Approximation combinatoire de multiflot fractionnaire : améliorations

M. Bouklit[1], D. Coudert[2], J.-F. Lalande[2] et H. Rivano[2*]

1 : LIRMM
2 : Projet MASCOTTE*, CNRS, I3S & INRIA Sophia Antipolis.*
bouklit@lirmm.fr, {David.Coudert,Jean-Francois.Lalande,Herve.Rivano}@sophia.inria.fr
[*] France Télécom Recherche & Développement Sophia Antipolis.
Ce travail a été partiellement financé par le projet européen CRESCCO et l'action COLOR DYNAMIC.

Motivés par la recherche d'algorithmes performants de dimensionnement de réseaux optiques WDM, nous considérons les $(1+\varepsilon)$-approximations du calcul de multiflot fractionnaire. Nous proposons des améliorations d'un algorithme de la littérature en utilisant des calculs de plus courts chemins dynamiques, éventuellement spécialisés au cas du routage optique dans les réseaux WDM multifibres sans conversion.

Keywords: Multiflot, approximation, réseau optique, WDM, routage optique, arrondi aléatoire, plus courts chemins dynamiques.

---

1 Introduction

Les réseaux à fibres optiques déployant les technologies de multiplexage en longueurs d'onde (WDM) apparaissent, grâce à une bande passante inégalée, comme la solution la plus adaptée aux réseaux d'infrastructure. Toutefois le déploiement de ce type de réseaux demande de gigantesques investissements et ne se fera pas sans l'assurance de performances du système. Cette assurance s'obtient, notamment, par un dimensionnement efficace du réseau. Ce problème est largement considéré par la communauté scientifique, d'un point de vue théorique ou pratique et dans différents contextes et modèles [Kum98, MS00, LS00, KS01, Riv01, CR02, FPR$^+$03].

Dans la suite, nous considérons un réseau multifibre d'infrastructure déployant la technologie WDM avec des équipements de conversion limitée en longueur d'onde. Le dimensionnement d'un tel réseau pour une instance de communication s'obtient en minimisant les ressources (nombre de fibres sur chaque lien, nombre de longueurs d'onde disponibles, nombre de conversions en longueur d'onde autorisées à chaque noeud) nécessaires à l'existence d'un routage optique de l'instance de communication. Dans un travail précédent, nous avons montré que ce problème se modélisait par un *multiflot* dans un graphe auxiliaire [CR02].

L'objet de cet article est de proposer des améliorations de l'algorithme d'approximation combinatoire du multiflot fractionnaire proposé dans [Fle00]. Cet algorithme a vocation à être utilisé comme la routine principale d'un algorithme d'approximation aléatoire du multiflot entier.

Le multiflot Le problème du multiflot est utile dans un grand nombre d'applications, notamment lorsqu'il est question de calculer des routes pour des entités qui sont en concurrence pour certaines ressources modélisées par des capacités sur les arcs du graphe support des routages possibles.

De manière formelle, un *réseau de flot* est un graphe $G=(V,E)$ avec des capacités $c(e)$ sur ses arcs, les arcs seront les supports du flot, les sommets seront des points intermédiaires ou bien des sources ou destinations du flot. On se donne un multi-ensemble de *commodités* $C=\{(s_i,t_i),\ s_i,t_i\in V,\ i=1,\ldots,k\}$. Un *flot* f_i de s_i à t_i est une pondération des arêtes de E respectant des *contraintes de conservation de flot* [FF62]. L'*intensité* du flot est la quantité de flot quittant la source s_i, qui et égale à celle arrivant à la destination t_i. Un *multiflot* de C est une réunion de flots pour chaque commodités respectant les capacités de G. Il s'agit donc d'un ensemble de pondérations sur E de sorte que $\forall e\in G,\ \sum_i f_i(e)\leq c(e)$.

Si les pondérations sont contraintes à être entières, le problème est $\mathcal{NP}$-difficile et non approximable en général. Cependant, Raghavan [Rag94] a donné un algorithme aléatoire de construction de multiflot basé sur un arrondi aléatoire de la relaxation linéaire du problème, c'est à dire le multiflot fractionnaire. Il s'agit du même problème mais où les pondérations sont réelles positives. Ce problème de *multiflot fractionnaire* se résout en temps polynomial puisqu'il s'écrit comme un programme linéaire avec un nombre polynomial de variables et de contraintes. Cela dit, les temps de calcul et l'espace mémoire nécessaires à la résolution de ce problème peuvent devenir très grand à mesure que G et C augmentent : la complexité en moyenne du simplex sur un tel programme linéaire est de l'ordre de $O\left(|C|^3|E|^2|V| + |C|^2|E|^3\right)$ [BT97]. Plus récemment, une amélioration de l'algorithme de Raghavan a été donnée dans [CR02]. Dans cette version qui calcule une meilleure approximation du multiflot entier, un multiflot fractionnaire est résolu pour chaque commodité dans un processus itératif. La question de la rapidité de résolution du multiflot fractionnaire n'en devient que plus cruciale.

