Conception de réseaux haut débit sous contrainte de sécurisation

Jérôme Truffot

N° d'ordre : 1793
EDSPIC : 387

Université Blaise Pascal - Clermont-Ferrand II

Ecole Doctorale
Sciences Pour l'Ingénieur de Clermont-Ferrand

Thèse

présentée par

Jérôme Truffot

pour obtenir le grade de

Docteur d'Université

Spécialité : Informatique

CONCEPTION DE RÉSEAUX HAUT DÉBIT SOUS CONTRAINTE DE SÉCURISATION

Soutenue publiquement le 27 novembre 2007 devant le jury

M.	Alain QUILLIOT	Président
M.	Abdel LISSER	Rapporteur
M.	Walid BEN-AMEUR	Rapporteur
M.	Adam OUOROU	Examinateur
M.	Christophe DUHAMEL	Co-directeur de thèse
M.	Philippe MAHEY	Co-directeur de thèse

REMERCIEMENTS

Je tiens en premier lieu à remercier Philippe Mahey qui m'a fait découvrir les joies de l'optimisation lors de mes études à l'ISIMA. Je le remercie d'avoir proposé et encadré cette thèse, ainsi que pour la confiance et la liberté qu'il m'a accordées sur un sujet de recherche si riche.

Je souhaite ensuite remercier Christophe Duhamel pour avoir co-diriger ces travaux de thèse. Je le remercie d'avoir été si présent, toujours à l'écoute, encourageant la moindre de mes idées et prodiguant de précieux conseils dans les moments de doute. Je le remercie également pour nos discussions pointues, techniques, que j'affectionne tant, aussi bien sur les méthodes d'optimisation que l'implémentation.

Je remercie vivement mes rapporteurs Abdel Lisser et Walid Ben-Ameur d'avoir pris le temps de lire, jugé et apprécié ce mémoire et le travail qu'il représente. J'ai grandement apprécié l'intérêt qu'ils ont porté à mon travail ainsi que leurs remarques constructives.

Je remercie également Alain Quilliot, directeur du LIMOS, d'avoir fait en sorte que cette thèse se déroule dans les meilleures conditions au sein du laboratoire et d'avoir accepté de présider le jury.

Merci aussi à mes camarades quotidiens du laboratoire qui m'ont tant appris et avec qui j'ai pu partagé toutes mes passions. Ils ont été le soutien et la bonne humeur indispensable à la réussite de cette thèse. Une mention spéciale va à Bruno Bachelet, avec qui j'ai partagé le bureau, pour avoir subi au plus près mes joies, mes doutes et mes interrogations. Également Antoine Mahul pour sa disponibilité et son soutien dans mes "geekeries" les plus inavouables, Loïc Yon qui m'a initié à la vie de thésard, d'enseignent à l'ISIMA et de trésorier de l'association des anciens élèves, et Jonas Koko qui a été mon mentor pour le cours de Fortran 90 et pour bien d'autres choses. Je ne peux oublier aussi Christophe Devaulx pour son soutien et ses conseils, ainsi que Fabrice Baray avec qui, même de loin, j'ai partagé ma passion pour le baby-foot, la coinche, les énigmes et la randonnée.

Je souhaite aussi inclure dans ces remerciements toutes les personnes du LIMOS et de l'ISIMA qui par leur sympathie et leur disponibilité font de l'ISIMA un lieu convivial et stimulant. Merci également à toutes les personnes qui m'ont soutenu et encouragé directement et indirectement tout au long de ses années : mes parents, l'ensemble de la famille et mes amis de toujours, de l'ISIMA ou de mon tonnerrois natal.

RÉSUMÉ

L'augmentation constante des débits de transmission des données a fait évoluer les réseaux IP vers de nouveaux services. La tendance actuelle est à la convergence des réseaux où chaque opérateur veut offrir une multitude de services à ces clients. Toutefois, la maîtrise de la qualité de service reste encore aujourd'hui un défi majeur pour la recherche. Dans ce contexte, ces travaux de thèse étudient l'influence des nouveaux protocoles dans la conception de réseaux haut débit tolérants aux pannes.

Dans un premier temps, nous nous sommes intéressés aux spécificités du routage dans les réseaux MPLS (MultiProtocol Label Switching). Dans l'approche classique du routage IP, chaque paquet est indépendant et les décisions de routage se prennent à chaque noeud intermédiaire du réseau. Cependant, l'engorgement des tables de routage et la gestion de la qualité de service font partie des multiples raisons qui ont motivé l'émergence de protocoles favorisant les mécanismes de commutation. C'est le cas du protocole MPLS qui prévoit que les décisions de routage soient prises au niveau du noeuds d'entrée du réseau. Ainsi, toute une partie du flux est associée à un chemin dans le réseau. Pour limiter la taille des tables de commutation, il est alors nécessaire de limiter le nombre de ces chemins. Cette contrainte forte sur le support des flux change considérablement la complexité des problèmes de routage. Notre contribution porte principalement sur la modélisation des problèmes de flots k-séparables par des programmes linéaires en nombres entiers adaptés à une résolution exacte. La nature du problème nous a amenés à considérer une généralisation de l'application de la méthode du Branch and Price au problème du multiflot monorouté.

D'autre part, nous nous sommes intéressés à la contrainte de délai de bout-en-bout. Cette contrainte est peu étudiée dans la littérature car elle revêt un caractère non linéaire et entier très complexe. Cependant, nos modèles de flots k-séparables offrent des possibilités intéressantes pour gérer le délai de bout-en-bout sur les chemins actifs. Le problème de minimisation du délai de bout-en-bout maximal est alors résolu par combinaison de plusieurs méthodes.

ABSTRACT

The rise of modern broadband communication networks made IP networks evolve toward new services. The current trend is the convergence of networks where each operator wants to offer a multitude of services to his customers. However, the control over the quality of service is still a major challenge for research. In this context, these thesis work studies the influence of the new protocols in the design of high-speed fault-tolerant networks.

As a first step, we were interested in the specificities of routing in MPLS networks (MultiProtocol Label Switching). In the traditional approach of IP routing, each packet is independent and routing decisions are made at each intermediate node of the network. However, the bottleneck of traffic routing tables and the management of the quality of service are among the reasons that led to the emergence of protocols supporting mechanisms for commutation. This is the case of the MPLS protocol which provides that routing decisions are taken at the input nodes of the network. Thus, any part of the stream is associated with a path in the network. To limit the size of the commutation tables, it is necessary to limit the number of those paths. This strong constraint on the design of the flow considerably changes the complexity of routing problems. Our contribution focuses on the modeling of k-splittable flow problems by mixed integer linear programs adapted to an exact resolution. The nature of the problem led us to consider a broader use of the Branch and Price method with the unsplittable multicommodity flow problem.

On the other hand, we were interested in the end-to-end delay constraint. This constraint is not studied much in the literature because it takes on very complex non-linear and integer characteristics. However, our models of k-splittable flow offer interesting possibilities to manage the end-to-end delay on the active paths. The problem of minimizing the maximum end-to-end delay is then solved by a combination of several methods.

TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES TABLEAUX

INTRODUCTION

Cadre général

L'histoire des réseaux de télécommunication s'est fondée sur la spécialisation des réseaux à un seul type d'information et un seul type de service : les réseaux à commutation de circuit pour la téléphonie, les réseaux hertziens pour la télévision, les réseaux IP pour les données... La souplesse des protocoles liés à l'Internet, principalement TCP/IP[1], leur a permis d'évoluer rapidement pour faire face à la démocratisation massive de l'informatique. De plus, l'augmentation constante des débits de transmission des données a fait évoluer les réseaux IP vers de nouveaux services. La tendance actuelle est alors à la convergence des réseaux où chaque opérateur veut offrir une multitude de services à ces clients : accès à internet, téléphonie, télévision interactive, vidéo à la demande, etc...

Cette multiplication des services est accompagnée d'une exigence croissante et diversifiée quant aux besoins de qualité des transmissions. La notion de qualité de service permet de formaliser ces exigences en terme de critères de performance : bande-passante, délai moyen, délai de bout-en-bout, taux de perte des paquets... L'architecture classique des réseaux IP, développée dans les années 70 et initialement prévue pour le transfert de données, ne permet plus de garantir la qualité de service nécessaire au bon fonctionnement de ces nouveaux services. Il a donc été nécessaire de définir de nouvelles architectures réseaux pour répondre à ces nouveaux besoins. En particulier, le protocole MPLS[2] propose un compromis intéressant entre la rapidité des commutations ATM[3] et la souplesse du routage IP. De plus, il intègre une gestion de l'ingénierie de trafic et de la tolérance aux pannes. Toutefois, la maîtrise de la qualité de services reste encore aujourd'hui un défi majeur pour la recherche.

Les opérateurs de télécommunication ont toujours souhaité proposer des services performants pour des coûts les plus faibles possibles. Ils ont alors été poussés à s'intéresser et à s'impliquer dans les domaines de recherche touchant à l'optimisation des réseaux. Les évolutions considérables du monde des télécommunications ont renouvelé leur intérêt dans les études d'optimisation pour une construction et une utilisation performantes de leurs réseaux. Les enjeux économiques liés au modèle concurrentiel et le développement de réseaux de plus en plus complexes, susceptibles de transporter des flux de natures très variées, ont également motivé l'élaboration de modèles sophistiqués et d'algorithmes de résolution adaptés.

[1] *Transmission Control Protocol / Internet Protocol*
[2] *MultiProtocol Label Switching*
[3] *Asynchronous Transfer Mode*

Présentation des travaux

Dans ce contexte, ces travaux de thèse étudient l'influence des nouveaux protocoles dans la conception de réseaux haut débit tolérants aux pannes. En collaboration avec nos partenaires[4] du projet PRESTO[5], nous avons étudié les différentes problématiques soulevées par l'optimisation du routage des flux de données dans les réseaux de télécommunication. En particulier, nous avons proposé un modèle d'optimisation du routage sur un réseau multicouche MPLS sur WDM[6] prenant en compte à la fois les spécificités du routage MPLS, le partage des contraintes de tolérance aux pannes entre les couche et les contraintes de délai de bout-en-bout. Devant la complexité d'un tel modèle, nous nous sommes proposés d'étudier deux des principales difficultés.

Dans un premier temps, nous nous sommes intéressés aux spécificités du routage dans les réseaux MPLS. Dans l'approche classique du routage IP, chaque paquet est indépendant et les décisions de routage se prennent à chaque nœud intermédiaire du réseau. La notion de multiflots utilisée en optimisation des réseaux convient parfaitement pour la modélisation de ces flux. Cependant, l'engorgement des tables de routage et la gestion de la qualité de service font partie des multiples raisons qui ont motivé l'émergence de protocoles favorisant les mécanismes de commutation. C'est le cas du protocole MPLS qui prévoit que les décisions de routage soient prises au niveau du nœuds d'entrée du réseau. Ainsi, toute une partie du flux est associée à un chemin ou *Label Switch Path* (LSP) dans le réseau. Pour limiter la taille des tables de commutation, il est alors nécessaire de limiter le nombre de ces LSP. Cette contrainte forte sur le support des flux change considérablement la complexité des problèmes de routage.

Pour modéliser cette contrainte, Baier *et al.* introduisent la notion de flot k-séparable consistant en un flot routé sur au plus k chemins [BKS05]. Les travaux présentés dans la littérature portent généralement sur la complexité des problèmes de flot k-séparable et leur résolution par des algorithmes d'approximation [BKS05, KS06, MS06]. D'autres travaux étudient la réduction du nombre de chemins portant du flot par des heuristiques [BOdKK01, DM07, MTUP04]. Notre contribution porte principalement sur la modélisation des problèmes de flots k-séparables par des programmes linéaires en nombres entiers adaptés à une résolution exacte. La nature du problème nous a amenés à considérer une généralisation de l'application de la méthode du *Branch and Price* au problème du multiflot entier [AdC03, BHV00].

D'autre part, nous nous sommes intéressés à la contrainte de délai de bout-en-bout. Cette contrainte est peu étudiée dans la littérature car elle revêt un caractère non linéaire et entier très complexe [BaO06]. Cependant, nos modèles de flots k-séparables offrent des possibilités intéressantes pour gérer le délai de bout-en-bout sur les chemins actifs. Le problème de minimisation du délai de bout-en-bout maximal est alors résolu par combinaison de plusieurs méthodes : la méthode du Branch and Price, l'algorithme de Shahrokhi–Matula [SM90], l'algorithme de Frank–Wolfe [FW56],...

La complexité du problème se retrouve dans l'instabilité numérique de notre approche. C'est pourquoi nous l'avons comparée avec une résolution similaire du problème de minimisation du délai moyen. Cette démarche offre un bon compromis entre la qualité des solutions calculées et des temps de calcul acceptables. Notre approche semble alors être une piste intéressante pour de futurs travaux de recherche.

[4] l'INRIA Sophia-Antipolis et l'ENST Paris.

[5] Protection et sécurisation des nouvelles architectures de réseaux. Projet de l'ACI CNRS Sécurité Informatique. `http://www-sop.inria.fr/mascotte/PRESTO/`

[6] *Wavelength Division Multiplexing*

Organisation du manuscrit

Ce mémoire est composé de quatre chapitres dont voici un bref résumé.

Le **chapitre 1** présente plus en détail le protocole MPLS et son influence dans les problèmes de conception de réseaux. Nous introduisons également les objectifs et l'existant en matière d'optimisation dans les réseaux de télécommunication. Nous discutons ensuite le modèle de routage multicouche dans un réseau MPLS sur WDM développé avec nos partenaires du projet PRESTO. Ce modèle est le point de départ des différentes problématiques étudiées dans les chapitres suivants. Enfin, nous détaillons les notions de base de théorie des graphes et de programmation linéaire nécessaires à la compréhension de ce manuscrit.

