Chemins disjoints de poids minimum pour la sécurisation de réseaux de télécommunications

David Coudert

Chemins disjoints de poids minimum pour la sécurisation de réseaux de télécommunications

David Coudert

Projet MASCOTTE, CNRS-I3S-INRIA, 2004, route de Lucioles, BP 93, F-06902 Sophia-Antipolis Cedex, France

Cette étude s'intéresse à la planification de réseaux de télécommunications tolérants aux pannes. Nous cherchons à établir, pour chaque couple de nœuds du réseau, deux chemins de communication disjoints, l'un étant réservé à la protection de l'autre. Pour un réseau à n nœuds et m liens de communications, nous donnons un algorithme en $O(n(m+n\log n))$, permettant de calculer depuis un nœud donné et vers chacun des autres nœuds deux chemins arc-disjoints dont la somme des poids est minimale. Ceci améliore la complexité des solutions basées sur les algorithmes de flot de poids minimum, qui est en temps $O(m(m+n\log n)\log n)$ pour un seul couple de sommets.

Keywords: protection, réseaux optiques, flots, plus court chemins, chemins disjoints

1 Introduction

Cette étude s'intéresse à l'établissement de chemins de réserves, prédéterminés et dédiés, pour la planification de réseaux de télécommunications tolérants aux pannes. En effet, avec l'accroissement du trafic et de la taille des réseaux, les réseaux tolérants aux pannes sont devenus un point essentiel dans le dimensionnement et la planification des grands réseaux de télécommunications. Cette exigence se retrouve, entre autre, dans les réseaux WDM (Wavelength Division Multiplexing) [Bea00, TDDC98] et dans les réseaux MPLS (Multi Protocol Label Switching) [Awd99, MSOH99].

La sécurisation du réseau, indispensable pour pallier les pannes d'équipements et de liens, consiste à établir un re-routage du trafic interrompu dans le cas d'une panne et à sur-dimensionner la couche optique en conséquence [ZS00]. Les ressources dédiés à la sécurisation sont appelés les ressources de réserves. Deux méthodes de sécurisation se distinguent : la *protection* et la *restauration*.

La protection est une technique automatique utilisant des ressources de réserve prédéterminées et dédiées [TDDC98]. Un chemin de réserve est pré-affecté entre deux sommets pour remplacer le chemin dégradé ou interrompu. La protection permet d'obtenir une réponse automatique et immédiate en cas de panne, au détriment d'un sur-dimensionnement du réseau. La restauration est une mécanique de sécurisation qui établi un nouveau chemin en fonction des ressources disponibles dans le réseau au moment de la panne [MS00].

Nous nous intéressons ici à la *sécurisation par protection* d'une connexion établie entre deux nœuds du réseau [Bla00]. Pour cela, nous cherchons à établir deux chemins disjoints entre deux nœuds du réseau, l'un des chemins permet d'établir la connexion et l'autre sert à sa protection. Le critère de choix retenu pour ses deux chemins est que la somme de leur coût soit minimum. Une étude similaire pour la sécurisation par protection de liens dans les réseaux ATM se trouve dans [SL00].

Nous modélisons ce problème de protection et nous montrons que c'est un problème de *flot de poids minimum*, dans la section 2. Ensuite, dans la section 3, nous donnons un algorithme efficace pour déterminer, depuis une source et vers une destination, deux chemins disjoints dont la somme des coûts est minimum. Puis, dans la section 4, nous généralisons cet algorithme pour obtenir un algorithme calculant, depuis une source et vers chacun des autres nœuds, deux chemins disjoints dont la somme des coûts est minimum. Enfin, nous concluons ce travail dans la section 5.

Notations Le graphe $G = (V(G), A(G))$ est le graphe orienté à $|V(G)| = n$ sommets et $|A(G)| = m$ arcs. Un arc d'un sommet u vers un sommet v est noté (u,v) et est de poids $w((u,v))$. Le poids $w(C)$ du chemin C correspond à la somme des poids des arcs qu'il emprunte. Enfin, $T(s)$ dénote un arbre des plus courts chemins enraciné en s dans le graphe pondéré.