Approximer le multiflot fractionnaire Étant donné que l'algorithme précédent utilise des multiflots fractionnaires pour calculer une solution entière approchée, il n'est pas nécessaire de se baser sur des multiflots fractionnaires optimaux. En effet, l'approximation due à l'arrondi aléatoire induit un écart additif de l'ordre de la racine carrée des capacités fractionnaires [Rag94, CR02]. Si ces capacités fractionnaires sont des $(1+\varepsilon)$-approximations des capacités optimales, l'ordre de grandeur de l'approximation finale ne changera pas. Il est donc naturel d'exploiter cette marge de manoeuvre pour tenter d'accélérer sensiblement l'algorithme d'approximation du multiflot entier.

Un premier travail, [GK98], propose un algorithme combinatoire de calcul d'une $(1+\varepsilon)$-approximation du multiflot fractionnaire en $O\left(|C|\frac{|E|^2|V|^2}{\varepsilon^2}\right)$. Dès lors il n'est plus nécessaire d'utiliser la programmation linéaire, bien plus coûteuse à mettre en oeuvre. Cet algorithme a ensuite été amélioré dans [Fle00] pour diminuer encore la complexité à $O\left(\frac{|E|^2\log|V|}{\varepsilon^2}\right)$. Il est à noter que ces algorithmes sont principalement basés sur des calculs de plus courts chemins dans un graphe auxiliaire. C'est ce point, crucial pour la complexité de l'algorithme, que nous nous proposons d'améliorer.

2 Rappels sur l'algorithme de Fleischer

Algorithme 1 $(1+\varepsilon)$-approximation du multiflot maximum par Fleischer [Fle00]

Entrée: $G = (V,E),\ |V| = n,\ |E| = m,\ C = \{(s_i,t_i),\ i = 1\dots k\} \subseteq V^2, \varepsilon > 0$
Sortie: l métrique telle que tout chemin $s_i \to t_i$ est de longueur > 1.
Sortie: f $(1+\varepsilon)$-approximation du max-multiflot de C dans G

1: {Initialisation} $\forall e \in E,\ l(e) = \delta = (1+\varepsilon)((1+\varepsilon)n)^{-\frac{1}{\varepsilon}},\ f_i(e) = 0$
2: {Borne sur la longueur d'un SSSP } $\lambda = \delta$
3: **tant que** $\lambda \leq 1+\varepsilon$ **faire**
4:   **pour tout** $i = 1\dots k$ **faire**
5:     $P \leftarrow$SSSP $(s_i \to t_i)$
6:     **tant que** $l(P) \leq (1+\varepsilon)\lambda$ **faire**
7:       $c_m \leftarrow \min_{e\in P} c(e)$
8:       $\forall e \in P,\ f_i(e) \leftarrow f_i(e) + c_m/(\log_{1+\varepsilon}\frac{1+\varepsilon}{\delta})$
9:       $\forall e \in P,\ l(e) \leftarrow l(e)(1+\varepsilon\frac{c_m}{c(e)})$
10:       $P \leftarrow$SSSP $(s_i \to t_i)$
11:   $\lambda \leftarrow \lambda(1+\varepsilon)$

L'algorithme d'$(1+\varepsilon)$-approximation du multiflot maximum proposé par Fleischer [Fle00] et reporté dans l'algorithme 1 se base sur le dual du programme linéaire du multiflot dans sa formulation *sommet-arc*. Ce dual s'interprète comme la construction d'une métrique l sur les arcs du graphe telle que le poids d'un arc correspond à la quantité de flot qui le traverse. L'algorithme 1 construit donc un flot maximum valide en ne considérant que la métrique associée.

Le poids initial d'un arc est une constante $\delta > 0$, correspondant à une quantité de flot nulle (ligne 1).

Ensuite, l'algorithme procède à des poussées de flots successives le long de plus courts chemins (*single source shortest paths*, SSSP) pour chaque commodité (lignes 3-11). Pour la commodité de s_i à t_i, on pousse sur le plus court chemin P de capacité minimale c_m la quantité de flot $c_m/(\log_{1+\varepsilon}\frac{1+\varepsilon}{\delta})$, puis pour chaque arc $e \in P$ on augmente $l(e)$ d'un facteur $1+\varepsilon\frac{c_m}{c(e)}$ (lignes 5-10). L'algorithme termine lorsque tous les plus courts chemins ont une longueur supérieure à 1.