Le **chapitre 2** définit la notion de flot k-séparable. Nous rappelons également les principaux résultats de complexité et propriétés de ces flots. C'est également l'occasion d'évoquer les relations entre les flots et les coupes. Plusieurs résultats classiques dans ce domaine sont étendus à différentes natures de flot. Enfin, nous détaillons les différents modèles que nous avons étudiés. Notre démarche a été d'étudier dans un premier temps l'ajout de cette nouvelle contrainte sur une problème classique d'optimisation. C'est pourquoi nous présentons nos modèles dans le cadre du problème du flot k-séparable maximal. Puis, nous avons généralisé notre approche à d'autres problèmes de flot k-séparable.

Le **chapitre 3** rentre dans les détails de la méthode de résolution de type branch and price. Cela comprend aussi bien la résolution du sous-problème de génération de colonnes que les différentes règles de branchement ou différentes améliorations. Enfin, nous discutons les résultats numériques obtenus sur différentes topologies de graphes.

Le **chapitre 4** présente notre étude de la contrainte de délai de bout-en-bout. Après une introduction aux différentes méthodes employées, nous détaillons leur composition au sein de notre algorithme. Les résultats numériques illustrent la complexité du problème. Pour établir une comparaison, nous avons utilisé une résolution approchée qui propose des solutions de qualité dans des temps d'exécution beaucoup plus faibles.

CHAPITRE 1

RÉSEAUX HAUT DÉBIT ET SÉCURISATION

L'essor des réseaux de télécommunications a considérablement modifié la conception des cœurs de réseaux. En augmentant le trafic, elle a rendu cruciaux les problèmes de dimensionnement et de routage. Elle a également contribué à la diversification des offres et de la tarification en fonction du service et de la qualité. En effet, tout opérateur se doit maintenant d'offrir une certaine qualité de service et d'assurer la fiabilité de son réseau. Celle-ci peut être variable en fonction du type de demande, allant d'une protection avec garantie absolue de connexion (exemple : chirurgie à distance) jusqu'à l'absence de protection. C'est pourquoi la communauté scientifique cherche à répondre aux problèmes liés à la tolérance aux pannes des nouveaux réseaux de télécommunications, aussi bien au niveau des technologies physiques qu'au niveau des protocoles et des services. Dans cet optique, l'IETF[1] a mis en place le groupe de travail MPLS (*MultiProtocol Label Switching*) chargé de développer un nouveau protocole apportant l'efficacité de la commutation de la couche de liaison à la couche réseau. D'autre part, l'actuelle gestion de la tolérance aux pannes s'effectue généralement sur des couches séparées, entraînant un surdimensionnement des réseaux. La gestion de la sécurisation sur l'ensemble des couches d'un réseau est aujourd'hui au cœur des problématiques de recherche dans le domaine des réseaux.

Ce chapitre présente les activités liées à la conception de réseaux haut débit et les différentes méthodes de sécurisation. Pour cela, nous présentons le protocole MPLS et ses implications dans les nouveaux sujets de recherche. Nous expliquons notamment pourquoi ce protocole change considérablement les problèmes classiques d'optimisation dans les réseaux. Ensuite, nous présentons un état de l'art des différentes études et des différentes approches proposées par la communauté scientifique. Puis nous détaillons un modèle de collaboration multi-couche de la sécurisation. Enfin, nous présentons les notions de bases de l'optimisation dans les réseaux nécessaires à la compréhension des modèles et algorithmes du manuscrit.

[1] *Internet Engineering Task Force*, groupe international qui participe à l'élaboration de standards pour l'Internet.

1.1 Le protocole MPLS

1.1.1 Motivation

Au début des années 1990, les télécommunications vocales et les lignes spécialisées représentaient la majeure partie du trafic. Des infrastructures réseaux telles que SONET/SDH (*Synchronous Optical NETwork/Synchronous Digital Hierarchy*) apportent justement pour ce type de trafic une garantie sur le niveau de performance et de fiabilité. Mais depuis le milieu des années 1990, le trafic de données a très rapidement augmenté, au point de dépasser le trafic vocal, et cette tendance ne semble pas faiblir. Le besoin de transporter plus de données pour un coût le plus faible possible a poussé les opérateurs de télécommunications à chercher de nouvelles solutions technologiques aussi bien au niveau des infrastructures que des protocoles utilisés.

La conception d'architecture réseaux s'opère par couches d'abstraction successives. Chaque couche correspond à une fonction particulière et utilise les services de la couche inférieure pour offrir des services à la couche supérieure. Pour assurer l'interconnexion des réseaux, l'ISO[2] a développé à la fin des années 1970 le modèle de référence OSI (*Open Systems Interconnection*). Ce modèle, constitué de sept couches, permet de délimiter les fonctions assurant le fonctionnement d'un réseau et les grands principes de coopération entre les couches. Il reste toutefois un modèle théorique puisqu'en pratique une couche peut couvrir plusieurs fonctionnalités.

Ainsi les architectures les plus courantes associent les fonctionnalités de différents protocoles (cf. figure 1.1). La couche IP (*Internet Protocol*) offre un support au développement d'applications et de services pour les utilisateurs. L'ingénierie de trafic et la qualité de service sont assurées par la couche ATM (*Asynchronous Transfer Mode*). Puis la couche SONET/SDH gère le transport des flux ATM sur le réseau optique WDM (*Wavelength Division Multiplexing*). Une telle architecture est difficile à étendre pour une grande quantité de données et s'avère inefficace d'un point de vue économique. Généralement, ces remarques sont valables pour tout type de réseaux multi-couches car une seule couche peut limiter toute l'architecture réseau dans sa mise à l'échelle, aussi bien pour des raisons techniques qu'économiques. D'autre part, l'acheminement des données est soumis à l'encapsulation des paquets : chaque couche ajoute des informations de contrôle qui induisent une surcharge du réseau et un traitement plus complexe aux noeuds intermédiaires.

C'est pourquoi la tendance actuelle est à la simplification de cet empilement de couches. Pour cela, il est nécessaire de développer des protocoles capables de mutualiser les fonctionnalités de plusieurs couches. Ainsi, MPLS apporte à la couche IP une gestion de la qualité de service et de l'ingénierie de trafic plus flexible que les solutions proposées par ATM. De même, les fonctionnalités de SONET/SDH comme les mécanismes de tolérance aux pannes sont transmises à la couche WDM grâce à des évolutions technologiques. D'autres fonctionnalités telles que le formatage des flux pour leur transmission physique n'ont pas pu être adaptées aussi rapidement, nécessitant l'utilisation d'une couche SONET réduite nommée *thin* SONET [Gro04]. Ainsi, la suppression des couches intermédiaires fait tendre les architectures réseaux actuelles vers le couple IP sur WDM [Liu02, RLA04].

Ce modèle IP/WDM a l'intérêt d'associer les capacités de croissance des réseaux optiques face à l'augmentation continue du trafic Internet, et la flexibilité des protocoles de contrôle IP. Les investis-

[2] International Standards Organisation

sements des principaux opérateurs de télécommunications semblent d'ailleurs porter actuellement sur l'apport de la fibre optique directement chez l'utilisateur. D'autre part, les offres *multi play*, Internet, téléphonie, télévision, vidéos à la demande, font converger les différents types de trafic (données, voix, vidéo) sur la couche IP.

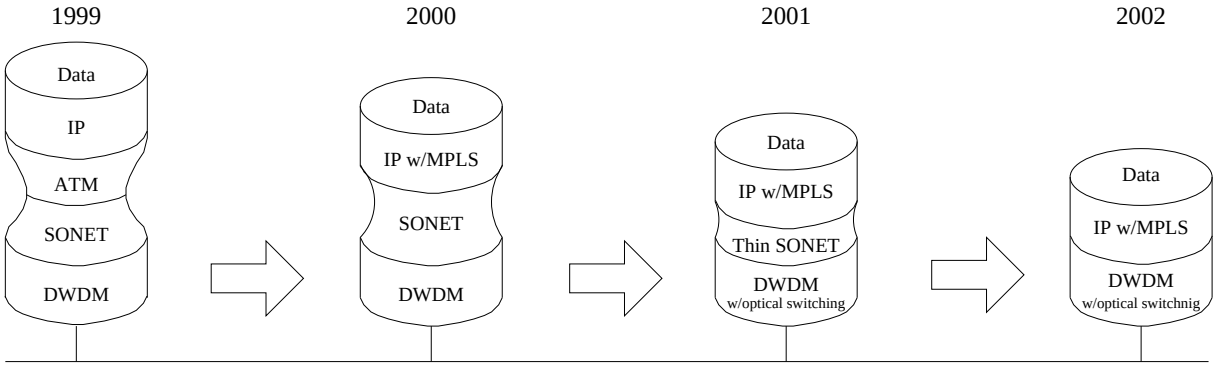


FIG. 1.1 – Evolution des architectures réseaux (inspiré de [BDL+01b]).

Historiquement, les réseaux de télécommunications se sont construits sur une grande variété de solutions technologiques et de réseaux, qui se sont développées sans nécessairement de souci d'interopérabilité ni d'interconnexion. Le regroupement de fonctionnalités évoluées au sein de la couche IP nécessite alors la coopération de ces moyens en dépit de la multiplicité des protocoles et avec le souci de faire face à la demande croissante en débit.

C'est dans ce contexte que plusieurs entreprises ont développé des solutions propriétaires comme, par exemple, « Tag Switching » de Cisco, « ARIS » d'IBM, et « Cell-Switched Router » de Toshiba. En 1997, l'IETF forme le groupe de travail MPLS afin de développer et normaliser ces différentes approches. L'un des objectifs initiaux est d'accélérer le traitement des paquets dans les équipements intermédiaires en désengorgeant les tables de routage. Avec le développement de routeurs extrêmement performants, cet aspect est passé au second plan. Depuis, c'est l'ensemble de ses fonctionnalités qui donne tout son intérêt à MPLS avec notamment les possibilités suivantes :

- L'ingénierie de trafic (*Traffic Engineering*) : MPLS permet d'assurer une certaine qualité de service et d'affecter des critères de performance à différentes classes de trafic. Ces regroupements de trafic s'établissent en fonction des contraintes de qualité à respecter, définissant une classe de service (*class of service*, CoS).
- Les réseaux privés virtuels : MPLS permet aux opérateurs de réseaux de créer des tunnels IP sans utiliser de systèmes de cryptage ou d'applications spécifiques.
- Réduction du nombre de couches : MPLS permet de déployer la plupart des fonctionnalités des protocoles comme SONET/SDH et ATM sur la couche 3, simplifiant ainsi la gestion du réseau.

1.1.2 Description du protocole MPLS

Le couple de protocoles TCP/IP (*Transmission Control Protocol / Internet Protocol*) s'est développé dans le cadre d'études sur les réseaux à transfert de paquets. Le concept de réseaux interconnectés, Internet, initié au milieu des années 70 par le DARPA (*Defence Advanced Research Projects Agency*) a

largement contribué à son essor, si bien qu'ils sont devenus un passage quasi-obligé pour les protocoles de la couche applications tels que HTTP, FTP, SMTP, etc . . . Cependant, les applications récentes ont des besoins croissants en terme de qualité de service, d'ingénierie de trafic et de maintenance des réseaux. Les chercheurs et les opérateurs de réseaux ont alors évolué vers de nouveaux protocoles comme MPLS [RVC01].

1.1.2.1 Routage IP classique

Le protocole IP effectue un routage des paquets non fiable, suivant le modèle « best effort » et dans un mode non connecté. Le service est dit non fiable car la remise n'est pas garantie. Le modèle « best effort » signifie que la remise est effectué « au mieux », sans garantie de qualité de service, sans distinction de priorité entre les paquets et sans ressources préallouées. Un paquet peut être perdu, dupliqué ou remis hors séquence. Le mode non connecté signifie que l'émetteur émet ses paquets sans prendre contact avec le récepteur. Ainsi, les paquets constituant un même message sont indépendants les uns des autres. Les paquets d'un même flux peuvent donc emprunter des chemins différents.

Pour déterminer ces chemins et acheminer l'information à travers différents sous-domaines interconnectés, le protocole IP a choisi la souplesse du routage au détriment de la puissance de la commutation. Le principe du routage nécessite que chaque paquet soit muni de l'adresse complète du destinataire. Chaque nœud de transfert intermédiaire, appelé routeur, doit déterminer le prochain relais (ou next-hop) le plus approprié pour que le paquet rallie sa destination. Pour cela, il dispose d'une table de routage qui lui permet d'associer l'adresse indiquée dans le paquet (ou plus généralement un préfixe de l'adresse) avec l'un de ses ports de sortie. Pour construire et maintenir cette table, le protocole IP peut utiliser différents protocoles de routage dynamique tels que OSPF, *Open Shortest Path First*, RIP, *Routing Information Protocol*, IS-IS, *Intermediate System to Intermediate System*, pour le routage intra-domaine ou BGP, *Border Gateway Protocol*, pour le routage inter-domaine. La souplesse de routage proposée par ces protocoles a néanmoins un coût en terme de ressources machine. En effet, les préfixes d'adresses IP stockés dans la table de routage étant de longueur variable et l'ordre n'étant pas imposé, le routeur doit examiner l'ensemble de la table de routage pour décider quelle entrée correspond au mieux à l'adresse de destination du paquet.

