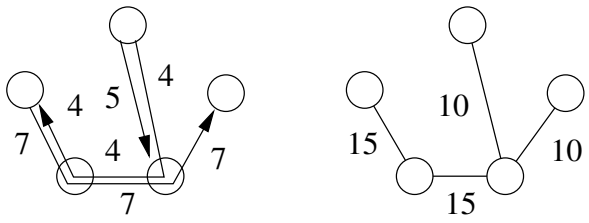

FIG. 1 – Routage des demandes et liens dans un réseau de télécommunications

Certains éléments de cette conjonction de requêtes ont été abordés dans des études précédentes, notamment l'utilisation de fonctions de coût en escalier [8], l'ajout de contraintes sur les routages [10] ou encore le routage par des chemins [1]. Par ailleurs, les problèmes de routage de télécommunications dans lesquels le temps imparti est réduit sont en général abordés par des heuristiques éventuellement basées sur des méthodes exactes, que ce soit la programmation par contraintes [13] ou la programmation linéaire [5]. Aucune étude cependant n'aborde tous ces aspects conjointement. De plus, la technique la plus populaire pour le dimensionnement de réseaux reste la programmation linéaire en nombres entiers mais la programmation par contraintes semble plus adaptée à la nécessité de robustesse vis-à-vis de contraintes en partie

arbitraires et à la limitation de temps.

Le choix des méthodes de résolution s'est donc naturellement porté sur la programmation linéaire en nombres entiers (PLNE) et la programmation par contraintes (PPC).

2 Description du problème

Le problème qui nous intéresse consiste à choisir pour tout arc (i,j) d'un graphe G un type de lien $y_{ij} \in [0 \dots k]$ parmi un nombre fixe de possibilités chacune étant caractérisée par un coût, une capacité maximale et éventuellement d'autres propriétés selon la variante du problème choisie. Toutes les demandes du problème doivent pouvoir être routées simultanément dans le réseau construit, chaque route étant un chemin reliant la source s_d et la destination t_d, éventuellement soumis à des contraintes additionnelles. Chaque demande d a un volume Vol_d et ce volume s'additionne avec celui des autres demandes dans chaque arc qu'elle emprunte. La variable $x_{ij}^d \in \{0,1\}$ indique si la demande d emprunte l'arc (i,j). L'objectif du problème est de minimiser le coût de construction du réseau. Le problème peut être décrit par le programme mathématique schématique suivant :

$$\min \sum_{ij \in G} \mathsf{Cout}\,[y_{ij}]$$

$$\forall ij \in G,\ \sum_d \mathsf{Vol}_d x_{ij}^d \ \leq\ \mathsf{Capacite}\,[y_{ij}] \qquad (1)$$

$$\forall d,\ (x_{ij}^d)_{ij \in G} \text{ chemin entre } s^d \text{et } t^d \qquad (2)$$

$$\text{contraintes supplementaires} \qquad (3)$$

$$x_{ij}^d \in \{0,1\} \quad y_{ij} \in [0 \dots k]$$

Modèle arc-noeud

Les contraintes supplémentaires se classent en trois catégories :

1. contraintes topologiques : s'exprimant sur les variables y_{ij} uniquement, elles imposent une certaine topologie au réseau par exemple une limite sur le degré des noeuds du réseau ou encore être un arbre.
2. contraintes sur les routages : s'exprimant sur les variables x_{ij}^d uniquement, elles imposent des conditions aux routes par lesquelles circulent les demandes, par exemple une longueur maximale, un flot maximal en certains noeuds, etc.
3. contraintes couplantes : elles portent à la fois sur les variables y_{ij} et sur les variables x_{ij}^d, par exemple certaines demandes dites "sécurisées" ne peuvent circuler que dans des arcs munis de liens sécurisés également.

Le propos du projet n'est pas d'espérer résoudre tout problème de dimensionnement de réseaux muni de contraintes supplémentaires quelconques, mais de permettre une certaine flexibilité dans la définition du problème avec une augmentation modérée du temps de calcul pourvu que les contraintes supplémentaires ne destructurent pas trop le problème de base.