L'argument principal de l'analyse de complexité, donné dans [GK98] et adapté pour cet algorithme dans [Fle00], est que, au pire, à chaque poussée de flot le plus court chemin ne contient qu'un seul arc et sa longueur augmente d'un facteur $1+\varepsilon$, car $c_m = c(e)$. La longueur initiale des arcs est δ et l'algorithme termine au pire lorsque tous les arcs ont longueur $1+\varepsilon$. Donc il y a, au pire, $m\log_{1+\varepsilon}\left(\frac{1+\varepsilon}{\delta}\right)$ poussées. Vu le choix de δ, imposé pour garantir le facteur d'approximation, cela donne $O\left(\frac{m\log_{1+\varepsilon} n}{\varepsilon^2}\right)$ poussées, chacune correspondant au calcul d'un plus court chemin dans un graphe où tous les arcs sont de poids positifs (SSSP>0) de coût $O(m+n\log n)$ avec Dijkstra. Donc, la complexité dans le cas général est $O\left(\frac{m\log_{1+\varepsilon} n}{\varepsilon^2}(m+n\log n)\right)$.

3 Les plus courts chemins incrémentaux

Dans l'algorithme 1, la boucle *tant que* (lignes $6-10$) effectue plusieurs calculs successifs de SSSP pour une même commodité $s \to t$ et, entre deux de ces calculs, seul la longueur l des arcs du chemin de s à t est augmentée. Aussi, on peut se contenter, ligne 10, de *mettre à jour* l'arborescence des plus courts chemins. En effet, l'augmentation de la longueur de certains arcs n'aura d'influence que sur une partie de l'arborescence. Malheureusement, les algorithmes classiques de mise à jour d'arborescence des plus courts chemins (*updated* SSSP, USSSP) ne considèrent que le cas de l'augmentation de la longueur d'un arc [FMSN00] alors que nous modifions la longueur de tous les arcs du chemin de s à t. Nous donnons donc ici une modification de l'algorithme de [FMSN00] pour ce cas plus général.

Dans [FMSN00], l'algorithme de mise à jour de l'arbre des plus courts chemins lors de l'augmentation de la longueur d'un arc se décompose en deux étapes. L'étape 1 colorie l'ensemble des sommets V du graphe comme suit :

- $q \in V$ est **rouge** si sa distance à s augmente.
- $q \in V$ est **rose** si q préserve sa distance à s mais doit changer de père dans $T(s)$.
- $q \in V$ est **blanc** si q ne change ni de distance ni de père.

L'étape 2 calcule les nouvelles distances pour les sommets *rouges* en appliquant une procédure similaire à l'algorithme de Dijkstra. Remarquons que les coloriages **rose** et **blanc** sont implicites puisque l'étape 2 ne considère que les sommets **rouges**. Le lemme suivant prouve la validité de ces deux étapes combinées.

Lemme 1 ([FMSN00]) *Lors de l'augmentation de la longueur d'un arc, si l'on fournit à l'étape 2 un coloriage valide des sommets, on obtient une arborescence des plus courts chemins correcte.*

En se basant sur le lemme 1, nous pouvons nous contenter de modifier l'étape 1 pour l'adapter au cas de l'augmentation de la longueur d'un plus court chemin. L'algorithme 2 opère ce coloriage, le lemme 2 prouve sa validité. Pour chaque $z \in V$, $D(z)$ représente la distance de z à s, et $P(z)$ désigne le père de z dans $T(s)$.

Le principe de l'algorithme 2 est similaire à la procédure de marquage de [FMSN00], seule l'initialisation diffère véritablement. L'algorithme commence par insérer les sommets du chemin $s \to t$ dans une queue M (ligne 1). Cette queue ne contiendra que les sommets impactés par la modification (les rouges et les roses). Ensuite, tant que M n'est pas vide (ligne 2), il traite le sommet le plus proche de s (ligne 3) et vérifie si sa distance est conservée (ligne 4). Dans ce cas, son père est mis à jour, le sommet est implicitement marqué rose (ligne 5), et tout le sous-arbre issu de ce sommet est implicitement marqué blanc vu qu'il ne sera jamais inséré dans M. Dans le cas contraire, le sommet est marqué rouge (ligne 7) et ses fils qui ne le sont pas déjà sont insérés dans M puisqu'ils doivent au moins être roses (lignes 8, 9).