D'autre part, avant d'envoyer le paquet au prochain relais, le routeur doit modifier certains champs de l'en-tête IP comme les adresses MAC (*Media Access Control*) destination et source ou le temporisateur TTL (*Time To Live*). Effectué sur l'ensemble des paquets d'un même flux, cette tâche peut s'avérer gourmande en ressources. Enfin, il faut prendre en considération la taille des adresses dans la recherche de la bonne sortie de la table de routage. Dans IPv4, l'adresses de routage est établie sur 4 octets. IPv6 nécessite 16 octets alors que pour commuter un paquet ATM (*Asynchronous Transfer Mode*) 24 ou 28 bits suffisent et pour la commutation des relais de trames (*Frame relay*) entre 10 et 23 bits sont utilisés. La décision de router prend donc du temps et en pratique, on constate que, pour une puissance donnée, un commutateur atteint un débit dix à cinquante fois supérieur à celui d'un routeur. Cependant, cette différence a tendance à se réduire avec l'apparition des routeurs gigabit, résultats des progrès dans le domaine des circuits VLSI (*Very-Large-Scale Integration*) [Puj00].

1.1.2.2 Commutation de labels

L'idée fondamentale de MPLS est de combiner la souplesse du routage et la puissance de la commutation pour transporter des paquets IP (voir d'autres types) sur des sous-réseaux travaillant en mode commuté. Pour cela, il remplace les mécanismes traditionnels de routage par des mécanismes plus rapides, basés sur la commutation de *labels*. Ainsi, l'adresse de destination n'est analysée qu'une seule fois par le routeur d'entrée du domaine MPLS (routeur *ingress*). Celui-ci détermine le chemin ou LSP (*Label Switch Path*) que doit emprunter le paquet et lui impose un label de taille fixe. À l'intérieur du domaine MPLS, un nœud de transfert (LSR, *Label Switching Router*) examine le label entrant du paquet et sa table de commutation puis remplace ce label par un label sortant et transmet le paquet au relais suivant correspondant. Enfin, le dernier routeur du domaine MPLS (routeur *egress*) enlève le label pour transmettre en sortie le paquet tel qu'il est entré. Ainsi le domaine MPLS est transparent pour les paquets. Ce schéma de consultation et de transfert MPLS offre la possibilité de contrôler explicitement le routage en fonction des adresses source et destination, facilitant ainsi l'introduction de nouveaux services IP.

D'autre part, MPLS permet également d'empiler les labels en ajoutant des en-têtes MPLS aux paquets. D'un point de vue théorique, il n'y a pas de limite de taille à cette pile de labels, ce qui permet de gérer des hiérarchies de LSP. L'intérêt réside principalement dans l'agrégation de flux. Plusieurs LSP distincts peuvent avoir une partie commune. Par souci d'allégement des tables de commutation ou par utilisation de sous-réseaux intermédiaires pour connecter deux sous-réseaux distincts, il peut être nécessaire d'affecter un label commun aux paquets empruntant ces différents LSP sans pour autant perdre l'information de leur label courant. On parle alors de tunnel MPLS. Dans [KR05], cette notion de tunnel est étendue aux supports physiques (longueurs d'onde, fibre optique, ...) dans GMPLS (*Generalized MultiProtocol Label Switching*). Le protocole GMPLS est une extension de MPLS aux flux de longueurs d'onde transmises sur les fibres optiques afin de faciliter l'agrégation des flux de même destination.

1.1.2.3 Distribution de labels

Le principe de distribution de labels est de regrouper les paquets devant recevoir le même traitement au sein d'une même classe d'équivalence appelée FEC (*Forwarding Equivalence Class*). Les paquets d'une même FEC disposent d'un même label au niveau d'un LSR donné. Ils suivent donc le même chemin commuté appelé LSP. Pour une FEC donnée, MPLS propose deux méthodes pour établir un LSP : le routage implicite et le routage explicite. Le routage implicite correspond au routage saut-à-saut défini par l'IGP (*Interior Gateway Protocol*). Chaque nœud doit donc mettre en oeuvre un protocole de routage de niveau 3 comme le routage IP classique. Ainsi, les décisions de routage sont prises indépendamment les unes des autres suivant un protocole de routage dynamique. Le routage explicite est la solution MPLS pour faire de l'ingénierie de trafic. Seul le routeur *ingress* analyse l'en-tête IP et détermine le chemin de bout-en-bout. Les routeurs intermédiaires se contentent alors de commuter les paquets. Avec ce système, un opérateur peut imposer des contraintes particulières sur le flux comme, par exemple, des contraintes de qualité de service. Le LSP emprunté n'est donc plus forcément le plus court chemin. Le routeur d'entrée peut répartir le trafic sur le réseau pour éviter les points de forte congestion et fournir une meilleure qualité de service due à une prise de décision globale (ingénierie de trafic) et non locale.

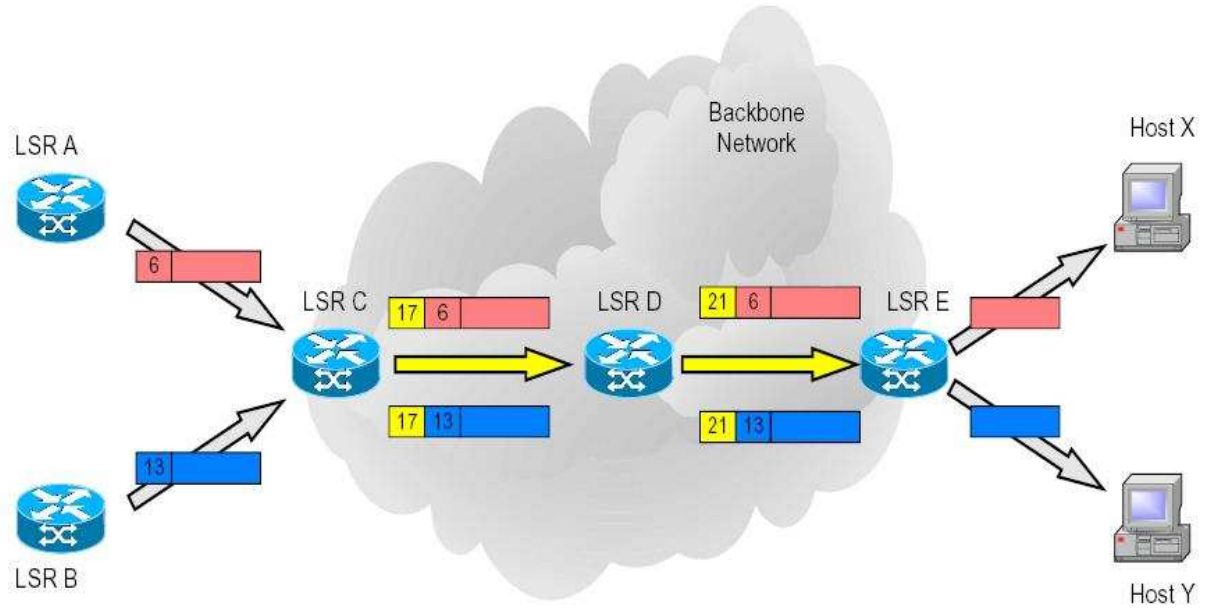


FIG. 1.2 – Routage dans les réseaux MPLS.

1.1.3 Sécurisation des réseaux MPLS

L'intérêt croissant des opérateurs réseaux pour l'ingénierie de trafic et les nouveaux services proposés par le protocole MPLS ont soulevé tout naturellement le problème de sécurisation des réseaux dans le sens de la tolérance aux pannes. En effet, si ces réseaux de nouvelle génération permettent de proposer des services évolués avec des contraintes fortes de qualité de service pour des applications comme la chirurgie à distance, il est indispensable que ces contraintes soient respectées. C'est pourquoi les problématiques de tolérance aux pannes ont fait l'objet de nombreuses études et d'approches différentes, généralement séparées en deux catégories : la restauration et la protection.

1.1.3.1 Restauration

La restauration correspond aux solutions réactives. Après détection et notification de la panne, les LSR activent les protocoles de re-routage dynamique pour déterminer de nouveaux chemins permettant de transférer le flux. Ces solutions permettent de prendre en compte la nouvelle topologie et l'état du réseau au moment de la panne pour mettre à jour les tables de commutation. Cela les rend plus flexibles mais le temps de réaction est plus long que pour la protection, ce qui peut être préjudiciable pour les flux à haute contrainte de qualité de service. Les algorithmes mis en jeu doivent donc être extrêmement rapides et apporter de bonnes solutions de re-routage en s'adaptant au fur et à mesure aux nouvelles données. Il paraît donc naturel de penser que ces algorithmes proposeront des solutions de moins bonne qualité que des algorithmes préventifs préparant de nouvelles routes à l'avance et avec des contraintes de temps de calcul moins importantes.

1.1.3.2 Protection

La protection correspond aux solutions proactives. Elles agissent en prévision de pannes en déterminant par avance les chemins de re-routage appelé LSP de *backup* ou chemin de protection. Pour être efficaces, ces LSP de backup doivent être alloués comme les LSP principaux et prêts à accueillir le trafic rerouté qui leur est dédié si nécessaire. La protection est généralement globale. Si l'on considère l'ensemble des scénarios de pannes possibles, ceci augmente fortement le nombre de LSP de backup et par là même la capacité nécessaire du réseau pour assurer à la fois le routage du trafic et sa protection. Le principal inconvénient vient du fait qu'entre l'état sans panne et l'état avec panne, un grand nombre de reconfigurations peuvent s'avérer nécessaires. L'impact d'ordre technique dans la modification des tables de commutation peut alors engendrer un délai non négligeable avant la mise en place des chemins de protection.

Un grand nombre de stratégies de protection ont été proposées dans la littérature et la plus grande partie se limite à une couche unique subissant une unique panne d'un lien. Parmi ces stratégies, il convient de distinguer certaines approches. Par exemple, certaines méthodes favorisent la protection locale ou protection par arête qui consiste à déterminer et réserver un chemin contournant chaque lien du réseau. À l'opposé, d'autres méthodes préfèrent la protection par chemin qui consiste à prévoir pour chaque FEC deux chemins : un LSP principal et un LSP de backup. Chaque approche a ses avantages et ses inconvénients. Néanmoins, la protection par chemin est plus utilisée que la protection par arête car elle utilise beaucoup moins de capacité même si son délai de mise en place est plus long [IMG98, RM99].

De même, on distingue habituellement la protection dédiée de la protection partagée. La première signifie que le chemin de protection, alloué pour remplacer un LSP principal donné en cas de panne, ne doit pas être utilisé dans un autre contexte. À l'inverse, la protection partagée permet de mutualiser des LSP de backup entre plusieurs LSP principaux qui ne peuvent pas être affectés par la même panne. Cette approche permet bien entendu de limiter le surdimensionnement du réseau en partageant de la capacité de secours. Cependant, la protection dédiée permet d'envoyer la même information sur les chemins principaux et les chemins de backup. Cette stratégie appelée protection `1+1` permet d'assurer la continuité du trafic. En effet, le nœud de destination reçoit les paquets en double garantissant leur réception en cas de panne. À l'opposé, la protection `1:1` consiste à n'activer le chemin de secours qu'après la détection et la notification de la panne. Cette stratégie permet de diminuer la quantité de trafic circulant sur le réseau. Elle permet également d'utiliser la protection partagée. Dans ce cas, le principe a été généralisé à la protection `M:N` signifiant que `M` chemins principaux sont protégés par `N` chemins de secours. Notons enfin qu'en pratique, les opérateurs font circuler sur ces canaux des flux non prioritaires. Ces flux peuvent être interrompus à tout moment et remplacés par des flux de protection, le temps que la situation revienne à la normale.

Notons que cette description des principales stratégies de protection et restauration peut être étendue au protocole GMPLS (Generalized MPLS) [BDL$^+$01a] et aux réseaux optiques (couche transport) [MM00, PKGK03, SRM02].

Toutefois, un réseau de communications n'est pas constitué de la seule couche physique. Généralement, une entreprise fournit le support physique et une seconde couche, via un protocole de communications, exploite le support physique pour véhiculer des informations. En se plaçant dans le cas où les deux couches appartiennent à la même société, il peut être intéressant de considérer tous les problèmes énumérés précédemment dans les deux couches en même temps. La stratégie de routage et de protection est

notamment changée, puisqu'il peut y avoir partage de la bande passante réservée à la sécurisation entre les deux couches, ce qui se traduit par des économies financières significatives. Ensuite, concernant la restauration, il peut être intéressant de considérer des stratégies qui agissent au niveau des deux couches pour rétablir une communication, là aussi en partageant les moyens. Dans certains cas, on s'aperçoit qu'intervenir dans telle ou telle couche est plus économique et plus rapide. Cet aspect multicouche se traduit généralement par un problème de dimensionnement couplé à un problème de routage. Différents travaux sont là aussi disponibles : [CdMD+02, DGA+99, ISS03, dMCG+02, QMJ03, SS03, dSVG01].

1.1.4 Qualité de Service

L'UIT (*Union Internationale des Télécommunications*) a défini la notion de Qualité de Service, ou QoS (*Quality Of Service*), par la recommandation E.800 comme « l'effet collectif du service de performance qui détermine le degré de satisfaction d'un utilisateur du système ». Cette définition très générale lie la QoS à des appréciations subjectives des utilisateurs. Cependant, la qualité de service est généralement traduite en terme de critères de performance des transmissions de données. Traditionnellement, les opérateurs de réseaux prennent en considération les critères de QoS suivants :

Le débit : il représente une quantité d'informations qui circule, par unité de temps, entre une source et une destination. Il peut être calculé en moyenne sur toute la durée de la transmission. Cela correspond tout logiquement au débit moyen. D'autre part, le débit pic, communément appelé bande passante, désigne le taux de transfert maximum pouvant être maintenu entre les deux extrémités de la transmission.