Les données complètes et les résultats du projet Rococo se trouvent à l'adresse `www.prism.uvsq.fr/Rococo`. Il s'agit de 21 graphes quasi-complets (à la fois liens possibles et demandes) allant de 4 à 25 noeuds. Dans le projet Rococo, 6 contraintes optionnelles étaient prédéfinies. Dans le projet Fado, ces contraintes ne sont que des exemples de contraintes qui pourraient survenir. On peut donc considérer les données de Rococo comme 2^6 instances du problème Fado.

3 Les difficultés de la résolution par programmation linéaire

Les approches par programmation linéaire en nombres entiers rencontrent des difficultés liées à l'intégrité, la connexité, les bornes inférieures et les contraintes supplémentaires.

Les difficultés de connexité sont illustrées par le fait qu'avec le modèle noeud-arc correspondant aux équations (1−3), l'utilisation du moteur de programmation par contraintes ILOG Solver [12] muni d'une stratégie ne prenant nullement en compte la fonction coût mais fixant à chaque fois un chemin complet obtient de meilleurs résultats que le branchement par défaut du solveur linéaire ILOG Cplex [11]. La raison en est qu'elle évite les problèmes de connexité et de convergence d'un multiflot (relaxation linéaire) vers un ensemble de chemins. Cplex au contraire fixe les arcs des demandes de façon éparpillée, créant de la sorte des sous-problèmes infaisables par la seule nécessité de construire un ensemble de chemins entre noeuds du graphe et non un ensemble de flots.

problème	Cplex	Solver
A04	3 s	0 s
A05	16 s	0 s
A06	515 s	35 s

TAB. 1 – Illustration des difficultés posées par la connexité. Le branchement de Cplex est basé sur l'impact d'une décision sur le coût, celui de Solver uniquement l'impératif de fixer un chemin de bout en bout avant de considérer tout autre décision.

La mauvaise qualité des bornes inférieures est due à la double relaxation présente dans le problème : en

effet, il est déjà difficile de borner précisément la capacité minimale $sum_{ij}\mathsf{Capacite}\,[y_{ij}]$ que va requérir le réseau or la borne sur le coût y est liée de façon complexe (via la fonction de coût en escalier). Ainsi, en supposant que l'on route les demandes par des flots et que l'on dispose de bornes exactes de capacité pour les 2^n sous-ensembles d'arcs du réseau, le calcul du coût minimal correspondant revient à résoudre un problème de multi-sac-à-dos lequel est notoirement difficile.

Les difficultés rencontrées par le modèle PLNE noeud-arc avec les contraintes supplémentaires, détaillées par Le Pape [16], sont dues au fait que la réduction de l'espace de recherche par les contraintes supplémentaires n'est pas correctement répercutée sur la borne inférieure ou les variables de décision. Ainsi, la contrainte limitant la longueur des chemins à 3 arcs contre 6 autrement réduit considérablement la taille de l'espace de recherche dans le problème à 7 noeuds A07. Pourtant, le temps de résolution avec Cplex s'en trouve considérablement augmenté.

$$\min \sum_{ij \in G} \mathsf{Cout}\,[y_{ij}]$$

$$\forall ij \in G,\ \sum_{d} \sum_{p|ij \in p} \mathsf{Vol}_d x^p \ \leq\ \mathsf{Capacite}\,[y_{ij}] \qquad (4)$$

$$\forall d,\ \sum_{p:s^d \to t^d} x^p = 1 \qquad (5)$$

$$x^p \in \{0,1\} \quad y_{ij} \in [0 \ldots k]$$

Modèle noeud-chemin

Afin de pallier l'absence de solution réalisable après 10 minutes, ont été étudiées des heuristiques de recherche locale pour les problèmes linéaires en nombres entiers généraux tels le local branching ou le RINS [6].