2 Modélisation

Le réseau de communication est modélisé par un graphe orienté symétrique et pondéré. Les nœuds (routeurs) du réseau correspondent aux sommets du graphe, et les fibres optiques aux arcs. Le poids des arcs (u,v) et (v,u) (arcs symétriques) sont égaux.

Notre problème de protection de chemin s'exprime alors de la façon suivante :

Problème 2.1 *Étant donné un graphe orienté et pondéré $G = (V,A)$, sans arcs de poids négatif, et deux sommets s et t, trouver deux chemins sommet-disjoints dont la somme des poids soit minimum, de s vers t.*

En effet, deux chemins sommet-disjoints assurent bien une protection en cas de panne de routeur ou de lien. Si seule la protection en cas de panne de lien est considérée, alors le problème s'exprime comme suit :

Problème 2.2 *Étant donné un graphe orienté et pondéré $G = (V,A)$, sans arcs de poids négatif, et deux sommets s et t, trouver deux chemins arc-disjoints dont la somme des poids soit minimum, de s vers t.*

Il est important de remarquer qu'une modification simple du graphe $G = (V(G), A(G))$ permet de réduire le problème sommet-disjoints au problème arc-disjoints [AMO93]. Pour cela, nous construisons le graphe $H = (V(H), A(H))$ où à tout sommet $u \in V(G)$ nous associons deux sommets u^- et u^+ dans $V(H)$ et à tout arc $(u,v) \in A(G)$ nous associons l'arc $(u^+, v^-) \in A(H)$. De plus, nous ajoutons à $A(H)$ les arcs (u^-, u^+), de poids $w((u^-, u^+)) = 0$. Ainsi, $|V(H)| = 2|V(G)| = 2n$ et $|A(H)| = |A(G)| + |V(H)| = m + n$. Cette construction est illustrée par la Figure 1.

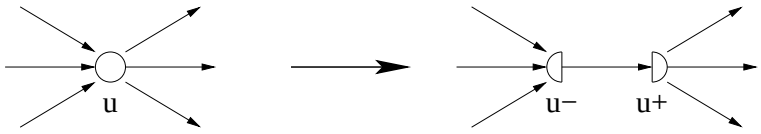

FIG. 1 – Réduction chemin sommet-disjoints → chemin arc-disjoints

Comme il n'existe qu'un seul arc entre les sommets u^- et u^+ et que cet arc est à la fois le seul arc sortant du sommet u^- et le seul arc entrant dans le sommet u^+, deux chemins arc-disjoints dans le graphe H seront bien sommet-disjoints dans le graphe G. De plus, une solution optimale dans H l'est aussi dans G [AMO93].

2.1 Réduction au problème du flot de poids minimum

La formulation générale du problème du flot de poids minimum est la suivante :

Problème 2.3 (flot de poids minimum)
Soit $G = (V,A)$, un graphe orienté, pondéré par la fonction w sur les arcs, et soit c une fonction de capacité sur les arcs. Étant donnés deux sommets s et t, et un entier k, minimiser $\sum_{a \in A} f(a)w(a)$, où $f(a)$ est le flot traversant l'arc a, tout en assurant :

- $\sum_{w \in \Gamma^+(s)} f((s,w)) = k$ *et* $\sum_{v \in \Gamma^-(s)} f((v,s)) = 0$
- $\sum_{v \in \Gamma^-(t)} f((v,t)) = k$ *et* $\sum_{w \in \Gamma^+(t)} f((t,w)) = 0$
- $\forall u \in V - \{s,t\}$, $\sum_{v \in \Gamma^-(u)} f((v,u)) = \sum_{w \in \Gamma^+(u)} f((u,w))$ *(contrainte de conservation du flot)*
- $\forall a$, $f(a) \leq c(a)$ *(contrainte de capacité des arcs)*

Lorsque la capacité des arcs du graphe est de 1, ($\forall a \in A,\ c(a) = 1$), la solution de ce problème est en variables entières [CCPS98]. Ainsi, le problème 2.3 revient à trouver k chemins arc-disjoints depuis un sommet s vers un sommet t, dont la somme des poids est minimale. Lorsque de plus $k = 2$, la résolution de ce problème fournie une solution optimale au problème 2.2.