Lemme 2 (validité du coloriage) *Soient $G=(V,E)$ un graphe, $T(s)$ l'arborescence des plus courts chemins enracinée en s et $P_{s,t}$ le plus court chemin de s à t. Si on augmente la longueur de tous les arcs du chemin $P_{s,t}$ alors l'algorithme 2 colorie correctement tous les sommets de G.*

Algorithme 2 Marquage des sommets pour la mise à jour de l'arbre des plus courts chemins

Entrée: $T(s)$ un arbre des plus courts chemins issus de s, $s \rightarrow t$ un plus court chemin.
Sortie: Une bonne coloration des sommets rouges.

1: {Initialisation} $\forall y \neq s$ un sommet de $s \rightarrow t$, Enfiler(M,$\langle y, D(y)\rangle$).
2: **tant que** Non-Vide(M) **faire**
3:    $\langle z, D(z)\rangle \leftarrow$ Extraire-Min(M).
4:    **si** $\exists\, q \notin M$ voisin non rouge de z t.q. $D(q) + l(q,z) = D(z)$ **alors**
5:       $P(z) \leftarrow q$ {z est rose}
6:    **sinon**
7:       $couleur(z) \leftarrow$ rouge
8:       **pour tout** v fils de $z \notin M$ **faire**
9:          Enfiler(M,$\langle v, D(v)\rangle$)

Preuve : La validité du coloriage, découle des propriétés suivantes.

($\mathcal{P}_1$) Tous les sommets de $P_{s,t}$ à l'exception de la racine s sont insérés dans M.

($\mathcal{P}_2$) Tous les fils d'un sommet rouge sont insérés dans M. Tous les fils qui ne sont pas sur le chemin $P_{s,t}$ d'un sommet rose ne sont jamais insérés dans M.

($\mathcal{P}_3$) Un sommet est blanc si et seulement si il n'a jamais été inséré dans M.

($\mathcal{P}_4$) Les sommets sont extraits de M par ordre de distance croissante : si u et q sont présents dans M, pas nécessairement simultanément, alors $d(u) < d(q)$ implique que u est extrait avant q.

Les propriétés ($\mathcal{P}_1$) et ($\mathcal{P}_2$) sont des conséquences triviales de l'algorithme 2 (ligne 1 pour ($\mathcal{P}_1$), ligne 4 à 9 pour ($\mathcal{P}_2$)). ($\mathcal{P}_3$) est due au fait que l'on n'enfile que les fils des sommets rouges qui sont rouges ou roses.

Reste à prouver ($\mathcal{P}_4$). On note $t(x)$ la "*date*" à laquelle le sommet x est extrait de M. ($\mathcal{P}_4$) est trivialement vraie lorsque le premier sommet est extrait.

Soit v un sommet. Supposons, par récurrence, que ($\mathcal{P}_4$) est vraie pour tout $u, u' \in \Delta(v)$, l'ensemble des sommets extraits avant v. Au temps $t(v)$, $v = \min\{x \in M\}$. Soit $P(v) \in \Delta(v)$ le père de v.

– $\forall u \in \Delta(v)$ t.q. $t(u) < t(P(v))$, ($\mathcal{P}_4$) et le fait que le père de v ait une distance à s inférieure à celle de v impliquent que $D(u) \leq D(P(v)) \leq D(v)$.
– $\forall u \in \Delta(v)$ t.q. $t(P(v)) < t(u)$, par définition de $\Delta(v)$ $t(u) < t(v)$. De plus, puisque v est inséré dans M au plus tard quand $P(v)$ en est extrait (lignes 8 et 9), il en suit que v est déjà dans M à $t(u)$. Ainsi, u et v sont dans M simultanément à $t(u)$. M étant une file, à $t(u)$ $u = \min\{x \in M\}$, et donc $D(u) < D(v)$.

Conséquemment, ($\mathcal{P}_4$) est vraie pour $\Delta(v) \cup \{v\}$.

□

Remarquons qu'il est très difficile d'exprimer la complexité effective de l'algorithme de mise à jour, USSSP. Dans [FMSN00], les auteurs l'expriment en fonction du nombre α de sommets changeant soit de père soit de distance, et d'une constante β dépendant de la classe de graphes considerée ($\beta \leq 3$ pour les graphes planaires, $\beta \leq d$ pour les graphes de degré maximum d, ..., $\beta = O(\sqrt{m})$ pour les graphes généraux). La complexité d'USSSP est alors $O(\alpha\beta \log n)$ et est toujours inférieure à la complexité de l'algorithme de Dijkstra, $O(m + n\log n)$.