Une approche de génération de colonnes [16] a permis de réduire quelque peu les problèmes de connexité. En effet, dans le modèle noeud-chemin transcrit par les équations (4 - 5) les variables x^p ne sont plus des arcs mais des chemins complets, si bien que tout branchement de Cplex se traduit par le choix d'un chemin. De plus, la prise en compte de certaines contraintes supplémentaires portant sur chaque chemin individuellement - par exemple la limitation en longueur - s'en trouve simplifiée puisqu'il suffit de ne pas générer de chemin violant les contraintes voulues. Cependant, les problèmes d'intégrité ne sont pas pour autant résolus et Cplex a tendance à fractionner les demandes afin de profiter d'un chemin intéressant du point de vue du coût mais ne pouvant router l'intégralité du volume de la demande.

L'approche par branch-cut-and-price [3] ajoute à la génération de colonnes des inégalités de bipartition destinées à mieux borner la capacité minimale du réseau dans une optique semblable à celles utilisée pour le dimensionnement de réseau avec coûts linéaires à charge fixe [4].

Toutes ces méthodes ont été couplées à des stratégies de recherche dédiées.

Indépendamment Menne, Grötschel et Möhring [14] ont mené une étude polyédrique du problème, exhibant plusieurs familles d'inégalités valides. Les coupes de *subtour* s'efforcent d'améliorer la prise en compte de la connexité, celles de bipartition et de *general upper bounding* à améliorer les bornes sur la capacité du réseau. Enfin certaines inégalités sont spécifiquement dédiées aux 6 contraintes du problème Rococo. Il reportent des difficultés semblables avec le modèle arc-noeud même après l'adjonction de ces coupes et l'utilisation de stratégies de branchement dédiées.

4 Résolution par méthodes PPC

Les méthodes de programmation par contraintes ne sont pas concernées par les problèmes de convergence puisqu'elles agissent directement sur des valeurs entières. Les problèmes de connexité peuvent être contournés par une stratégie de branchement consistant à n'affecter que des chemins de bout en bout. Et la fonction objectif peut être prise en considération dans la stratégie de branchement en routant les demandes par exemple par les chemins de plus petit coût marginal [16].

En revanche, leur faiblesse considérable réside en l'absence de toute borne inférieure. Cela conduit à une courbe d'évolution par paliers caractéristique (figure 2) : la stratégie de recherche ne parvenant pas à prouver la sous-optimalité d'un sous-arbre, il faut patienter le temps d'examiner quasiment toutes les combinaisons possibles car le retour en arrière se produit très bas dans l'arbre, lorsque presque toute la solution est fixée. La situation se dégrade avec l'accroissement de la taille des instances.

Une approche par recherche à grands voisinages (*large neighbourg search* - LNS) permet de contourner partiellement l'absence de borne inférieure : la recherche arborescente n'est menée que sur une partie du problème, le reste ayant été préalablement fixé à partir d'une solution connue ; quand la recherche ne semble plus progresser on la déplace sur une autre partie du problème.

La méthode par LNS [17] détient les résultats les plus robustes. Depuis sa mise en place, les améliorations de bornes supérieures sont demeurées rares : Cambazard [2] annonce n'avoir amélioré aucune so-

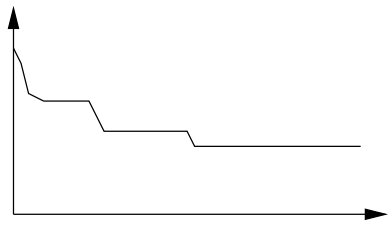

FIG. 2 – Courbe d'évolution du modèle PPC : des solution réalisables sont très rapidement trouvées mais l'amélioration est de plus en plus lente car le solveur ne parvenant pas à prouver la sous-optimalité de certaines branches de l'arbre de recherche, il faut les explorer quasi-intégralement.

lution sur 1344, Gendron 1 solution (communication personnelle), Menne 2 solutions [14] (une dizaine de solutions sur l'instance C16 sont en attente de confirmation). Enfin, une méthode de PPC arborescente avec un calcul incrémental très rapide des plus courts chemins et une limitation du nombre de chemins explorés par noeud a permis l'amélioration d'une quinzaine de solutions sur la série test A.