Les meilleurs algorithmes pour le problème du flot de poids minimum sont dus à Galil et Tardos [GT88] : $O(n^2(m+n\log n)\log n)$, et Orlin [Orl88] : $O(m(m+n\log n)\log n)$. Ces algorithmes sont généraux, valables pour tout k. Nous allons donc regarder plus précisément le problème pour $k=2$.

3 Algorithme de 2-flot de poids minimum pour un couple donné

Les algorithmes de flot de poids minimum fonctionnent par augmentations successives du flot [CCPS98] : trouver un flot de valeur 1 et de coût minimum puis augmenter sa valeur de 1 tout en garantissant la minimalité de la solution. Pour réaliser cette augmentation, les algorithmes commencent par trouver un chemin entre la source et la destination, puis, par recherche successives de *chaînes augmentantes*, diminuent le coût de cette solution jusqu'à converger vers la solution minimale.

Une synthèse de cette technique nous conduit, pour le problème 2.2, à l'algorithme suivant :

Algorithme de 2-flot de poids minimum

1. Calculer le chemin C_0 de poids minimum entre s et t dans G. $C_0 = (s = u_0, u_1, u_2, \cdots, u_k = t)$.
   Soit $C_0^{\leftarrow} = (t = u_k, u_{k-1}, \cdots, u_0 = s)$.
2. Construire le graphe G' tel que : $V(G') = V(G)$ et $A(G') = A(G) - \{(u,v) \in C_0\}$.
   Les arcs $(u,v) \in C_0^{\leftarrow}$ prennent le poids $-w((u,v))$ (poids négatifs).
3. Calculer le chemin C_1 de poids minimum entre s et t dans G'.
4. Construire les chemins C_α et C_β à partir de C_0 et C_1 privés des arcs (u,v) tels que $(u,v) \in C_0$ et $(v,u) \in C_1$. Pour cela, on procède de la façon suivante :
   Démarrer du sommet s en suivant le chemin C_1. Avancer le long de C_1 jusqu'à rencontrer un sommet qui appartient également à C_0. Suivre C_0 jusqu'à rencontrer un sommet qui appartient également à C_1. Réitérer jusqu'à arriver en t. Nous avons obtenu C_α. Le même procédé, en suivant d'abord C_0 donne C_β.

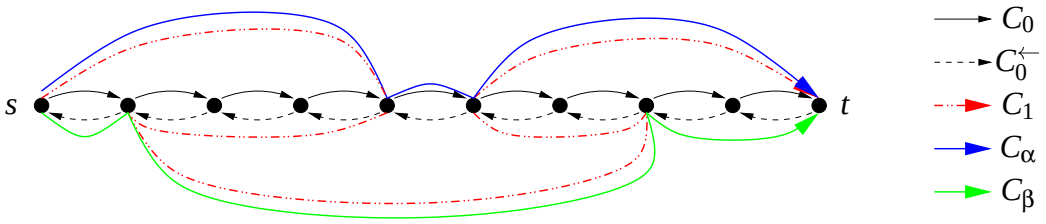

FIG. 2 – Chemins considérés par l'algorithme de 2-flot de poids minimum.

Proposition 3.1 *Dès que les chemins C_0 et C_1 existent, les chemins C_α et C_β existent.*

Preuve: Si les chemins $C_0^{\leftarrow}$ et C_1 sont arcs disjoints, alors il suffit de prendre $C_\alpha = C_0$ et $C_\beta = C_1$.