Complexité globale Nous pouvons maintenant exprimer la complexité de l'algorithme de Fleisher en fonction du coût de la mise à jour de l'arborescence des plus courts chemins. En effet, d'après l'analyse de complexité de l'algorithme de Fleisher, nous savons que l'algorithme effectue, entre les lignes 4 et 10, m calculs de SSSP dont k ne peuvent pas être remplacés par des calculs de USSSP (ligne 5). Aussi, l'algorithme calcule $(m-k)$ USSSP et la complexité en fonction de la complexité SSSP et USSSP devient :

$$\frac{\log_{1+\varepsilon} n}{\varepsilon^2}\,(k.sssp + (m-k).Usssp)\,.$$

Dans la suite, nous allons exprimer la complexité de l'algorithme d'$(1+\varepsilon)$-approximation du multiflot fractionnaire en le dédiant au problème du routage optique dans les réseaux WDM multifibres. Par mesure de simplification, nous utiliserons $O(m + n\log n)$ comme complexité de USSSP, sauf précision explicite.

4 Routage optique et flot

Notre objectif est d'utiliser ce calcul du multiflot fractionnaire dans un algorithme de dimensionnement de réseau optique WDM. Dans un travail précédent [CR02], nous avons modélisé le problème du routage optique par un multiflot entier dans un graphe auxiliaire. Nous avons aussi présenté un algorithme d'approximation du multiflot entier fonctionnant comme un arrondi aléatoire d'une suite de multiflots fractionnaires. Le nombre et le coût des calculs de multiflots motivent la recherche des meilleures performances.

Dans un premier temps nous exprimons la complexité du calcul de multiflot fractionnaire approché en fonction des paramètres du réseau WDM. Ensuite nous spécialiserons nos algorithmes de calcul de plus courts chemins pour améliorer encore les performances en l'absence d'équipement de conversion en longueurs d'onde.

4.1 Cas général

Il s'agit d'un graphe fait de couches (une par longueur d'onde) reliant d'un coté, pour une commodité N, une source SN à toutes les copies sN_i de la source du graphe d'origine, de l'autre reliant tous les puits t_i à des noeuds intermédiaires D_sN_t puis à SN'. Les couches sont reliées entre elles par des motifs modélisant les conversions[†]. La figure 1 donne une représentation de ce graphe, sans conversion. Les requêtes qui ont la même source sont regroupées dans un flot visant plusieurs destinations. Par exemple, deux requêtes dans le réseau d'origine issues de $s1$ et à destination de t et u seront modélisées par un flot partant du noeud $S1$, lui même relié aux w copies de s, $(s1_1,\ldots,s1_w)$. Puis les w copies de t, $(t_1,\ldots,t_w)$ sont reliées au noeud D_s1_t qui concentre le traffic à destination de t ; de même pour les noeuds $(u_1,\ldots,u_w)$ reliés à D_s1_u. Enfin, les destinations des requêtes issues de $S1$ sont reliées au noeud $S1'$.

Si le réseau WDM a n nœuds et m liens, lorsqu'il y a w longueurs d'onde et une instance de communication qui contient au moins toutes les paires possibles[‡] – il peut y avoir des multiplicités –, le graphe auxiliaire a $O(n^2)$ sommets et $O(wn^2)$ arcs. Si l'on ne regarde pas plus loin, la complexité de l'algorithme de Fleischer est donc :

$$O\left(\frac{\log_{1+\varepsilon} n}{\varepsilon^2} wn^4(w+2\log n)\right).$$

Pour aller plus loin, remarquons que la structure du graphe amène à une spécialisation simple de l'algorithme de plus courts chemins : nous ne nous intéressons pas à un arbre des plus courts chemins, mais plutôt à un faisceau allant d'une source à un puit. De plus, l'orientation des arcs aux extrémités du faisceau nous permet d'ignorer les sources et puits inutiles. Ainsi, le nombre de sommets considérés à chaque calcul de plus courts chemins n'est pas en $O(n^2)$ mais en $O(nw)$ et le nombre d'arcs est en $O(mw)$. Toutefois, le nombre d'arcs considérés dans l'algorithme de multiflot reste en $O(wn^2)$, soit la totalité du graphe.

En conséquence, un calcul de multiflot coûte :

$$O\left(\frac{\log_{1+\varepsilon} n}{\varepsilon^2} n^2w^2(m+n\log nw)\right).$$

4.2 Cas sans conversion

En l'absence d'équipement de conversion en longueurs d'onde dans le modèle précédent, le graphe auxiliaire sera comme celui présenté en figure 1. Dans ce cas, la structure en couche est prépondérante, et il est possible de spécialiser les algorithmes de SSSP et de USSSP en faisant des traitements couche par couche que l'on rassemble ensuite dans les sources et les puits.