Nos efforts se sont donc désormais concentrés non sur l'amélioration pragmatique des résultats mais sur l'étude de la structure du problème avec en vue une amélioration des bornes inférieures.

4.1 Hybridation avec un modèle linéaire

L'utilisation d'une relaxation linéaire en tout noeud de l'arbre de recherche PPC a été essayée par Cambazard [2] et s'est montrée trop gourmande en temps de calcul pour être compétitive avec les méthodes déjà citées. Ces résultats sont cohérents avec les expériences menées au sein d'ILOG qui introduisaient certains aspects de la programmation par contraintes dans les méthodes PLNE que ce soit par des stratégies de branchement dédiées ou le renforcement de bornes par des algorithmes spécialisés. Cambazard signale également des résultats insuffisant en utilisant une borne inférieure par relaxation lagrangienne et en utilisant les coûts réduits calculés par le programme linéaire comme heuristique de branchement.

4.2 Bornes sur la capacité minimale du réseau

L'étude d'une borne inférieure basée sur des algorithmes combinatoires nous a donc paru incontournable. La relaxation linéaire mélange dans le calcul de la borne inférieure les arguments de coût et ceux de capacité. La contrepartie d'une borne inférieure combinatoire est de ne plus pouvoir procéder ainsi, du moins dans un premier temps.

A défaut de pouvoir borner inférieurement le coût, nous nous sommes concentrés sur l'amélioration de la capacité minimale du réseau ou de ses parties, dans le même esprit que les inégalités métriques ou les importantes inégalités de bipartition utilisées par les travaux de programmation linéaire [4] [8].

Dans la mesure où il ne semble pas raisonnable de relaxer les contraintes d'intégrité, nous nous sommes tournés vers des relaxations obtenues en "oubliant" une partie des contraintes du problème. Ainsi l'oubli de la composante de routage du dimensionnement de réseaux conduit à un problème de rangement (*packing*) où seule la dimension capacitaire doit être prise en considération ; inversement, l'oubli de la composante capacitaire du problème conduit à un problème de chemins et d'accessibilité dans un graphe.

Ces deux composantes doivent ensuite être reliées afin de coopérer dans la réduction de la combinatoire du problème.

4.3 Relaxation vers un problème de rangement

En oubliant la structure de graphe du problème et en fixant la longueur de tout chemin à une constante L_d on obtient un problème de rangement avec des boîtes de taille variable (l'indice ij est remplacé par b et $y_{ij} \in [0 \dots k]$ devient $y_b^k \in \{0,1\}$) :

$$\min \sum_b \sum_k \mathsf{Cout}_b^k y_b^k$$

$$\forall b,\ \sum_k y_b^k = 1 \tag{6}$$

$$\forall d,\ \sum_b x_b^d = \mathsf{L}_d \tag{7}$$

$$\forall b,\ \sum_d \mathsf{Vol}_d x_b^d \ \leq\ \sum_k \mathsf{Capacite}_b^k y_b^k \tag{8}$$

$$y_b^k \in \{0,1\} \quad x_b^d \in \mathbb{N}$$

L'étude à la fois en programmation linéaire en nombres entiers et en programmation par contraintes de ce problème a souligné l'importance des contraintes redondantes suivantes :

1. sac-à-dos agrégé : la capacité totale de l'ensemble des boîtes doit être supérieure à la somme des capacités d'éléments à disposer. En sommant (8) sur b on obtient

$$\sum_b \sum_d \mathsf{Vol}_d x_b^d \ \leq\ \sum_b \sum_k \mathsf{Capacite}_b^k y_b^k$$

$$\rightsquigarrow \sum_b \sum_k \mathsf{Capacite}_b^k y_b^k \ \geq\ \sum_d \mathsf{Vol}_d \mathsf{L}_d \tag{9}$$