Le délai : il représente la durée de transfert d'un paquet entre deux routeurs du réseau. Celle-ci inclut aussi bien le délai de propagation physique le long du ou des liens empruntés que le délai de transmission induit par la mise en file d'attente dans les nœuds intermédiaires. La qualité de propagation des cables réseaux et la charge actuelle des réseaux font qu'il est classique de voir dans la littérature le délai de propagation négligé par rapport au délai de transmission. En outre, le délai est habituellement calculé séparément le long de chaque lien du réseau pour effectuer une moyenne. On parle alors de délai moyen à opposer au délai de bout-en-bout qui considère le délai le long d'un chemin entier, de la source jusqu'à la destination.

La congestion : elle correspond au rapport entre la quantité d'information circulant sur un lien (débit) et la capacité de ce lien. Si elle est élevée, cela signifie que le routeur de destination du lien atteint sa capacité de traitement des paquets. En conséquence, sa file d'attente s'allonge et le délai de transmission également. Lorsque le débit tend vers la capacité de traitement du lien, le délai de transmission tend vers l'infini. Ce phénomène d'effet barrière assure le respect des capacités dans les solutions de routage calculées par des algorithmes cherchant à minimiser ou limiter la congestion maximale du réseau. Lorsque qu'un lien à lui seul limite tout ou partie d'un réseau, on parle de goulot d'étranglement. Ces goulots apparaissent souvent lorsque le routage des paquets utilise un simple plus court chemin passant par le même lien qui sature devant l'affluence de trafic. Le délai de transfert des paquets s'en ressent nettement et la qualité de service chute. C'est pourquoi il est nécessaire de mettre en place des protocoles de routage ou de re-routage (en cas de panne) plus adaptés aux besoins actuels de QoS.

La gigue : elle désigne un phénomène de fluctuation du signal au cours de la transmission. Elle est définie comme une variation du délai pour un même débit. Une trop forte gigue affecte en particulier les flux multimédias temps-réels en détruisant les relations temporelles des trains de données

transmis régulièrement par le flux multimédia, entravant ensuite la compréhension du flux par le récepteur.

Le taux de perte : il est défini comme le ratio entre la quantité de données perdues par rapport à la quantité de données émises par la source.

En général, il convient de distinguer les services qui peuvent s'adapter vis-à-vis de la congestion du réseau des services qui nécessitent des garanties sur un ou plusieurs critères de QoS pour fonctionner. Les premiers sont des services dis "élastiques". Il s'agit typiquement des services IP classiques tels que le transfert de fichier ou les services de messagerie. Certes, la satisfaction de l'utilisateur est meilleure lorsque la bande passante est la plus élevée possible, lorsque les délais sont faibles et qu'il n'y a pas de pertes. Mais ces services n'ont pas d'exigences fortes en terme de QoS. Ils pourront toujours s'adapter à une dégradation du réseau.

À l'inverse, certains services comme les services multimédias ont besoin d'une qualité de service minimale pour fonctionner. Ils ne peuvent pas ou peuvent difficilement s'adapter à l'état du réseau. Par exemple, les services de diffusion de flux audio ou vidéo (*streaming*) ont besoin de garantie d'un débit minimal car en deçà, leur qualité se dégrade fortement et le service devient inutilisable. Ces services sont également sensibles, parfois dans une moindre mesure, à d'autres critères de QoS. Les services interactifs, par exemple, sont particulièrement exigeants en terme de délai : il faut que le service réagisse au plus vite à l'action de l'utilisateur. La téléphonie, pour sa part, demande à la fois des garanties de débit et de délai au réseau, mais peut tolérer quelques pertes de paquets, contrairement à la vidéo.

Pour répondre à ces exigences, le protocole MPLS propose deux mises en œuvre possibles de la QoS. Sur un même LSP, les trafics sont regroupés par classe de service (CoS, *Class of Service*) suivant les critères de QoS qu'ils requièrent. Puis, les LSR sont en charge de traiter les paquets différemment selon leur CoS. Il est également possible de créer plusieurs LSP entre une même paire de routeurs d'entrée et de sortie du domaine MPLS. Chaque LSP possède ses propres critères d'acheminement suivant la priorité du trafic devant l'emprunter. Ainsi, le traitement de la différentiation de service est fait une seule fois et les LSR intermédiaires peuvent se contenter de commuter les paquets sans analyser la classe de service.

1.2 Optimisation des réseaux de télécommunication

Ces travaux de thèse se placent dans le cadre général de la conception de réseaux avec contraintes de sécurisation. Ce domaine regroupe un nombre important de problématiques diverses dont l'objectif principal est de proposer des stratégies permettant à un réseau de rester opérationnel même en cas de pannes (qui peuvent prendre des formes diverses). Mais cette problématique de sécurisation ne peut pas être considérée seule. En effet, il faut également prendre en compte des critères de qualité de service, très importants pour les clients, et des critères économiques de performance du réseau. Bien que notre étude se place principalement dans le cadre des réseaux MPLS présentés dans la section 1.1, il est nécessaire de prendre en considération toute la complexité de l'optimisation des réseaux en général.

1.2.1 Présentation générale

Les opérateurs de télécommunication ont toujours souhaité proposer des services performants pour des coûts les plus faibles possibles. Les domaines de recherche touchant à l'optimisation des réseaux ont apporté pour cela des outils et des méthodes efficaces. Les évolutions considérables du monde des télécommunications ont renouvelé les efforts de recherche dans les études d'optimisation pour une construction et une utilisation performantes des réseaux. Les enjeux économiques liés au modèle concurrentiel et le développement de réseaux de plus en plus complexes, susceptibles de transporter des flux de natures très variées, ont également motivé l'élaboration de modèles sophistiqués et d'algorithmes de résolution adaptés.

De par leur structure, les réseaux sont naturellement représentés sous forme de graphes. Ils constituent ainsi un terrain d'application très fertile à la théorie des graphes. Par exemple, les problématiques liées à la topologie des réseaux illustrent parfaitement certains problèmes théoriques comme les problèmes d'arbres couvrants, de graphes planaires ou de biconnexité. De même, les problèmes de routage dans les réseaux peuvent s'apparenter à des problèmes, dans un graphe, de plus court chemin, de parcours eulérien, de cycles, etc ...

D'autre part, le développement rapide de la programmation linéaire a offert de nombreuses applications possibles des modèles linéaires aux problèmes d'optimisation de réseaux. Les progrès constants réalisés par les solveurs ont même permis de traiter des instances de plus en plus proches des problèmes réels (plus grandes, plus complexes). Ainsi, ils ont contribué en grande partie à l'obtention de résultats significatifs concernant l'optimisation de réseaux avec demandes mono-routés (un seul chemin par demande) [BHV00, Kle96, FT00], avec routage selon plus courts chemins [BAMLG01, BAG03, Gou01] ou l'optimisation des réseaux sécurisés [BM00, MV04]. Aujourd'hui, ces approches semblent également s'imposer pour la conception et l'optimisation de réseaux IP sur WDM, contrôlés par MPLS.

En particulier, un flux de données entre deux nœuds du réseaux se modélise simplement par la notion de flot. Ainsi, les flux entre chaque paire origine-destination sont représentés par des flots distincts qui se partagent les ressources du réseau. L'ensemble de ces flots est appelé multiflot. Les problèmes de multiflot (*Multicommodity Flow*) se sont ainsi imposés comme l'un des éléments essentiels des problèmes d'optimisation de réseaux de télécommunications [Ken78, LC00, Min89].

Dans [GLON04], les auteurs distinguent principalement trois grandes catégories de problèmes d'optimisation des réseaux de télécommunications. Ces trois catégories correspondent aux différentes échelles de temps relatives à la vie d'un réseau de télécommunication. À long terme, les problèmes de planifications reposent sur des problèmes de synthèse de réseaux (*Network Design*) [BMW89, Bar96]. Ces problèmes cherchent principalement à définir la topologie du réseau : choix des nœuds, des liens entre les nœuds, des points d'accès ou d'interconnexion. L'objectif est principalement de déterminer une structure du réseau à moindre coût et adaptée à une utilisation performante du réseau durant une période relativement importante. Typiquement, ces problèmes ne s'intéressent pas directement à l'écoulement du trafic mais à des caractéristiques structurelles susceptibles de favoriser cet écoulement. Par exemple, les problématiques de sécurisation peuvent prendre ici la forme de contraintes sur le degré de connexité [Mah94].

À moyen terme, le déploiement d'un réseau consiste à installer des capacités sur une topologie donnée, de manière à permettre l'écoulement d'un trafic. Les problématiques étudiées à cette étape de la

vie du réseau relève du dimensionnement du réseau. Étant donné l'impact de la topologie sur les coûts de déploiement, de nombreuses études traitent des problèmes d'optimisation combinant à la fois la synthèse et le dimensionnement du réseau [AR02, BM00]. Ces problèmes intègrent souvent des contraintes liées aux équipements et aux règles d'ingénierie comme des politiques de protection, des contraintes de délai ou de longueur de chemin. Celles-ci influent sur la topologie du réseau (degré de connexité, diamètre) ou sur le mode d'écoulement de la demande (contraintes de routage). La difficulté de ces problèmes vient principalement de la combinatoire liée aux choix des topologies. Les progrès récents des techniques d'optimisation combinatoire laissent à penser que la résolution de ces problèmes devrait également s'améliorer.

Enfin, à court terme, la gestion opérationnelle des réseaux concerne l'allocation des ressources aux différents services demandant à router du trafic. Les problèmes qui en découlent influent naturellement sur le dimensionnement du réseau ou même de sa topologie. Ils interviennent donc souvent comme sous-problèmes dans les études de synthèse de réseaux. Par exemple, la recherche d'une meilleure performance incite à la comparaison des différents mécanismes de routage. Cependant, les problèmes d'allocation de ressources s'appliquent également dans les cas où le réseau n'est plus extensible et où il faut gérer les écarts entre les prévisions et le trafic réel. Ils visent aussi à apporter des solutions de re-routage en réponse aux perturbations du réseau.

Toutes ces problématiques sont bien entendu complémentaires et interagissent ensemble mais leur complexité fait qu'en pratique, elles sont habituellement traitées séparément. D'autre part, il existe de nombreux thèmes transversaux étudiés à chaque étape de la vie du réseau, à l'instar des problèmes de sécurisation des réseaux qui ont fait l'objet d'un grand nombre d'études.

Les travaux de thèse présentés dans ce manuscrit portent principalement sur des problèmes de routage avec contraintes de qualité de service. Ils rentrent dans la catégorie des problèmes d'allocation de ressources.

1.2.2 Problèmes d'allocation de ressources

Les problèmes d'allocation de ressources traitent de l'écoulement de la demande dans un réseau donné. Ils considèrent donc généralement à la fois des problèmes de routage, comme le partage des flux, le choix des routes, et des problèmes de qualité de service associés à l'écoulement du trafic, tels que des garanties de débit ou de délai. Or, la manière dont les flux circulent dans le réseau a une influence non négligeable sur le dimensionnement du réseau ou même sur sa topologie. C'est pourquoi ces problèmes peuvent faire partie d'études plus globales de synthèse de réseaux. Mais devant la complexité de telles études, les modèles utilisés sont des représentations approximatives, simplifiés de l'écoulement du trafic et des contraintes de qualité associées, se contentant de déterminer des solutions de routage admissibles pour le réseau et/ou les contraintes imposées.

En revanche, dans des études plus spécifiques, lorsque la topologie et le dimensionnement du réseau sont fixés, les modèles employés représentent plus finement le routage afin de permettre une optimisation de la qualité de service. Pour traiter ces problèmes, un grand nombre d'outils pratiques et théoriques sont disponibles. En particulier, les multiflots offrent une modélisation simple et efficace de la circulation des flux dans le réseau. Ils représentent les décisions de routage localement à chaque nœud du graphe, transitant les flux entrants sur les liens sortants du routeur. Un multiflot ne modélise donc pas directement

le parcours d'un paquet dans le réseau mais l'utilisation globale des bandes passantes de chaque lien, les interactions entre les flux issus des différentes demandes et leur influence sur la qualité de service. Vu l'importance des multiflots dans le domaine de l'optimisation des réseaux, nous en proposons une description détaillée en section 1.4.3.3.

La qualité de service peut revêtir des aspects bien différents (débit, délai, pertes de paquets, ... cf. section 1.1.4). Traditionnellement, quel que soit le critère de qualité utilisé, l'étude porte sur une moyenne statistique sur tout le réseau. Ces résultats statistiques se basent généralement sur des hypothèses simplificatrices comme l'hypothèse de Kleinrock, très utilisée en optimisation des réseaux de télécommunication. Dans [Kle64], Kleinrock modélise les systèmes d'attentes du réseau comme des files d'attente M/M/1 indépendantes. Cette approximation est fausse en pratique car les files d'attente des routeurs sont reliées les unes aux autres par le réseau et s'influencent donc mutuellement. Elle offre cependant une estimation statistique globale sur le réseau qui donne des résultats intéressants en pratique.

Un des défis récents de l'optimisation dans les réseaux est de ne plus prendre en compte uniquement un aspect statistique moyen sur la globalité du réseau et des flux qui le parcourent mais de considérer la vie des paquets tout au long du chemin qu'ils empruntent. En effet, même avec un délai moyen faible sur les liens du réseau, la distribution des délais de bout-en-bout des paquets peut être très étalée. Ainsi certains paquets peuvent parcourir le réseau très rapidement et d'autres, au contraire, très lentement. Or les contraintes de qualité de service actuelles, notamment pour le délai, cherchent à s'assurer qu'un certain critère, par exemple le délai de bout-en-bout, ne dépasse pas un seuil donné.