Sinon, supposons que les chemins $C_0^{\leftarrow}$ et C_1 aient seulement la chaîne $(x_1, \cdots, x_P)$ en commun. Nous avons

$$\begin{aligned} C_0^{\leftarrow} &= (t = u_L, \cdots, u_i = x_1, \cdots, x_P = u_j, \cdots, u_0 = s) \\ C_0 &= (s = u_0, u_1, \cdots, x_P = u_j, \cdots, x_1 = u_i, \cdots, u_L = t) \\ C_1 &= (s = v_0, v_1, \cdots, v_k = x_1, \cdots, v_l = x_P, \cdots, v_M = t) \end{aligned}$$

L'algorithme précédent nous donne les chemins

$$\begin{aligned} C_\alpha &= (s = u_0, \cdots, u_i = x_P = v_l, \cdots, v_M) \\ C_\beta &= (s = v_0, \cdots, v_k = x_1 = u_j, \cdots, u_L) \end{aligned}$$

Par définition de la chaîne $(x_1, \cdots, x_P)$, C_α et C_β sont arcs disjoints et vont bien de s à t.

Si plusieurs chaînes sont communes à $C_0^{\leftarrow}$ et C_1, il suffit d'appliquer le même algorithme sur chacune d'elles. □

Notons que $w(C_\alpha) + w(C_\beta) = w(C_0) + w(C_1)$. Il reste à montrer que cette solution est optimale.

Proposition 3.2 *La somme $w(C_\alpha) + w(C_\beta)$ des poids des chemins C_α et C_β, de s à t dans G, est minimale.*

Afin de prouver la proposition 3.2, nous avons besoin du lemme suivant :

Lemme 3.3 *Supposons que la somme $w(C_\alpha) + w(C_\beta)$ des poids des chemins C_α et C_β, de s à t dans G, est minimale. Si le chemin C_α (respectivement C_β) emprunte les arcs (u_i, u_{i+1}) et (u_j, u_{j+1}) de C_0, $0 \leq i < i+1 < j < k$ et que le chemin C_β (respectivement C_α) n'emprunte aucun des arcs $(u_{i+1}, u_{i+2}), \cdots, (u_{j-1}, u_j)$, alors le chemin C_α (respectivement C_β) emprunte également tous les arcs $(u_{i+1}, u_{i+2}), \cdots, (u_{j-1}, u_j)$.*

Preuve: Comme un plus court chemin est formé de plus court chemins, les arcs $(u_{i+1}, u_{i+2}), \cdots, (u_{j-1}, u_j)$ forment un plus court chemin entre les sommets u_{i+1} et u_j. Si C_α (respectivement C_β) n'emprunte pas ce plus court chemin alors l'hypothèse de minimalité de la somme des poids des chemins C_α et C_β est contredite. □

De ce lemme découlent les deux corollaires suivants :

Corollaire 3.4 *Les arcs (s, u_1) et (u_{k-1}, t), du chemin C_0, appartiennent soit à C_α soit à C_β.*

Corollaire 3.5 *Les arcs (u_1, s) et (t, u_{k-1}) du chemin $C_0^{\leftarrow}$ n'appartiennent pas au chemin C_1.*

Preuve: (proposition 3.2)
Supposons qu'il existe deux chemins C_A et C_B arc-disjoints entre s et t dans G, tels que $w(C_A) \leq w(C_B)$ et $w(C_A) + w(C_B) < w(C_\alpha) + w(C_\beta)$.

1. Si les chemins C_A et C_0 sont arc-disjoints. Alors, C_A est un chemin dans G'. Par conséquent, $w(C_A) + w(C_0) < w(C_0) + w(C_1)$, et $w(C_A) < w(C_1)$. Or C_1 est un plus court chemin de s à t dans G'. Contradiction. Il en est de même si les chemins C_B et C_0 sont arc-disjoints.
2. Si les chemins C_A et C_B empruntent tout deux des arcs du chemin C_0
   (a) Les chemins C_A et C_B respectent le lemme 3.3 et le corollaire 3.4, sinon il y a contradiction ;
   (b) Par l'absurde, on construit à partir de C_A et C_B le chemin C_1' de G', en utilisant la construction inverse de celle permettant de construire les chemins C_α et C_β à partir de C_0 et C_1.
   Partir du sommet s en suivant le chemin qui ne suit pas l'arc (s, u_1). Le poursuivre jusqu'à intersecter le chemin $C_0^{\leftarrow}$. L'intersection a lieu en u_i. Ensuite, emprunter le chemin $C_0^{\leftarrow}$ jusqu'à intersecter l'autre chemin, en u_j, $j < i$. Réitérer à partir de u_j jusqu'à arriver en t.
   Ainsi, $w(C_0) + w(C_1') = w(C_A) + w(C_B) < w(C_\alpha) + w(C_\beta) = w(C_0) + w(C_1)$ et donc $w(C_1') < w(C_1)$, ce qui est absurde puisque C_1 est un plus court chemin de s à t dans G'.