2. variables de gaspillage : on introduit la variable d'écart (*slack*) ω_b pour chaque inégalité de sac-à-dos (8) que l'on répercute sur leur somme (9)

$$\sum_b \sum_k \mathsf{Capacite}_b^k y_b^k \geq \sum_d \mathsf{Vol}_d \mathsf{L}_d$$
$$\rightsquigarrow \sum_b \sum_k \mathsf{Capacite}_b^k y_b^k = \sum_d \mathsf{Vol}_d \mathsf{L}_d + \sum_b \omega_b \quad (9 \text{ bis})$$

avec

$$\forall b, \sum_d \mathsf{Vol}_d x_b^d + \omega_b = \sum_k \mathsf{Capacite}_b^k y_b^k \quad (8 \text{ bis})$$

Ces contraintes redondantes permettent sur les sous-problèmes de rangement correspondants des gains d'un facteur proche de $\times 1000$ lorsque résolus avec Cplex. Les gains avec Solver sont également très élevés mais le modèle naïf (6-8) est dès le départ très lent en raison de l'absence de toute borne inférieure.

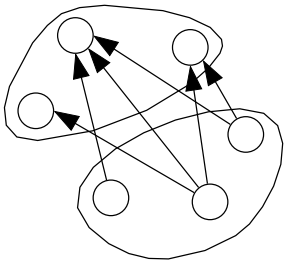

FIG. 3 – Demandes traversant la coupe $(S, \overline{S})$. La capacité transversale totale doit être supérieure au flot transversal.

La contrainte de sac-à-dos agrégé (9) n'est qu'un cas particulier de contrainte capacitaire obtenue en éliminant les variables x par sommation. Le même procédé est à l'origine des inégalités de biparition : toute coupe $(S, \overline{S})$ du graphe doit être traversée au moins une fois par les demandes reliant un point s^d de S à un point t^d de $\overline{S}$ (figure 3). En sommant les inégalités de sac-à-dos (8) sur les arcs de la coupe on obtient exactement les inégalités de bipartition

$$\sum_{ij \in (S,\overline{S})} \sum_d \mathsf{Vol}^d x_{ij}^d \leq \sum_{(i,j)\in(S,\overline{S})} \mathsf{Capacite}[y_{ij}]$$
$$\rightsquigarrow \sum_{(i,j)\in(S,\overline{S})} \mathsf{Capacite}[y_{ij}] \geq \sum_{(s^d,t^d)\in(S,\overline{S})} \mathsf{Vol}^d \quad (10)$$

Nous avons utilisé les inégalités de bipartition sur l'ensemble des arcs entrant et sortant de tout noeud. La différence avec l'équation (10) utilisée en programmation linéaire est que nous ne simplifions pas le membre droit de façon à ne conserver que la constante $\sum_d \mathsf{Vol}^d$ mais gardons les variables x_{ij} ce qui permet à l'inégalité de se renforcer au fur et à mesure que les branchements ou la propagation ajoutent des demandes entrant dans le noeud.

4.4 Impact de la longueur des chemins sur la capacité minimale

Nous montrerons d'abord que parmi les informations contenues dans le problème de routage, les bornes sur la longueur des chemins semble être celle dont l'impact sur la borne inférieure de capacité est le plus important.

Dans le problème de dimensionnement de réseaux, les longueurs des chemins L_d sont des variables et non des constantes. Or la longueur des chemins contraint fortement la capacité minimale du réseau donc indirectement son coût. En effet, si les chemins sont très longs, alors en vertu du terme $\sum_d \mathsf{Vol}_d L_d$ de l'équation (9 bis) le réseau a besoin d'une capacité importante pour router toutes les demandes, d'où un coût élevé. Inversement, si les chemins sont très courts, il y a peu de marge pour regrouper les demandes et le terme $\sum_b \omega_b$ domine l'équation (9 bis) ; par exemple si on impose que les chemins soient de longueur 1, alors chaque demande d doit être routée par le seul arc $s^d \rightarrow t^d$ et on gaspille nécessairement la différence entre le volume de la demande et la plus petite capacité pouvant la contenir. La figure 4 schématise la variation de la capacité minimale du réseau en fonction de la longueur des chemins.