Sous l'hypothèse de Kleinrock, les délais de propagation sur chaque lien peuvent se modéliser par des fonctions convexes [Kle64]. Cette modélisation donne une estimation des délais moyens satisfaisante en pratique. Il parait alors naturel de modéliser les délais de bout-en-bout en sommant les délais sur chaque arc du chemin. Cependant, même si cette approche reste valable en théorie sous l'hypothèse forte d'indépendance de Kleinrock, aucune étude n'a discuté de sa qualité d'estimateur en pratique. Dans [CB00], les auteurs proposent une borne sur le délai de bout-en-bout sous une hypothèse également très forte et qui semble discutable en pratique (la congestion maximale du réseau doit être inférieure à l'inverse du nombre maximal de *hops* sur l'ensemble des chemins parcourus par les paquets). En attendant des études plus complètes sur l'estimation de ce délai de bout-en-bout, les rares travaux à notre connaissance sur l'optimisation de ce délai utilisent l'approche classique de Kleinrock (cf. [BaO06]).

D'autre part, avec des protocoles à commutation de paquets tels que MPLS, le routage des paquets ne s'établit plus de routeur en routeur mais globalement sur le réseau en déterminant les chemins à emprunter. Les modèles par multiflots montrent pour ces problèmes une certaine limite car ils représentent l'agrégation de ces chemins d'où résulte une perte d'information. Un routage des paquets par chemins induit une limitation sur le nombre de chemins utilisés afin d'alléger les tables de commutation. Or l'optimisation de certains critères de qualité de service, tels que le délai moyen, tend à une dispersion des flux qui se répartissent sur l'ensemble du réseau.

Les travaux de thèse que nous présentons dans ce manuscrit se proposent d'étudier ces problématiques de limitation du nombre de chemins empruntés et de délai de bout-en-bout en utilisant la notion de flots k-séparables (cf. chapitre 2). Les méthodes de résolution employées proviennent des méthodes d'optimisation combinatoire et d'optimisation convexe couramment utilisées en optimisation des réseaux de télécommunication.

1.2.3 Les méthodes de résolution

Un grand nombre de méthodes sont utilisées pour résoudre les différents problèmes d'optimisation des réseaux. On distingue là aussi plusieurs familles d'algorithmes apportant des réponses différentes. Les études présentées dans ce manuscrit de thèse concernent essentiellement des algorithmes exacts. Ce sont généralement des méthodes itératives convergeant, au moins de façon théorique, vers une solution optimale. En effet, les limitations physiques inhérentes aux ordinateurs ne permettent pas d'effectuer des calculs exacts. L'accumulation des erreurs de calcul peut parfois s'avérer catastrophique pour la stabilité de la méthode employée. Parmi ces algorithmes, on distingue généralement différents types de méthodes suivant les caractéristiques du problème ou du modèle à résoudre.

Par exemple, les modèles de multiflots, fréquemment utilisés en optimisation des réseaux de télécommunication, relèvent habituellement de la programmation linéaire. Les premiers résultats théoriques pour la résolution de ces modèles datent de la fin des années 1940 avec notamment l'algorithme du simplexe par G. Dantzig et la théorie de la dualité par J. Von Neumann. Depuis, le traitement des programmes linéaires n'a cessé de s'améliorer avec, en particulier, des algorithmes polynomiaux dans le pire des cas. Le premier algorithme de ce type, proposé par L. Khachiyan dans [Kha79], est basé sur la méthode des ellipsoïdes en optimisation non-linéaire. Cependant, les performances de cet algorithme sont décevantes en pratique. Cependant, cette méthode mène à d'intéressants résultats en optimisation combinatoire : l'optimisation sur un polyèdre ne dépend pas du nombre de contraintes du système décrivant le polyèdre, mais plutôt du problème dit de séparation lié à ce système [GLS81]. Ce problème consiste, étant donné une solution, à déterminer si cette solution vérifie le système, et sinon à trouver une contrainte du système qu'elle viole. Autre résultat important, N. Karmarkar a présenté dans [Kar84] un algorithme de points intérieurs performant à la fois en théorie et en pratique.

Beaucoup de problèmes nécessitent de modéliser des décisions telles que les choix des nœuds et des arcs composant la topologie du graphe dans un problème de synthèse de réseau. Par conséquent, les modèles intègrent des variables discrètes empêchant l'application directe de l'algorithme du simplexe qui ne peut donner qu'une solution fractionnaire. La plupart des méthodes exactes résolvant ces modèles sont basées sur le principe de branchements et évaluations (*Branch & Bound*) [LD60].

Même si les performances des solveurs et des ordinateurs n'ont cessé de s'améliorer, la résolution des grands programmes linéaires a toujours été un des défis majeurs de l'optimisation combinatoire en générale et de l'optimisation des réseaux en particulier. Plusieurs techniques de décomposition ont été développées pour tirer parti des structures particulières de certains modèles, notamment lorsque des blocs se distinguent à l'intérieur de la matrice des contraintes [DW60, Ben62, GK87]. Les méthodes de *Branch & Bound* ont alors été complétées par des techniques de génération de coupes (*Branch & Cut*), des techniques de génération de colonnes (*Branch & Price*) ou les deux (*Branch & Cut & Price*). Ainsi, dans [AdC03, BHV00], les auteurs proposent d'appliquer la méthode du *Branch & Price* à un problème de routage où chaque demande doit être routée sur un seul chemin.

L'optimisation de la qualité de service a introduit, dans certains cas, des non-linéarités dans le modèles. L'un des exemples le plus étudiés concerne la modélisation des délais de propagation sous forme de fonctions convexes [Kle64]. Les problèmes d'optimisation de multiflots convexes ont fait l'objet de nombreuses études [OMV00]. Parmi les méthodes les plus utilisées, la méthode de déviation de flot a profité de sa relative simplicité [FGK73]. Il s'agit d'une adaptation de l'algorithme de Frank-Wolfe [FW56] tirant parti de la structure des problèmes de multiflots. Cependant, cette méthode a tendance à

disperser les flots sur le réseau en gardant une quantité de flot positive sur tous les chemins générés. La décomposition de ces flots aboutit alors à l'utilisation d'un grand nombre de chemins, ce qui peut être gênant pour certains types de routages. Un tel comportement peut être amélioré par des méthodes de projection qui font décroître le flot sur certains chemins [BG92]. D'autres approches heuristiques cherchent également à limiter la dispersion des flots [BOdKK01, DM07].

Depuis, des techniques de l'optimisation convexe se sont montrées plus efficaces pour aborder ces problèmes. Par exemple, nous pouvons citer la méthode des faisceaux [LNN95], la méthode du centre analytique (ACCPM) [GV02] et les méthodes proximales [Mah02, OMV00].

Les approches « classiques », cherchant à optimiser la moyenne d'un critère pour l'ensemble d'un réseau, ont montré leur limite dans leur incapacité à contenir la disparité entre les différentes valeurs du critère sur chaque partie du réseau. C'est pourquoi des approches plus récentes de type « minmax » s'attache à considérer la pire réalisation du critère optimisé. Le cas des problèmes de flots concurrents illustre bien cette approche en montrant le lien entre la minimisation de la congestion maximale et la maximisation de la charge. Il est intéressant de remarquer que malgré le caractère linéaire des modèles, les principales approches ont utilisé des fonctions de pénalités convexes. D'une part, l'algorithme de Shahrokhi et Matula a été l'un des premiers algorithmes d'approximation polynomiale pour résoudre le problème de maximisation de la charge pour des flots concurrents avec des capacités uniformes [SM90]. D'autre part, Bienstock et Raskina ont proposé une adaptation de la méthode de déviation de flot basée sur la fonction de Kleinrock modélisant les délais de propagation. Cette méthode a abouti à un algorithme d'approximation polynomial pour résoudre le problème de minimisation de la congestion maximale [BR02]. Ces deux approches complémentaires ont donné lieu à une généralisation de l'utilisation de fonctions potentiels pour résoudre approximativement des programmes linéaires modélisant des problèmes de type « minmax » [Bie02].

1.3 Sécurisation dans les réseaux multicouches

À l'occasion de l'ACI Sécurité informatique[3], nous avons participé au projet PRESTO[4] dont l'objectif est de répondre aux problèmes de protection et sécurisation (tolérance aux pannes) posés par l'évolution des réseaux. Nous avons travaillé en collaboration avec l'INRIA Sophia-Antipolis et l'ENST Paris. L'association de trois équipes aux cultures différentes mais complémentaires et maîtrisant des outils divers (réseaux et protocoles, graphes et algorithmique, et optimisation) a permis de relever certains défis soulevés par ces nouvelles problématiques.

Dans le cadre de ce projet, nous avons été amenés à élaborer un modèle collaboratif pour la sécurisation d'un réseau multicouche. L'analyse des relations entre les couches optiques et MPLS ainsi que la flexibilité de la protection sur la couche optique nous a permis de développer un modèle mathématique pour le routage de la couche MPLS incluant la collaboration entre les couches, des critères de qualité de service et des critères économiques.

[3] Action Concertée Incitative conduite par le ministère délégué à la recherche et aux nouvelles technologies, le CNRS, l'INRIA et la DGA (`http://acisi.loria.fr/`).

[4] Protection et sécurisation des nouvelles architectures de réseaux.
`http://www-sop.inria.fr/mascotte/PRESTO/`

1.3.1 Les réseaux multicouches

Au niveau de la couche optique, les problématiques traitées se posent sur un graphe physique dont les nœuds représentent les routeurs optiques WDM et dont les cables sont composés de plusieurs fibres regroupant chacune un nombre donné de longueurs d'onde. L'opérateur doit établir pour chaque paire origine-destination un certain nombre de routes, c'est-à-dire des chemins le long desquels l'information est commutée au niveau des brasseurs optiques et ne remonte donc pas au niveau MPLS. Ces routes sont utilisées par le plan de commande MPLS pour transférer les paquets entre les routeurs/commutateurs MPLS. Ainsi, le fait que le signal transportant un paquet passe par plusieurs nœuds optiques intermédiaires avant de remonter au niveau MPLS est totalement transparent sur la couche MPLS. Les routes optiques ou *lightpaths* offrent donc une capacité de débit entre deux nœuds du domaine MPLS. L'établissement de ces routes peut alors être considéré comme une vision à long terme pour laquelle l'opérateur essaie de prévoir suffisamment de bande passante pour ses clients.

D'autre part nous rappelons que la notion de délai telle qu'on l'entend dans les problèmes de routage IP classique n'a pas de sens en optique puisqu'elle est principalement liée à l'attente des paquets IP dans les routeurs avant leur traitement et leur réexpedition. Or un flux optique ne peut technologiquement pas être stocké avant d'être traité. Les problèmes de qualité de service seront pris en considération uniquement sur la couche IP.

Au niveau de la couche MPLS, le problème de routage consiste à utiliser au mieux les capacités offertes par la couche optique pour répondre à la demande de débit entre les routeurs *ingress* et *egress* du domaine MPLS. Pour cela, il est également nécessaire d'établir des routes, appelés LSP dans la terminologie MPLS. Le principe est le même que pour la couche optique et ces LSP offrent une capacité de débit pour le réseau logique MPLS qui exprime les demandes entre les routeurs en bordure du domaine (voir figure 1.3). La mise en place de ces LSP correspond donc à une vision à moyen ou court terme car ils doivent suivre l'évolution des demandes.

Une telle architecture favorise un contrôle séparé de chaque couche où les solutions de routage sont indépendantes les unes des autres. Or les problématiques de routage incorporent maintenant des objectifs de protection. Ainsi, le routage de la couche MPLS prévoit des chemins de protection en supplément des chemins utilisés. Par conséquent, la demande en capacité sur la couche optique augmente, obligeant les opérateurs à surdimensionner leur réseau physique. De plus, toute cette capacité est assurée par des lightpaths au niveau WDM qui sont également protégés par des chemins de backup dans le cas d'un modèle de sécurisation séparé. Les coûts d'installation augmentent à leur tour pour une protection dédoublée qui n'est pas forcément nécessaire. Économiquement, il est donc indispensable de mettre en place un contrôle partagé de la sécurisation.