□

3.1 Calcul de la complexité de notre algorithme

L'algorithme de 2-flot de poids minimum que nous avons décrit utilise des algorithmes de recherche de plus court chemin dans des graphes sans cycles de poids négatifs. Nous rappelons dans la table 1 la complexité et les conditions d'utilisations des algorithmes les plus adaptés à notre problème. L'algorithme de Ramalingam et Reps [RR96] consiste à maintenir dynamiquement une structure d'arbre de plus court chemin lors de la suppression où de l'insertion d'un arc dans le graphe ainsi que lors de la modification du poids d'un arc. Le coût d'une opération est $O(m + n \log n)$.

Proposition 3.6 *L'algorithme de 2-flot de poids minimum a une complexité en temps de $O(L(m + n \log n))$.*

Preuve: Dans notre algorithme, le calcul du plus court chemin, C_0, entre s et t, peut se faire en calculant l'arbre $T(s)$ des plus courts chemins depuis s. Ce calcul s'effectue en temps $O(m + n \log n)$ en utilisant l'algorithme de Dijkstra. Ensuite, nous supprimons un par un les L arcs du chemin C_0 et nous diminuons un par un le poids des L arcs du chemin $C_0^{\leftarrow}$. L'utilisation de l'algorithme de Ramalingam et Reps pour maintenir l'arbre des plus courts chemin donne un coût par opération de $O(m + n \log n)$. Ainsi, nous obtenons le chemin C_1 en temps $O(2L(m + n \log n))$. Enfin, l'extraction des chemins C_α et C_β se fait en temps $O(|C_\alpha| + |C_\beta|)$, où $|C_\alpha| + |C_\beta| \leq m$. Finalement, la complexité en temps de l'algorithme est de $O(L(m + n \log n))$, où L est le nombre d'arcs du chemin C_0. □

	complexité en temps	condition
Dijkstra [Dij59]+[FT87]	$O(m+n\log n)$	arcs de poids positifs
Bellman-Ford [CLR90]	$O(nm)$	tout graphes
Yap [Yap83]	$O(L(m+n\log n)+L^3)$	L arcs de poids négatifs
Ramalingam-Reps [RR96]	$O(L(m+n\log n))$	L arcs de poids négatifs

TAB. 1 – Algorithmes de recherche de plus courts chemins à origine unique.

4 Algorithme de 2-flot de poids minimum depuis une source

Nous allons maintenant nous attacher à l'élaboration d'un algorithme incrémental pour calculer depuis une source s et vers chacun des sommets $t \neq s$, deux chemins arc-disjoints de poids minimum.

Nous montrons tout d'abord que le calcul du 2-flot de poids minimum depuis un sommet s vers un sommet t, nous permet d'obtenir le 2-flot de poids minimum de s vers chacun des sommets u situés sur le plus court chemin de s à t, sans sur-coût.

Proposition 4.1 *Étant donné le plus court chemin $C_0 = (s = u_0, u_1, u_2, \cdots, u_{L-1}, u_L = t)$ du sommet s vers le sommet t dans un graphe G, le calcul du 2-flot de poids minimum depuis s vers chacun des sommets u_i, $1 \leq i \leq L$, se fait en temps $O(L(m+n\log n))$.*

Preuve: Le chemin C_0 nous donne les plus courts chemins C_O^i de s à u_i dans le graphe G.