L'algorithme de SSSP pourrait se résumer comme suit : pour chaque couche calculer un SSSP enraciné en la copie de la source considérée puis, pour chaque nœud du réseau, déterminer par quelle couche du graphe passe le véritable plus court chemin grâce à un traitement en $O(w)$ aux extrémités. La complexité d'un tel algorithme est $O\left(w(m+n\log n)\right)$.

[†] Cette structure n'est pas plus détaillée pour des raisons de place. Le lecteur intrigué pourra regarder [CR02] pour plus de détails.

[‡] Les réseaux WDM étant majoritairement des réseaux d'infrastructure, il semble raisonnable de penser qu'un noeud va initier au moins une communication vers tous les autres.

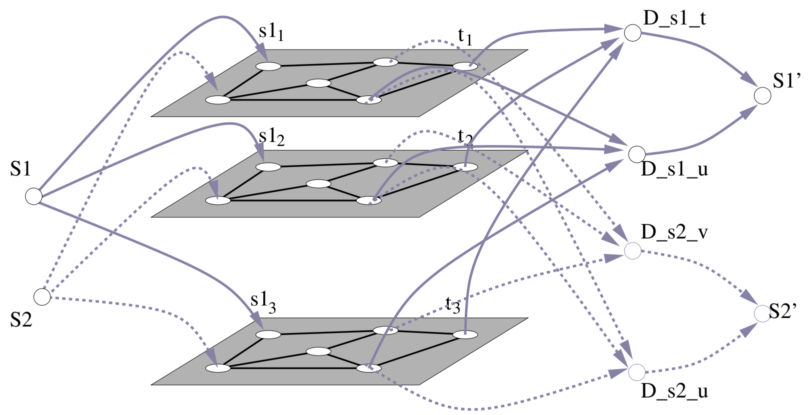

FIG. 1: Exemple de graphe auxiliaire pour 4 requêtes issues de 2 sources, $w = 3$, pas de conversion

Un chemin ne pouvant passer qu'à travers une seule couche, ce qui n'était pas vrai en présence de convertisseurs, la spécialisation de l'USSSP est encore plus efficace : il suffit de faire la mise à jour de la structure dans la couche où la métrique a changé puis de mettre à jour les liaisons aux extrémités, ce qui se fait en temps $O(n \log w)$ en utilisant un tas. Selon les remarques données en Section 3, la complexité totale d'USSSP devient $O(\alpha\beta \log n) < O(m + n \log n)$. Il est important de noter que dans ce cas, on peut minorer le gain de l'utilisation d'USSSP par un facteur w.

En conclusion, dans le cadre du dimensionnement d'un réseau WDM sans conversion, l'algorithme de Fleischer prend un temps au pire

$$O\left(\frac{\log_{1+\varepsilon} n}{\varepsilon^2} n^2 w (m + n \log nw)\right)$$

soit une amélioration d'un facteur $O(w)$ par rapport à la précédente version et d'un facteur $O\left(\frac{n^4 w^2 \varepsilon^2}{\log n}\right)$ par rapport au $O(n^8 w^3)$ nécessaire à la résolution du programme linéaire présenté dans [CR02].

4.3 Expérimentations

Des expérimentations ont été réalisées sur un PIV, cadencé à 2.2Ghz avec 512MB de mémoire. Notre implémentation met en œuvre l'algorithme de Belman-Ford, qui se trouve être plus efficace que l'algorithme de Dijkstra sur des graphes de moins de quelques milliers de sommets. L'algorithme de mise à jour est lui aussi basé sur Belman-Ford.

Une première série de tests concerne un réseau dorsal Américain appelé réseau *pan-american* constitué de 65 noeuds représentant les grandes agglomérations Américaines interconnectées par 78 liens. La matrice de trafic utilisée est constituée de 1305 requêtes de communication connectant 192 paires de noeuds. Les temps de calculs représentés en figure 2, montrent que notre implémentation de l'algorithme 1 permet de traiter des réseaux de grande taille, à l'inverse du solveur de programme linéaire CPLEX. En particulier, il devient possible de traiter des réseaux WDM *denses* utilisant plusieurs centaines de longueurs d'onde par fibre alors que CPLEX échoue sur le réseau *pan-american* à partir de ce seuil critique : 13180 secondes sont nécessaires pour la dernière instance admissible pour CPLEX comprenant 99 longueurs d'onde. Au delà, la taille mémoire du programme linéaire dépasse les capacités de la machine.