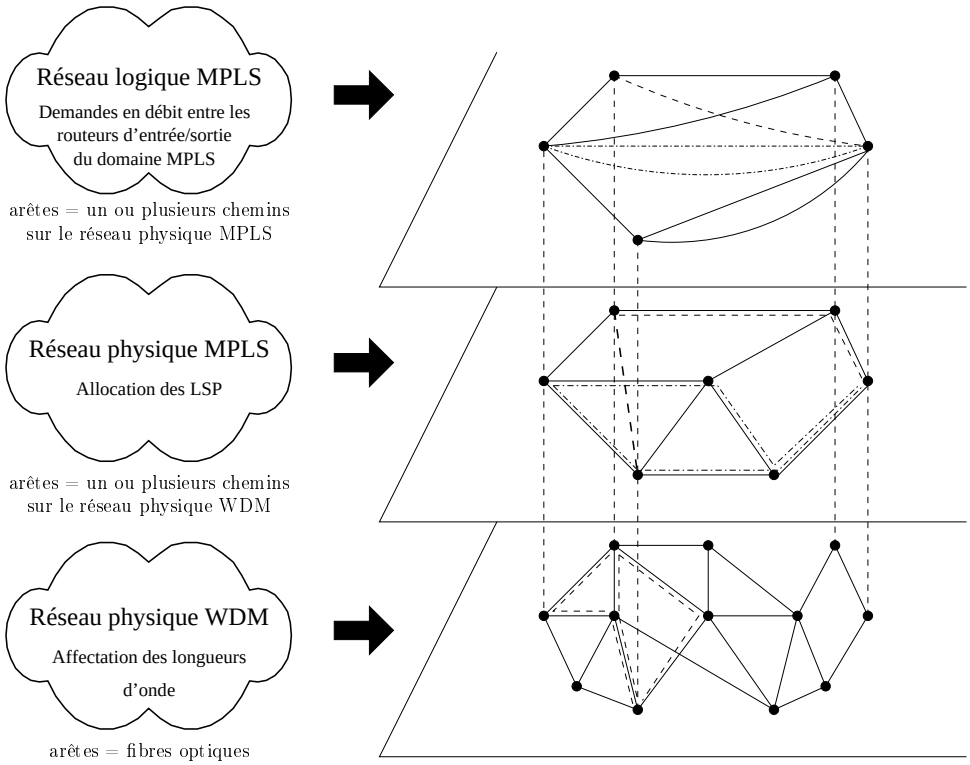
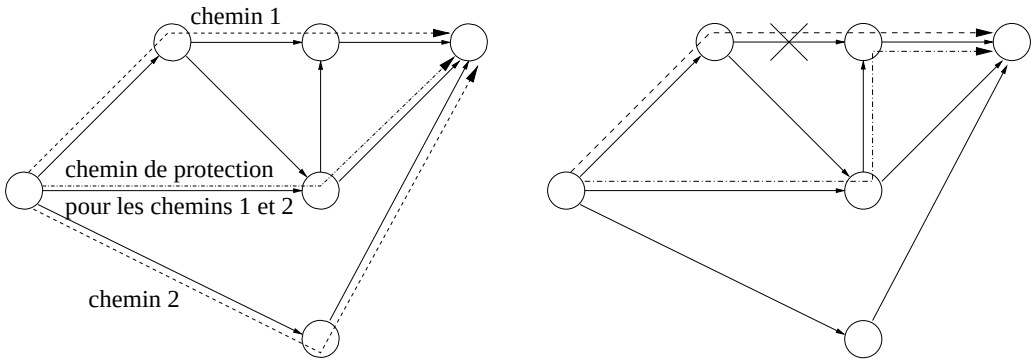


FIG. 1.3 – Réseau multicouche.

1.3.2 La protection sur la couche optique

Dans la pratique, les routeurs sont fortement sécurisés et très souvent dédoublés. Nous négligerons donc les pannes de nœud. De plus, nous ne considérerons qu'une seule panne d'arc (de lien) possible en même temps. Au niveau de la couche WDM, plusieurs stratégies sont possibles. La plus simple est de ne considérer qu'un seul chemin de protection (ou chemin de *backup*) pour chaque chemin principal. Pour assurer le routage quelle que soit la panne d'un arc du chemin principal, il faut que le chemin de protection soit arcs-disjoint par rapport au chemin principal. Cette stratégie permet de limiter la taille des tables de routage en ne considérant qu'un seul chemin de *backup* par chemin principal mais elle cause un surdimensionnement du réseau.

Dans le cadre de la couche optique, nous préférerons donc une approche de la sécurisation par protection et dépendante de la panne. Il s'agit de prévoir un chemin de protection pour tout arc et pour tout chemin principal passant par cet arc. Cette stratégie est plus gourmande en ressource mémoire car pour un chemin principal donné, plusieurs chemins de protection pourront être utilisés suivant l'arc en panne. D'un autre coté, elle permet de mettre en place différents partages de la capacité de protection. Par exemple, deux chemins principaux touchés par des pannes disjointes, en différents moments, peuvent avoir le même chemin de protection. D'autre part, puisque l'on considère un chemin de protection par panne, celui-ci peut réutiliser la capacité du chemin principal. La figure 1.4 illustre ces deux exemples de partage de capacité de protection.



(a) Pannes disjointes (b) Réutilisation de la capacité principale

FIG. 1.4 – Partage de la capacité de protection.

De plus, cette approche apporte une certaine flexibilité dans la sécurisation de la couche optique. En effet, rien n'impose que tous les chemins principaux soient protégés pour toutes les pannes possibles. Une telle protection partielle va permettre d'établir une collaboration avec la couche supérieure pour assurer une protection totale sur la globalité du réseau.

1.3.3 La collaboration entre les couches

Dans un premier temps, le problème de sécurisation a été considéré séparément sur chaque couche. Sur la couche optique, il s'agit de prévoir un re-routage total ou partiel des chemins en cas de panne. Le cas du re-routage total assure la transmission de chaque flux de données par un chemin de backup pour toute panne possible. Or chaque chemin de backup correspond à une certaine capacité de reserve, inutilisée en absence de panne. Cela induit alors un surdimensionnement important du réseau au niveau optique. Néanmoins, dans ce cas, toutes les capacités au niveau MPLS sont garanties quelle que soit la panne et la protection sur cette couche peut se limiter aux pannes d'un canal d'émission ou de réception (c'est-à-dire d'un lien MPLS).

À l'inverse, nous pouvons imaginer ne pas prendre en compte la sécurisation au niveau optique mais au niveau MPLS. Dans ce cas, nous ne surdimensionnons plus le réseau au niveau optique. Cependant, il nous faut assurer la sécurisation sur la couche MPLS. C'est-à-dire que nous devons prévoir des LSP de backup pour chaque LSP principal et pour chaque type de panne. Or, étant donné la relation entre les deux couches, la panne d'un arc sur la couche optique provoque la panne de chaque lightpath passant par cet arc et donc de chaque arc correspondant dans le graphe MPLS. Nous ne pouvons donc pas faire l'hypothèse sur la couche MPLS qu'au plus un arc peut tomber en panne à un instant donné. Ainsi, dans le cas où aucune sécurisation n'est prévue au niveau optique, une panne provoque une importante chute de capacité au niveau MPLS. Il devient alors peu probable de trouver un routage respectant les contraintes de demandes et/ou de qualité de service.

Il nous est alors apparu nécessaire d'établir une relation entre les deux couches afin d'éviter un surdimensionnement trop coûteux tout en assurant le routage des demandes et la qualité de service. Pour cela, nous considérons au niveau optique un re-routage partiel des lightpaths en cas de panne. Une panne se traduit alors au niveau MPLS par une diminution partielle de la capacité de certains arcs, ce qui offre

plus de chances d'assurer un routage satisfaisant quelle que soit la panne. De plus, une analyse à moyen ou long terme de l'utilisation du réseau permet d'améliorer le dimensionnement et la protection au niveau optique.

D'autre part, nous devons prendre en compte le délai dans les contraintes de qualité de service au niveau de la couche MPLS. Un lien e est habituellement modélisé par une file d'attente $M/M/1$ (hypothèse de Kleinrock). Dans ce cas, le nombre moyen N_e de bits dans le système et le délai moyen par bit T_e dépendent de la bande passante C_e du lien et du débit entrant x_e :

$$N_e = \frac{x_e}{C_e - x_e} \qquad T_e = \frac{1}{C_e - x_e} \tag{1.1}$$

Ainsi, le délai moyen θ_e d'un paquet s'exprime à l'aide de la taille moyenne d'un paquet λ et du délai moyen par bit T_e :

$$\theta_e = \lambda T_e = \frac{\lambda}{C_e - x_e} \tag{1.2}$$

Dans le domaine des réseaux de télécommunications, la gestion de la qualité de service traite essentiellement des grandeurs globales du réseau portant sur l'ensemble des arcs du réseau telles que la congestion maximale ou le délai moyen. En revanche, avec le protocole MPLS, la qualité de service est liée au choix de chemins de routage (LSP). Ainsi, la contrainte de délai porte sur le chemin de bout-en-bout et le délai associé est estimé par sommation du délai moyen sur chaque arc du chemin.

Une dernière question reste ouverte : faut-il optimiser globalement le routage et la protection pour chaque type de panne ou faut-il optimiser le routage principal en priorité puis la protection en fonction de ce routage ? En effet, il semble logique de donner la priorité au routage principal car il sera normalement beaucoup plus utilisé que les autres. Cependant, si nous décidons de ne pas remettre ce routage en cause pour la protection, il se peut que nous n'arrivions pas à trouver de chemins de protection satisfaisant les contraintes de qualité de service pour chaque chemin principal et pour chaque ensemble de pannes.

Nous avons proposé un modèle général regroupant un ensemble de problématiques énoncées précédemment. Il tient compte de l'aspect économique à travers la fonction objectif qui limite à la fois le nombre de LSP principaux et le nombre de reconfigurations dues aux pannes. Il gère une contrainte de qualité de service complexe relative aux délais de bout-en-bout. Enfin, en utilisant un principe de scénarios proposant des distributions de capacités différentes, il apporte une collaboration avec la couche optique pour une sécurisation partagée. Ces problématiques imposent une complexité importante à ce modèle. La limitation des LSP et la contrainte de délai de bout-en-bout seront étudiées dans les chapitres suivants.

1.4 Notions de base de l'optimisation dans les réseaux

Cette section rassemble un certain nombre de notions de base. Elles permettront au lecteur non familiarisé dans ces domaines de mieux appréhender les chapitres suivants de ce mémoire. La première

partie est consacrée aux notions fondamentales de la théorie des graphes. La terminologie spécifique de ce domaine et les propriétés importantes y sont évoquées. La deuxième partie décrit les bases de la programmation linéaire. Elles sont nécessaires à la compréhension des modèles mathématiques développés dans les chapitres suivants. Par souci de cohérence, les algorithmes de résolution utilisés pour ces modèles seront détaillés dans les chapitres correspondants. Enfin, nous appliquerons ces bases sur des problèmes classiques d'optimisation dans les réseaux.

1.4.1 Notions de théorie des graphes

Cette partie n'a pas vocation à fournir une liste exhaustive de toutes les notions de théorie des graphes. Nous nous limiterons aux notions de bases qui seront utilisées dans la suite de ce mémoire. Le lecteur souhaitant compléter ses connaissances dans ce domaine pourra se reporter aux ouvrages [GM95, AMO93].

De nombreux ouvrages citent le problème des sept ponts de Königsberg, résolu par Leonhard Euler en 1736, comme un des premiers résultats de théorie des graphes. La théorie moderne des graphes s'est développée bien plus tard, dans les années 1960, sous l'impulsion, entre autres, des travaux de Berge, Erdös, Edmonds, Ford et Fulkerson, ... Cependant, ce problème illustre parfaitement les relations profondes qu'il existe entre la théorie des graphes et la topologie. En effet, le problème posé à Euler lie les berges et des îlots de la rivière Pregolya par des ponts de la ville de Königsberg. Sans aucune mesure, le problème est fondamentalement topologique puisqu'il s'agit de trouver un chemin parcourant tous les ponts une et une seule fois.

Un **graphe** est donc une modélisation mathématique de relations entre des objets. Dans l'exemple précédent, les objets sont les berges et les îlots, et les ponts forment les liens entre ces objets. Mathématiquement, un graphe $G = (N, A)$ est défini par :

- un ensemble $N = \{v_1, \cdots, v_n\}$ de nœuds (ou sommets). On notera $|N| = n$ le nombre de **nœuds** du graphe. Par abus de langage, on désignera un nœud par son indice ($v_i = i$).
- un ensemble $A = \{a_1, \cdots, a_m\}$ d'**arcs**. Chaque arc $a \in A$ est un couple (i, j) de nœuds. On notera $|A| = m$ le nombre d'arcs du graphe.

Si on ne fait pas de distinction entre l'arc (i, j) et l'arc (j, i), le graphe est dit non orienté. L'usage veut alors qu'on nomme un arc «arête». Par la suite, nous utiliserons principalement des graphes orientés mais la plupart des définitions restent valides dans le cas non orienté.

La représentation graphique usuelle d'un graphe se fait par des cercles pour les nœuds et des flèches pour les arcs (des traits sans flèche pour les arêtes). Ainsi la figure 1.5 représente le graphe $G = (N, A)$ défini par :

- $N = \{1, 2, 3, 4, 5, 6\}$
- $A = \{(1, 2), (1, 3), (2, 3), (2, 4), (2, 5), (3, 5), (4, 6), (5, 4), (5, 6)\}$

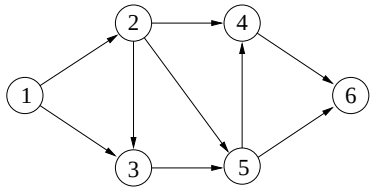


FIG. 1.5 – Représentation graphique d'un graphe.

Deux arcs ayant au moins une extrémité en commun sont dits **adjacents**. On appelle **cocycle** d'un ensemble $S \subset N$ de nœuds l'ensemble des arcs incidents aux nœuds de S, et on le note $\omega(S)$. On distincte souvent l'ensemble $\omega^-(S)$ des arcs entrants du cocycle et l'ensemble $\omega^+(S)$ des arcs sortants du cocycle.

Un **chemin** entre deux nœuds désigne une succession d'arcs les reliant. Les arcs doivent être pris dans le «bon sens», c'est-à-dire de l'origine vers la destination. Par exemple, sur la figure 1.5, les arcs $(1,2)$, $(2,4)$ et $(4,6)$ forment un chemin de 1 à 6. Si au moins un arc est pris à contre-sens, la succession d'arcs décrit une **chaîne**. Par exemple, la chaîne $\{(1,3),(2,3),(2,4),(4,6)\}$ relie les nœuds 1 et 6. Un chemin est dit élémentaire s'il ne passe pas deux fois par le sommet. Un graphe est dit **connexe** lorsqu'il existe au moins une chaîne entre tout couple de nœuds. Un cycle est un chemin dont l'origine et la destination d'un chemin sont confondues.