Soit $T_i(s)$, l'arbre des plus courts chemins dans le graphe privé des arcs (u_{j-1}, u_j), $1 \leq j \leq i$, et où le poids des arcs (u_j, u_{j-1}), $1 \leq j \leq i$, a été diminué. Cet arbre contient le plus court chemin de s à u_i permettant de déduire un 2-flot de poids minimum de s à u_i.

A partir de l'arbre $T_i(s)$, nous pouvons construire l'arbre $T_{i+1}(s)$, en supprimant l'arc (u_i, u_{i+1}) et en diminuant le poids de l'arc (u_{i+1}, u_i). Le coût de cette construction est en $O(2(m+n\log n))$, en utilisant l'algorithme de Ramalingam et Reps. De plus l'arbre $T_{i+1}(s)$ permet d'obtenir le 2-flot de poids minimum de s à u_{i+1}.

Ainsi, nous construisons l'ensemble des arbres $T_i(s)$, $1 \leq i \leq L$, à partir de l'arbre $T_0(s)$ des plus courts chemins dans G, en temps $O(L(m+n\log n)$, et chaque $T_i(s)$ permet d'obtenir le 2-flot de poids minimum de s vers u_i. □

Proposition 4.2 *Étant donnés un graphe orienté symétrique et pondéré $G = (V, A)$ et un sommet s, un algorithme incrémental permet de calculer un 2-flot de poids minimum depuis s vers chacun des sommets $t \neq s$ en temps $O(n(m+n\log n))$.*

Preuve: Nous utilisons la preuve de la proposition 4.1. Remarquons que nous pouvons reconstruire l'arbre $T_i(s)$ à partir de l'arbre $T_{i+1}(s)$ en temps $O(2(m+n\log n))$, en effectuant l'opération inverse. Ainsi, nous obtenons un coût pour construire l'arbre $T_{i+1}(s)$ à partir de l'arbre $T_i(s)$ puis revenir à $T_i(s)$ en $O(4(m+n\log n))$.

Soit $T_0(s)$, l'arbre des plus courts chemins depuis s dans G. En effectuant un parcours en profondeur le long de cet arbre et en construisant à chaque étape l'arbre $T_i(s)$ correspondant, nous obtenons un algorithme en $O(n(m+n\log n))$ pour calculer les 2-flot de poids minimum depuis s vers chacun des $n-1$ autres sommets du graphe. □

Une amélioration possible de cet algorithme est de construire l'arbre des plus courts chemins au fur et à mesure des besoins. Le principe est donc de partir de l'arbre $A_0(s)$, ne contenant que le sommet s, puis de construire pour chaque sommet u_i le plus petit arbre $A_i(s)$ des plus courts chemins contenant u_i. Le calcul s'arrête dès que le chemin de s à u_i est connu.

Pour construire l'arbre $A_i(s)$, il faut procéder de la façon suivante : soit u_j le père de u_i dans l'arbre $T(s)$ et $A_j(s)$ son arbre des plus courts chemins. L'arbre $A_i(s)$ reçoit une copie de l'arbre $A_j(s)$. Supprimer l'arc (u_j, u_i) de l'arbre $A_i(s)$, et si u_j est le père de u_i dans $A_j(s)$ alors supprimer le sous-arbre enraciné en u_i de $A_i(s)$. Ensuite, dans l'arbre $A_i(s)$, modifier le poids de l'arc (u_j, u_{pj}) (i.e. $w((u_j, u_{pj})) \leftarrow -w((u_j, u_{pj}))$), où u_{pj} est le père de u_j dans $T(s)$, et mettre à jours l'arbre $A_i(s)$, en utilisant l'algorithme de Ramalingam et Reps. Enfin, augmenter $A_i(s)$ jusqu'à ce qu'il contienne le chemin de s à u_i.

Nous avons implémenté un tel algorithme en utilisant un tas de Fibonacci pour maintenir la frontière courante, et ainsi éviter de sauvegarder tous les arbres. A chaque étape, nous mettons à jours la frontière en fonction de la suppression d'arcs et de la diminution des poids, puis nous relançons le moteur de l'algorithme de Dijkstra jusqu'à obtention du chemin souhaité. Cette implémentation se révèle très efficace puisque son exécution, sur une machine munie d'un processeur à 733MHz, et pour les graphes aléatoires à moins de 1 500 sommets et 30 000 arcs que nous avons testé, prend moins d'une seconde.