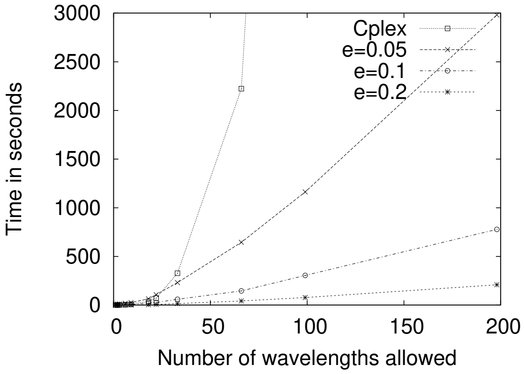

FIG. 2: Temps de calcul de l'Algorithme 1 pour différents ε comparé au temps de calcul nécessaire à CPLEX

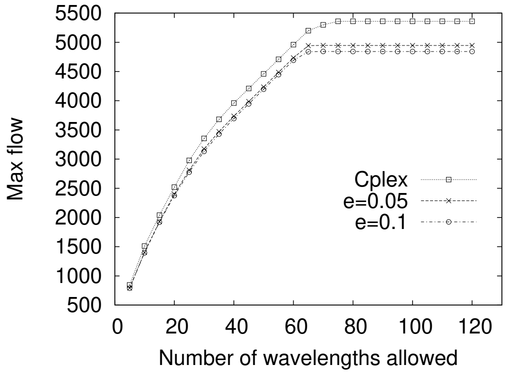

FIG. 3: Valeur du flot maximum comparée à la solution exacte donnée par CPLEX en fonction du nombre de longueurs d'onde

La deuxième série d'expérimentations, concerne le réseau NSFNET, décrit dans [SS02], constitué de 14 noeuds et 21 liens symétriques avec une instance de communication de 268 requêtes. Pour que la taille du problème soit conséquente les requêtes ont été multipliées par 20 et le nombre de fibres par lien à été arbitrairement fixé à 5. La figure 3 présente la valeur du flot maximal obtenu par l'algorithme 1 pour $\varepsilon = 0.05$ et $\varepsilon = 0.1$. Lorsque $\varepsilon = 0.1$, la valeur du flot est bien à 10% du flot maximal calculé par CPLEX (différence de 518 unités de flot). Par contre, pour $\varepsilon = 0.05$, la valeur du flot devrait se situer à moins de 5% de la valeur optimale, ce qui n'est pas le cas. En effet, lorsque ε est trop proche de zéro, des problèmes de précision numérique apparaissent. Ceux-ci, déjà évoqués dans [Fle00], sont dus à la manipulation simultanée de valeurs d'ordre de grandeur très différents.

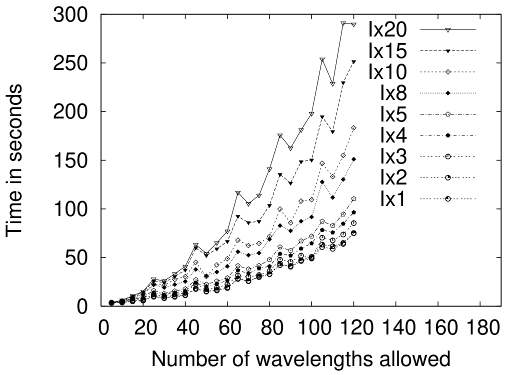

FIG. 4: Temps de calcul pour $\varepsilon = 0.05$ sur NSFNET et son instance I, multipliée par $k = 1 \dots 20$

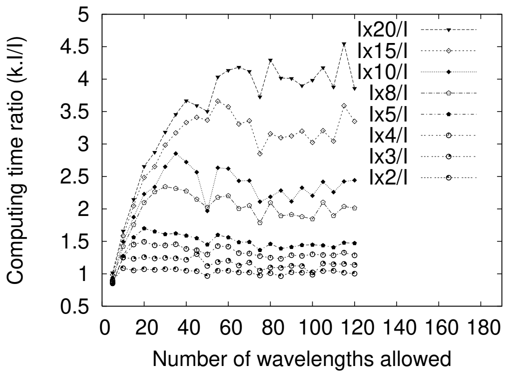

FIG. 5: Ratio Temps(k.I)/Temps(I) pour $\varepsilon = 0.05$

La figure 4 présente le temps d'exécution pour le calcul du flot maximal pour l'instance I du réseau NSFNET, celle-ci étant multipliée par un facteur k, tandis que la figure 5 donne le ratio entre ce temps de calcul et le temps pour l'instance I seule. La figure 4, comme la figure 2, illustre bien la dépendance linéaire entre le temps de calcul et le nombre de longueurs d'onde disponibles. Par ailleurs, la figure 5, montre une dépendance logarithmique à la taille de l'instance : pour un facteur multiplicatif de 20, le temps est multiplié par 4. Notons aussi que lorsque la valeur du flot maximum est atteinte, aux alentours de 60 longueurs d'onde d'après la figure 3, le temps n'augmente plus et le ratio Temps(k.I)/Temps(I) reste constant.