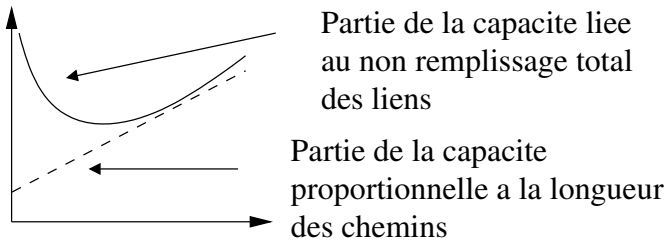

FIG. 4 – Capacité du réseau en fonction de la longueur des chemins

Ainsi, une borne précise sur la capacité minimale du réseau requiert donc des bornes sur la longueur des chemins. C'est pourquoi nous avons relié la longueur L_d de chaque chemin aux variables x_{ij}^d par l'équation (11) qui impose que si une demande d emprunte l'arc $i \rightarrow j$ elle est au moins aussi longue que le plus court chemin entre s^d et t^d passant par $i \rightarrow j$ dans le graphe

(x^d) des arcs que peut emprunter d.

$$\forall d,\ \forall ij \in G,\ [x^d_{ij} = 1] \Rightarrow$$
$$[L_d \geq \text{dist}_{x^d}(s^d, i) + 1 + \text{dist}_{x^d}(j, t^d)] \quad (11)$$

L'équation (12) est le penchant de (11) sur les variables y_{ij}.

$$\forall d,\ L_d \geq \text{dist}_y(s^d, t^d) \quad (12)$$

Tandis que (11) et sa contraposée relient L_d aux contraintes sur les routages, par exemple le fait que certains noeuds soient interdits à une demande pour ne pas excéder leur limite de trafic ou la longueur maximale des chemins, (12) s'efforce de capturer des informations issues de la topologie du réseau.

4.5 Génération de colonnes en programmation par contraintes

La stratégie de recherche actuelle fixe les chemins pour les demandes de bout en bout en s'efforçant d'instancier en priorité les chemins qui font croître le moins possible le coût global de la solution. Un algorithme de k plus courts chemins [7] est utilisé sur un graphe valué par une estimation du surcoût provoqué par l'utilisation de chaque arc $i \rightarrow j$ individuellement (obtenue par une méthode d'échantillonage dans laquelle chaque variable x^d_{ij} est successivement fixée à 1 et la variation de coût servant comme poids). La stratégie de recherche génère les chemins dans un ordre semblable au *limited discrepancy search* (LDS) [9] étendu au cas n-aire. On remarque que l'on retrouve ainsi le procédé de génération de colonnes de Dantzig et Wolfe (1961) utilisé en programmation linéaire : d'abord le problème où chaque demande peut être routée par les k meilleurs chemins est résolu, puis k est incrémenté. Ces résultats sont proches de ceux de Milano et van Hoeve [15] mais dans le cas du dimensionnement de réseaux l'interprétation en est beaucoup plus naturelle : Comme le surcoût du choix d'un plus court chemin est exact (localement), l'algorithme de tri par coûts réduits croissants de Milano et van Hoeve s'interprète aisément par la génération en ordre de coût croissant des chemins. Quant aux sous-problèmes générés par le LDS, ils correspondent bien à la résolution du problème de dimensionnement en limitant les chemins possibles à l'ensemble des k-meilleurs chemins. En d'autres termes, LDS apparaît comme une procédure de génération et résolution de sous-problèmes parce que le domaines des variables considérées est exponentiel (les chemins entre deux noeuds d'un graphe) donc représenté implicitement.