5 Conclusion

La sécurisation par protection d'une connexion consiste à établir deux chemins arcs ou sommets disjoints entre sa source et sa destination. Cette étude, proposée par Alcatel, nous a permis de proposer un algorithme en $O(L(m+n\log n))$, pour calculer un 2-flot de poids minimum d'un sommet s vers un sommet t, qui améliore la complexité des algorithmes connus pour résoudre ce problème. Nous avons ensuite généralisé notre algorithme et obtenu un algorithme en $O(n(m+n\log n))$ pour calculer un 2-flot de poids minimum depuis une source vers chacun des autres sommets du graphe.

Nous allons maintenant nous attacher à calculer la complexité de l'amélioration que nous proposons pour l'algorithme à source unique et à faire des tests sur des réseaux réels.

Nous avons récemment eu connaissance d'une autre solution à ce problème, due à Suurballe et Tarjan [ST84], qui semble intéressante.

6 Remerciements

Je tiens à remercier Jérôme Galtier, Hervé Rivano et Afonso Ferreira pour l'aide apportée à ce travail.

Références

[AMO93] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows*. Prentice-Hall, 1993.

[Awd99] D. O. Awduche. MPLS and Traffic Engineering in IP Networks. *IEEE Communications Magazine*, pages 42–47, Dec 1999.

[Bea00] B. Beauquier. *Communications dans les réseaux optiques par multiplexage en longueur d'onde*. PhD thesis, Université de Nice Sophia-Antipolis, Janvier 2000.

[Bla00] C. Blaizot. Alcatel–Photonic Networks Unit. Communication personnelle, 2000.

[CCPS98] W. Cook, W. Cunningham, W. Pulleyblank, and A. Schrijver. *Combinatorial Optimization*. Discrete Mathematics and Optimization. Wiley Interscience, 1998.

[CLR90] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. The MIT Press, 1990.

[Dij59] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1 :269–271, 1959.

[FT87] M. Fredman and R. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, 34(3) :596–615, July 1987.

[GT88] Z. Galil and E. Tardos. An $O(n^2(m+n\log n)\log n)$ min-cost flow algorithm. *Journal of the ACM*, 35(2) :374–386, April 1988.

[MS00] G. Mohan and A. Somani. Routing dependable connection with specified failure restoration guarantees in WDM networks. In *IEEE INFOCOM*, volume 2, pages 1761–1770, 2000.

[MSOH99] S. Makam, V. Sharma, K. Owens, and C. Huang. Protection/Restoration of MPLS Networks. work in progress, Internet Draft draft-makam-mpls-protection-00.txt, June 1999.

[Orl88] J. Orlin. A faster strongly polynominal minimum cost flow algorithm. In *Proceedings of the 20th ACM Symposium on Theory of Computing*, pages 377–387. ACM Press, 1988.

[RR96] G. Ramalingam and T. Reps. An incremental algorithm for a generalization of the shortest-path problem. *Journal of Algorithms*, 21(2) :267–305, 1996.

[SL00] R. Slosiar and D. Latin. A polynomial-time algorithm for the establishment of primary and alternate paths in ATM networks. In *IEEE INFOCOM*, volume 2, pages 509–518, 2000.

[ST84] J. Suurballe and R. Tarjan. A quick method for finding shortest pairs of disjoint paths. *Networks*, 14 :325–336, 1984.

[TDDC98] F. Tillerot, E. Didelet, A. Daviaud, and G. Claveau. Efficient network upgrade based on a WDM optical layer with automatic protection switching. San Jose (CA), USA, Feb. 22-27 1998.

[Yap83] C. Yap. A hybrid algorithm for the shortest path between two nodes in the presence of few negative arcs. *Information Processing Letters*, 16(4) :181–182, May 1983.

[ZS00] D. Zhou and S. Subramaniam. Survivability in optical networks. *IEEE Network*, 14(6) :16–23, 2000.