Soit s et t deux nœuds du graphe. Une (s,t)-**coupe** est un ensemble C d'arcs déconnectant s de t. En d'autres termes, tous les chemins de s à t passent par au moins un arc de C. Une coupe définit alors une partition de l'ensemble des nœuds $N = S \cup \bar{S}$ où $s \in S$, $t \in \bar{S}$ et pour tout arc $(i,j) \in C$, $i \in S$ et $j \in \bar{S}$.

Un **réseau** est un graphe orienté $G = (N, A)$ avec une valuation positive de ses arcs. La valuation $c_{i,j}$ d'un arc (i,j) est appelée la **capacité** de l'arc. Un **flot** F sur un réseau est également une valuation positive des arcs, c'est-à-dire un application de A dans $\mathbb{R}_+$, vérifiant la loi de conservation du flot : la quantité de flot entrant est égale à la quantité de flot sortant (1.3). Le flot « entre » dans le réseau en un sommet s appelé origine du flot et en « sort » en un sommet t appelé destination. On associe généralement une demande d à chaque paire origine-destination. Cette demande peut être variable comme, par exemple, dans un problème cherchant la plus grande demande acceptable par le réseau pour une paire origine-destination donnée. Un flot est dit réalisable lorsqu'il respecte les capacités des arcs.

$$\sum_{a \in \omega^-(v)} F_a - \sum_{a \in \omega^+(v)} F_a = \begin{cases} d & \text{si } v = t, \\ -d & \text{si } v = s, \\ 0 & \text{sinon.} \end{cases} \qquad \forall v \in N \tag{1.3}$$

Un **multiflot** est une généralisation de la notion de flot à plusieurs paires origine-destination. Soit $\mathcal{K}$ un ensemble de K paires origine-destination (s_1,t_1), …, (s_K,t_K) et d_1, …, d_K les demandes respectives. À chaque paire origine-destination (s_k,t_k) correspond un vecteur flot F^k. Un multiflot pour les K paires origine-destination est donc un ensemble de flots $(F^1,\ldots,F^K)$. Un multiflot réalisable respecte alors les contraintes de conservation de flot (1.4) et de capacités (1.5).

$$\sum_{a \in \omega^-(v)} F_a^k - \sum_{a \in \omega^+(v)} F_a^k = \begin{cases} d_k & \text{si } v = t_k, \\ -d_k & \text{si } v = s_k, \\ 0 & \text{sinon.} \end{cases} \qquad \forall v \in N \quad \forall k \in \mathcal{K} \tag{1.4}$$

$$\sum_{k \in \mathcal{K}} F_a^k \leqslant c_a \qquad \forall a \in A \tag{1.5}$$

1.4.2 Notions de programmation linéaire

La programmation linéaire est une méthode de modélisation mathématique développée durant la deuxième guerre mondiale pour diminuer les coûts liés à la logistique. Elle fut gardée secrète jusqu'en 1947, année de publication de l'article de George B. Dantzig sur l'algorithme du simplexe. Cette même année, John Von Neumann développait la théorie de la dualité. Pour être complet sur les origines de ce domaine, citons les travaux du mathématicien russe Leonid Kantorovich qui a utilisé des méthodes similaires avant Dantzig dans le domaine de l'économie. Cette association entre des méthodes pratiques performantes et des résultats théoriques puissants assura le succès de cette méthodologie auprès des entreprises qui, après guerre, l'utilisèrent pour leur planification quotidienne. La programmation linéaire est aujourd'hui utilisée dans de nombreux domaines et la théorie s'est étoffée de nombreux résultats. Nous nous limiterons dans cette partie aux notions de base utilisées dans la présentation de nos modèles dans les chapitres suivants. Pour une étude plus approfondie de ce domaine, le lecteur pourra consulter [BT97].

Comme pour la théorie des graphes, la programmation linéaire modélise des relations entre des objets. Ces objets représentent généralement des quantités, fractionnaires ou entières, mais ils peuvent être également des variables de décision (prenant la valeur 0 ou 1). À ces objets sont associés des **variables** et les relations entre ces variables sont modélisés par des **contraintes**. Une contrainte est une relation entre une combinaison linéaire et une constante. Ainsi, pour que la contrainte soit respectée, les variables doivent prendre des valeurs de façon à ce que la combinaison linéaire soit inférieure, supérieure ou égale à la constante, suivant la nature de la contrainte. En regroupant les variables x et les constantes b dans des vecteurs, et les coefficients des combinaisons linéaires dans une matrice A, on peut noter l'ensemble des contraintes sous la forme matricielle $Ax \leqslant b$.

Un programme linéaire est une modélisation d'un problème d'optimisation. L'ensemble des contraintes décrit un espace convexe ou polytope, appelé espace réalisable. Le problème consiste à trouver un vecteur de cet espace optimisant une fonction linéaire. Cette **fonction objectif** peut s'écrire comme le produit scalaire entre le vecteur des variables x et un vecteur ligne constant c, appelé vecteur coût. Un problème de programmation linéaire est mis sous **forme canonique** s'il s'écrit comme :

$$(\mathrm{P}) \begin{cases} \max\ z = cx \\ s.c. \\ \qquad Ax \leqslant b \\ \qquad x \geqslant 0 \end{cases} \tag{1.6}$$

La résolution numérique d'un tel programme linéaire par l'algorithme nécessite le passage à la forme standard, qui peut s'obtenir de n'importe quel programme linéaire moyennant éventuellement quelques transformations linéaires. Un problème de programmation linéaire est mis sous **forme standard** s'il s'écrit comme :

$$(\mathrm{P}_2) \begin{cases} \max\ z = cx \\ s.c. \\ \qquad Ax = b \\ \qquad x \geqslant 0 \end{cases} \tag{1.7}$$

Par rapport à la forme canonique, il suffit d'ajouter à chaque contrainte une nouvelle variable appelée **variable d'écart**. Cette variable, positive, représente l'écart entre la combinaison linéaire des variables et la constante de la contrainte. Soit n (resp. m) le nombre de variables (resp. contraintes) du programme linéaire dans sa forme canonique. Dans sa forme standard, le programme linéaire comporte donc $n + m$ variables et toujours m contraintes. Les variables d'écart jouent un rôle important dans l'initialisation du simplexe car elles forment la base initiale.

Une **base** B d'un programme linéaire mis sous forme standard est un ensemble de m variables (parmi les $n + m$) qui peuvent s'exprimer de manière unique (et affine) en fonction des n autres variables, dont l'ensemble sera noté N. Le vecteur des variables x peut alors être divisé en deux vecteurs : les variables faisant partie de la base, dites **variables en base**, forment le vecteur x_B et les variables restantes, dites **variables hors base**, forment le vecteur x_N. Par abus de langage, nous noterons également B (resp. N) la matrice composée des colonnes de A correspondantes aux variables en base (resp. hors base). Ainsi les contraintes du programme linéaire s'écrivent :

$$Bx_B + Nx_N = b \tag{1.8}$$
$$x_B \geqslant 0 \tag{1.9}$$
$$x_N \geqslant 0 \tag{1.10}$$

Ainsi, si les variables hors base sont fixées, les variables en base sont déterminées de manière unique si et seulement si il y a indépendance linéaire entre elles, c'est-à-dire si la matrice B correspondante est inversible. On parle alors de base valide. Enfin, il est important de vérifier que les valeurs des variables en base ainsi calculées sont admissibles. La seule contrainte sur ces valeurs est leur positivité. Il faut donc au moins que ces valeurs soient positives pour des variables hors base nulles, c'est-à-dire $B^{-1}b \geqslant 0$. Si cette condition est vérifiée, la base est dite réalisable. On peut montrer qu'à toute base réalisable correspond un sommet du polyèdre des contraintes. Le principe de l'algorithme du simplexe consiste à passer d'une base réalisable à une autre, et donc d'un sommet du polyèdre à un autre, tout en améliorant toujours la valeur de la solution associée.

Pour cela, considérons la base B courante au début d'une itération du simplexe. Les variables hors base ont une valeur nulle par définition. Le passage à une autre base réalisable se fait en permutant une variable en base avec une variable hors base. On choisit bien entendu de faire entrer en base une variable hors base qui va permettre d'améliorer la valeur de la fonction objectif. Divisons le vecteur coût c comme le vecteur des variables en deux vecteurs c_B et c_N. L'expression de la fonction objectif z et l'équation (1.9) permettent d'exprimer z sous la forme suivante :

$$z = c_B B^{-1} b + (c_N - c_B B^{-1} N) x_N \tag{1.11}$$

Notons $\bar{c} = c_N - c_B B^{-1} N$ le vecteur des **coûts réduits**. Une variable hors base x_i qui permet d'améliorer la valeur de la fonction objectif dispose d'un coût réduit $\bar{c}_i$ strictement positif (en maximisation). En effet, si on augmente x_i, sans toucher aux autres variables hors base, alors la valeur de la solution va augmenter. Mais dans le même temps, cette augmentation va affecter les valeurs des variables en base,

au point qu'une d'entre elle, x_j, va s'annuler (si aucune ne diminue alors le problème est non borné). Dans ce cas là, on obtient une nouvelle base réalisable qui est définie en remplaçant x_j par x_i. Si tous les coûts réduits sont négatifs ou nuls, on ne peut augmenter la valeur de la solution en touchant à une variable hors base. La base est alors dite optimale.

Une forme révisée du simplexe a été développée durant la seconde moitié des années 50 dans le but de réduire la quantité de calculs réalisés à chaque itération et surtout d'économiser l'occupation mémoire du programme. Elle repose sur le constat que seule la connaissance des variables en base est importante et seule l'obtention de B^{-1} nécessite un effort calculatoire, le reste des informations nécessaires au simplexe en découlant naturellement. Ce constat est très important pour les programmes linéaires utilisant un grand nombre de variables. Il est parfois impossible de gérer toutes les variables en mémoire, surtout que peu d'entres elles seront finalement utilisées dans les bases réalisables successives. Une solution consiste à ne stocker qu'un sous-ensemble des variables et de générer les variables hors base entrant en base.

Les contraintes d'un programme linéaire, comme la fonction objectif, sont des combinaisons linéaires de variables. De plus, une combinaison conique (coefficients non négatifs) d'inéquations de même type donne une inéquation valide. Une combinaison conique de contraintes peut donc fournir une borne de la fonction objectif avant même de se lancer dans la résolution par le simplexe. Il suffit que de cette combinaison conique de contraintes résulte une combinaison linéaire de variables majorant la fonction objectif. On peut alors se poser la question de savoir quelle est la meilleure borne possible. Notons y le vecteur ligne de taille m des coefficients de la combinaison conique de contraintes qui s'écrit alors :

$$yAx \leqslant yb \tag{1.12}$$

Cette combinaison linéaire majore la fonction objectif si et seulement si le coefficient de chaque variable x_i est supérieur au coût c_i correspondant (car les variables sont positives). D'autre part, on cherche à minimiser la borne. Le problème peut donc se formuler sous la forme du programme linéaire (D).

$$\text{(D)} \begin{cases} \min\ w = yb \\ s.c. \\ \qquad yA \ \geqslant\ c \\ \qquad y \ \geqslant\ 0 \end{cases} \tag{1.13}$$

Traditionnellement (P) est appelé programme **primal** et les variables x_i sont les **variables primales**. (D) est appelé programme **dual** de (P) et ses variables y_i sont appelées **variables duales**. On remarque qu'à chaque contrainte primale correspond une variable duale et qu'à chaque contrainte duale correspond une variable primale. On pourrait aussi remarquer que si l'on s'attache à déterminer une borne pour (D), on retombera sur (P). Les relations très fortes entre le primal et le dual s'expriment en partie dans les résultats suivants.

Théorème 1.1. *(Dualité faible) Soient $\bar{x}$ et $\bar{y}$ un couple de solutions réalisables pour (P) et son dual (D), de valeur respective $\bar{z}$ et $\bar{w}$. On a alors la relation*

$$\bar{z} \leqslant \bar{w} \tag{1.14}$$

Démonstration. Comme $A\bar{x} \leqslant b$ dans (P), on a $\bar{w} = \bar{y}b \geqslant \bar{y}A\bar{x}$.

Comme $\bar{y}A \geqslant c$ dans (P), on a $\bar{w} \geqslant \bar{y}A\bar{x} \geqslant c\bar{x} = \bar{z}$ □

Corollaire 1.1. *Soient $\bar{x}$ et $\bar{y}$ un couple de solutions réalisables pour (P) et son dual (D), de valeur respective $\bar{z}$ et $\bar{w}$. Si $\bar{z} = \bar{w}$ alors $\bar{x}$ et $\bar{y}$ sont optimales.*

Démonstration. Supposons que le primal (P) admette une solution optimale x^* de valeur z^* strictement supérieure à $\bar{z}$. Alors z^* est également strictement supérieure à $\bar{w}$, ce qui contredit le théorème 1.1 de dualité faible. Donc $\bar{x}$ est solution optimale de (P). Idem pour $\bar{y}$. □

Théorème 1.2. *(Dualité forte) Si (P) possède une solution optimale x^* alors sont dual (D) possède aussi une solution optimale y^* et leur valeur associées, z^* et w^* coïncident.*

Démonstration. Soit $x^* = (x^*_B,\ x^*_N)$ solution optimale de (P). Dans le dual, la partition liée à la base $(B\ N)$ correspond au système de contraintes suivant :

$$\begin{aligned} yB &\geqslant c_B \\ yN &\geqslant c_N \end{aligned}$$

Si l'on considère la solution duale y^* obtenue en posant $y^* = c_B B^{-1}$, alors cette solution est réalisable pour le dual :

- $yB = c_B B^{-1} B = c_B \geqslant c_B$ pour les premières contraintes,
- $c_N - yN = c_N - c_B B^{-1} N = \bar{c} \leqslant 0$ car la solution primale est optimale donc les coûts réduits sont négatifs. Donc $yN \geqslant c_N$ pour les secondes contraintes.