Perspectives

Dans cet article, nous avons présenté une évolution de l'algorithme combinatoire le plus performant pour le calcul de multiflot fractionnaire approché. L'utilisation d'algorithme de mise à jour de l'arbre des plus courts chemins permet de sensiblement accélérer le processus même s'il est difficile de chiffrer précisément le gain en complexité obtenu.

Dans le cadre de l'approximation du routage optique, l'application de cette évolution associée à la spécialisation des algorithmes de plus courts chemins permet d'obtenir un important gain en complexité. De plus, le fait de ne plus être contraint à résoudre un programme linéaire important rend possible le dimensionnement de réseaux WDM de grande taille et capacité, ce qui apparaissait comme une limitation de l'approche présentée dans [CR02].

Toutefois ce travail est loin d'être terminé. Lors de nos implémentations, nous avons rencontré des problèmes de précision numérique qui empèchent l'exploitation pratique de cet algorithme avec une grande précision. Nous envisageons de traiter ces difficultés par des méthodes de remise à l'échelle de la métrique au fur et à mesure de l'algorithme.

Enfin, une autre approche est possible qui consisterait à partir d'une solution du multiflot violant les capacités pour la transformer en une solution valide par des techniques de relaxation lagrangienne couplée à des descentes de gradient.

Remerciements

Nous tenons à remercier Christophe Paul pour son aide appréciable.

Références

[BT97] D. Bertimas and J. N. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1997.

[CR02] D. Coudert and H. Rivano. Lightpath assignment for multifibers WDM optical networks with wavelength translators. In *IEEE Globecom'02*, Taiwan, November 2002. OPNT-01-5.

[FF62] L. Ford and D. Fulkerson. *Flows in Networks*. Princeton University Press, 1962.

[Fle00] L. Fleischer. Approximating fractional multicommodity flows independent of the number of commodities. *SIAM J. Discrete Math.*, 13(4) :505–520, 2000.

[FMSN00] D. Frigioni, A. Marchetti-Spaccamela, and U. Nanni. Fully dynamic algorithms for maintaining shortest paths trees. *Journal of Algorithms*, 34(2) :251 – 281, February 2000.

[FPR+03] A. Ferreira, S. Pérennes, A. W. Richa, H. Rivano, and N. Stier Moses. Models, Complexity and Algorithms for the Design of Multifiber WDM Networks. In *ICT'03*, Papeete, French Polynesia, February 2003. To be published.

[GK98] Naveen Garg and Jochen Konemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. In *IEEE Symposium on Foundations of Computer Science*, pages 300–309, 1998.

[KS01] R. Krishnaswamy and K. N. Sivarajan. Algorithms for Routing and Wavelength Assignment Based on Solutions of LP–Relaxations. In *IEEE Communications Letters*, volume 5, pages 435–437. IEEE, October 2001.

[Kum98] V. Kumar. Approximating arc circular colouring and bandwidth allocation in all-optical ring networks. In *APPROX'98*, 1998.

[LS00] G. Li and R. Simha. On the Wavelength Assignement Problem in Multifiber WDM Star and Ring Networks. *IEEE Infocom*, 3 :1771–1780, 2000.

[MS00] L. Margara and J. Simon. Wavelength assignment problem on all-optical networks with k fibres per link. In *ICALP'00*, pages 768–779, 2000.

[Rag94] P. Raghavan. Probabilistic construction of deterministic algorithm : Approximating packing integer programs. *Journal of Computer and Systems Sciences*, 38 :683–707, 1994.

[Riv01] H. Rivano. Planification de réseaux optiques WDM k-fibres. In *AlgoTel'01*, pages 41–46, Saint Jean de Luz, France, mai 2001. INRIA.

[SS02] M.D. Swaminathan and K.N. Sivarajan. Practical routing and wavelength assignment algorithms for all optical networks with limited wavelength conversion. In *IEEE ICC*, New-York City, 2002. IO4 - 4.