5 Conclusion et perspectives

L'introduction des bornes inférieures améliore les résultats de l'approche PPC pour les petits problèmes par rapport au modèle naïf mais le lien entre la capacité minimale du réseau et le coût d'une solution demeure trop ténu si bien que les bornes sur la capacité ne se traduisent pas par une augmentation suffisante des bornes inférieures sur le coût pour contrebalancer l'augmentation de temps induite. Ainsi, les meilleurs algorithmes continuent d'être basés sur une forte composante heuristique et contournent l'absence de bornes inférieures par des sauts aléatoires dans l'espace de recherche. Les efforts actuels consistent à renforcer la relaxation par des inégalités capacitaires. Cependant, le sous-problème de multi-sac-à-dos reliant la capacité du réseau et son coût n'a pas encore reçu de traitement satisfaisant.

Références

[1] Cynthia Barnhart, Christopher A. Hane, and Pamela H. Vance. Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. *Operations Research*, 48(2) :318–326, March-April 2000.

[2] Hadrien Cambazard. Conception de réseaux et optimisation combinatoire : étude d'un problème de multiflot monorouté sous contraintes opérationnelles. Rapport de DEA Institut de Recherches en Informatique de Nantes, septembre 2003.

[3] Alain Chabrier, Emilie Danna, Claude Le Pape, and Laurent Perron. Solving a network design problem. *Annals of Operations Research*, 130 :217–239, 2004.

[4] Mervat Chouman, Teodor Gabriel Crainic, and Bernard Gendron. A cutting-plane algorithm based on cutset inequalities for multicommodity capacitated fixed charge network design. Technical Report CRT-2003-16, Centre de recherche sur les transports, Université de Montréal, 2003.

[5] David Coudert and Hervé Rivano. Routage optique dans les réseaux WDM multifibres avec conversion partielle. In *AlgoTel'02*, pages 17–24, 2002.

[6] Emilie Danna, Edward Rothberg, and Claude Le Pape. Exploring relaxation induced neighborhoods to improve MIP solutions. *Mathematical Programming*, 102(1) :71–90, 2005.

[7] David Eppstein. Finding the k shortest paths. *SIAM J. Computing*, 28(2) :652–673, 1998.

[8] Virginie Gabrel, Arnaud Knippel, and Michel Minoux. Exact solution of multicommodity network optimization problems with general step cost functions. *Operations Research Letters*, 25 :15–23, 1999.

[9] William D. Harvey and Matthew L. Ginsberg. Limited discrepancy search. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, volume 1, 1995.

[10] Kaj Holmberg and Di Yuan. A multicommodity network-flow problem with side constraints on paths solved by column generation. *INFORMS Journal on Computing*, 15(1) :42–57, 2003.

[11] ILOG. *ILOG Cplex 10.0 User Manual*, 2006.

[12] ILOG. *ILOG Solver 6.2 User Manual*, 2006.

[13] Muriel Lauvergne. *Réservation de connexions avec reroutage pour les réseaux ATM : une approche hybride par programmation par contraintes.* PhD thesis, Ecole des Mines de Nantes, mars 2002.

[14] Ulrich Menne. LP approaches to survivable networks with single path routing Diploma Theses Technische-Universität Berlin. Supervisor : Prof. Dr. Martin Grötschel, Prof. Dr. Rolf Möhring, decembre 2003.

[15] Michela Milano and Willem Jan van Hoeve. Reduced cost based ranking for generating promising subproblems. In *8th International conference on principles and practice of constraint programming (CP)*, pages 1–16, 2002.

[16] Claude Le Pape, Laurent Perron, Jean-Charles Régin, and Paul Shaw. Robust and parallel solving of a network design problem. In *8th International conference on principles and practice of constraint programming (CP)*, pages 633–648, 2002.

[17] Laurent Perron. Fast restart policies and large neighborhood search. In *International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR)*, 2003.