La valeur de la fonction objectif du programme dual est $w^* = y^* b = c_B B^{-1} b = c_B x_B = z^*$. Par le corollaire précédent, il s'ensuit que y^* est optimale pour (D). □

Notons que le choix de la solution complémentaire $y^* = c_B B^{-1}$ correspond effectivement à la valeur optimale de la solution duale. À partir de ce constat, la méthode de **génération de colonnes** permet de résoudre des programmes linéaires disposant d'un grand nombre de variables (pour plus de détails cf. [Las70]). Elle consiste dans un premier temps à résoudre le programme linéaire sur un nombre réduit de variables. Ce sous-ensemble $\mathcal{X}$ de variables doit être suffisant pour définir une base réalisable afin d'initier l'algorithme. Si une telle base existe, on peut exécuter l'algorithme du simplex et obtenir une solution optimale pour ce programme linéaire sur un sous-ensemble de l'espace des solutions réalisables. Si la solution est optimale pour le sous-ensemble de variables, cela signifie que toutes les variables hors base incluses dans $\mathcal{X}$ ont un coût réduit négatif. Cependant, il existe peut-être des variables n'appartenant pas à $\mathcal{X}$ dont le coût réduit est strictement positif. Une telle variable peut rentrer en base en améliorer la valeur de la fonction objectif.

La recherche d'une variable « améliorante » est appelée **sous-problème** de la génération de colonnes (*pricing problem*). Le nombre de variables hors base étant très important, on cherche uniquement la variable ayant le plus grand coût réduit. Pour cela, il est nécessaire de disposer de la valeur des variables

duales correspondant à la base optimale sur le sous-ensemble $\mathcal{X}$ (solution complémentaire $y^* = c_B B^{-1}$). Le sous-problème dépend alors essentiellement de la structure du modèle. Il est généralement beaucoup plus simple que le problème initial. Il arrive même souvent que ce soit un problème classique du domaine pour lequel il existe de nombreux algorithmes performants, assurant ainsi une génération rapide des colonnes.

La variable (ou colonne) ainsi générée par le sous-problème est ajoutée au sous-ensemble de variables $\mathcal{X}$. Une nouvelle base optimale peut alors être calculée, améliorant la valeur de la fonction objectif. De nouvelles variables sont générées successivement. L'algorithme s'arrête lorsque le sous-problème ne trouve plus de variable à coût réduit strictement positif. Cela assure que la solution optimale pour le sous-ensemble de variables $\mathcal{X}$ est optimale pour le programme linéaire tout entier.

1.4.3 Exemples de problèmes classiques

Toujours dans un but d'introduction, cette section présente des problèmes classiques d'optimisation dans les réseaux. Nous retrouverons ces problèmes dans les chapitres suivants dans des formes plus évoluées, plus complexes, avec des contraintes supplémentaires.

1.4.3.1 Plus court chemin

Soit $G = (N, A)$ un graphe muni d'une valuation l_a des arcs nommée longueur. On appelle longueur l_p d'un chemin p la somme des longueurs des arcs de p.

$$l_p = \sum_{a \in p} l_a \tag{1.15}$$

Le problème du plus court chemin consiste à trouver un chemin p entre deux nœuds s (la source) et t (la destination) tel que sa longueur soit minimale. Cette notion de longueur sur les arcs et les chemins induit la notion de distance entre les nœuds. En effet, pour une source s donnée, la distance est une valuation des nœuds définie pour chaque nœud v comme la longueur du plus court chemin entre s et v.

Notons que le nombre de chemins peut croître de façon exponentielle en fonction du nombre d'arcs du graphe. Ce problème peut donc paraître compliqué dans sa généralité. Cependant, dans la plupart des cas, il existe des algorithmes polynomiaux pour le résoudre. C'est notamment le cas lorsque les longueurs sont positives. L'algorithme de Dijkstra [Dij59] peut alors s'appliquer.

Le principe consiste à partitionner l'ensemble des nœuds par une coupe $S/\bar{S}$. Les nœuds de S sont dits *fixés* car leur distance par rapport à la source est calculée et ne sera plus remis en cause. À l'opposé, les nœuds de $\bar{S}$ sont dits *temporaires*. Chaque nœud v dispose d'une valeur $d(v)$ appelée *label* égale à la plus courte distance trouvée à un instant de l'algorithme. À chaque itération, le nœud i de $\bar{S}$ ayant le plus petit label passe dans S. Son label $d(i)$ vaut alors la distance minimale de la source à ce nœud. Ainsi, pour tout nœud j du cocycle sortant de i, le plus court chemin de la source à j passant par i vaut $d(i) + l_{i,j}$. Le label de j est alors mis à jour en cas d'amélioration. Ainsi, les distances $d(j)$ respectent

les conditions d'optimalité sur les arcs (théorème 1.3).

Théorème 1.3 (Conditions d'optimalité pour les plus courts chemins)**.** *Pour tout nœud $j \in N$, soit $d(j)$ la longueur d'un chemin de la source au nœud j. Alors les valeurs $d(j)$ représentent les distances par les plus courts chemins si et seulement si elles vérifient les conditions d'optimalité suivantes :*

$$d(j) \leqslant d(i) + l_{i,j} \qquad \forall (i,j) \in A \tag{1.16}$$

Dans [MPRD99], les auteurs étudient une généralisation de ce problème, appelée problème du chemin optimal. La complexité du problème dépend évidemment du critère à optimiser le long du chemin. Cependant, ils énoncent le principe d'optimalité faible qui, s'il est vérifié par un problème de chemin optimal, assure qu'on peut utiliser des algorithmes à label. Ces algorithmes, dont l'algorithme de Dijkstra, construisent l'arbre des plus courts chemins en déterminant les sous-chemins optimaux.

Principe 1.1. *(Optimalité faible) Il existe un chemin optimal constitué de sous-chemins optimaux.*

Par exemple, le problème du chemin de capacité maximale vérifie ce principe (la capacité d'un chemin est la plus petite capacité des arcs du chemin). Une forme légèrement modifiée de l'algorithme de Dijkstra permet donc de le résoudre. Dans la suite de ce mémoire, nous étudierons un problème de chemin optimal dont le critère à optimiser est tel que le problème ne satisfait pas au principe d'optimalité faible.

1.4.3.2 Flot maximum

Soit $G = (N, A)$ un réseau connexe. Les capacités $c_{i,j}$ sur les arcs limitent le flot en terme de bande passante. Le problème du flot maximum consiste à trouver la plus grande quantité de flot pouvant circuler sur G entre une source s et une destination t. Cela représente la quantité maximale d'information pouvant circuler entre deux nœuds du réseau avec les limitations physiques imposées par les liens.

L'algorithme de Ford & Fulkerson fait partie des premiers à avoir été spécialement conçu pour la résolution de ce problème. Bien que de nombreuses variantes ont amélioré ses performances et que d'autres types d'algorithmes lui sont maintenant supérieurs, il est une base théorique importante de la théorie des flots.

Son principe réside en la recherche d'une **chaîne augmentante**, chaîne qui permet d'accroître la quantité de flot circulant entre la source et la destination. Un chaîne peut prendre des arcs à « contre-sens » et ajouter du flot sur un arc à contre-sens revient à lui en retirer. Pour formaliser cette idée, construisons à partir d'un graphe G et d'un flot f sur ce graphe, le **graphe résiduel** G_f :

- A un arc (i, j) du graphe G est associé dans le graphe résiduel l'arc *forward* de capacité $c_{i,j} - f_{i,j}$. La capacité de l'arc *forward* traduit que l'on peut encore augmenter le flot sur l'arc (i, j) d'au plus $c_{i,j} - f_{i,j}$, ce qui correspond à la saturation de l'arc.
- A un arc (i, j) du graphe G est associé dans le graphe résiduel l'arc *backward* de capacité $f_{i,j}$. La capacité de l'arc *backward* traduit que l'on peut diminuer la valeur du flot sur l'arc (i, j) d'au plus $f_{i,j}$, ce qui correspond à annuler le flot.

L'algorithme de Ford & Fulkerson consiste à trouver un chemin de capacité strictement positive dans le graphe résiduel, puis d'augmenter le flot de cette capacité sur les arcs correspondants aux arcs *forwards* empruntés, et de le diminuer sur les arcs correspondants aux arcs *backwards*. Les notions présentées dans cet algorithme permettent de démontrer le théorème **flot maximal / coupe minimal** qui présente un résultat de dualité important entre les flots et les coupes. Dans le chapitre 2, nous étudierons l'influence de nouvelles contraintes sur ce problème de flot en analysant les répercutions sur cette relation de dualité. La capacité d'une coupe $S\backslash\bar{S}$, où $s \in S$ et $t \in \bar{S}$, est définie comme la somme des capacités des arcs de A reliant un nœud de S à un nœud de $\bar{S}$. La coupe minimale est alors définie comme la (s,t)-coupe de capacité minimale.

Théorème 1.4 (Flot maximum, coupe minimale)**.** *La valeur maximale du flot pouvant circuler sur un réseau est égale à la capacité de la coupe minimale.*

Démonstration. cf. [AMO93]. □

Pour illustrer ce résultat, nous présentons le problème du flot maximum sous la forme d'un programme linéaire (FM). Pour cela, nous ajoutons au graphe un arc entre la destination t et la source s ($A' = A \cup \{(t,s)\}$). Celui-ci assurera la symétrie dans les contraintes de conservation de flot. Associons ensuite à chaque arc (i,j) une variable $x_{i,j}$ indiquant la quantité de flot qui y circule.

$$\text{(FM)} \begin{cases} \max\ x_{t,s} & & & \\ s.c. & & & \\ \quad \displaystyle\sum_{(i,v)\in\omega^-(v)} x_{i,v} - \sum_{(v,j)\in\omega^+(v)} x_{v,j} = 0 & \forall v \in N & (a) \\ \quad x_{i,j} \leqslant c_{i,j} & \forall (i,j) \in A & (b) \\ \quad x_{i,j} \geqslant 0 & \forall (i,j) \in A' & (c) \end{cases} \tag{1.17}$$

Les contraintes (1.17.a) assurent le respect de la conservation du flot sur le réseau. Les contraintes (1.17.) limitent la quantité de flot circulant sur un arc à sa capacité.

La théorie de la dualité nous permet de déterminer le programme dual (CM). Pour cela, associons aux contraintes (1.17.a) les variables duales λ_v pour tout nœud v de N et aux contraintes (1.17.b) les variables duales $\pi_{i,j}$ pour tout arc (i,j) de A.

$$\text{(CM)} \begin{cases} \min \displaystyle\sum_{(i,j)\in A} c_{i,j}\pi_{i,j} & & & \\ s.c. & & & \\ \quad \pi_{i,j} \geqslant \lambda_j - \lambda_i & \forall (i,j) \in A & (a) \\ \quad \lambda_t - \lambda_s \geqslant 1 & & (b) \\ \quad \pi_{i,j} \geqslant 0 & \forall (i,j) \in A & (c) \\ \quad \lambda_v \text{ quelconque} & \forall v \in N & (d) \end{cases} \tag{1.18}$$

Les variables λ_v représentent ainsi des potentiels sur les nœuds. L'objectif de minimisation et la forme des contraintes font que les variables $\pi_{i,j}$ sont « plaquées » sur la valeur $\lambda_j - \lambda_i$. Elles représentent ainsi une différence de potentiels, c'est-à-dire une tension. L'additivité des tensions impose que, pour chaque chemin entre s et t, la somme des différences de potentiels successives est égale à la différence de potentiel entre s et t, c'est-à-dire 1 du fait de la contrainte (1.18.b) et de l'objectif à minimiser. La solution optimale est donc une distribution des tensions permettant à chaque chemin de vérifier cette propriété. De plus, l'objectif est de minimiser la somme des capacités pondérées par cette distribution des tensions. La solution optimale est donc une coupe (pour toucher tous les chemins) de capacité minimale. Cette coupe partitionne l'ensemble des nœuds en un ensemble S dont les potentiels valent celui de la source, et un ensemble $\bar{S}$ dont les potentiels valent celui de la destination.

Enfin, un autre résultat de la dualité vient confirmer cette relation forte entre flots et coupes. Le théorème des écarts complémentaires montre qu'à l'optimum, soit une contrainte primale est saturée (il y a égalité entre le membre de gauche et celui de droite), soit la variable duale associée est nulle (et inversement puisque le primal est le dual du dual). Ainsi, on observe qu'à l'optimum, les arcs de la coupe de capacité minimale correspondent aux variables duales non nulles. Les contraintes de capacité associées dans le primal sont donc saturées. Le flot sur ces arcs est donc égale à leur capacité.

1.4.3.3 Problèmes de multiflot

La formulation la plus intuitive d'un problème de multiflot est la formulation arc-sommet. Elle consiste à associer à chaque arc a de A et à chaque paire origine-destination k de $\mathcal{K}$ une variable f_a^k positive représentant la quantité de flot circulant sur a pour la paire origine-destination (s_k, t_k). Illustrons cette modélisation sur le problème de multiflot compatible qui consiste à chercher un multiflot routant les demandes $d_1, \ldots, d_K$ associées aux paires origine-destination en minimisant le dépassement des capacités c_a, $a \in A$. Si ce dépassement, noté ε, est nul, le multiflot sera dit compatible. Si ce n'est pas le cas, cela signifie qu'il faudra investir dans de nouvelles capacités sur les arcs. Le problème se modélise alors par le programme linaire (MFC) :

$$
\text{(MFC)} \begin{cases} \min\ \varepsilon \\ s.c. \\ \displaystyle\sum_{a \in \omega^-(v)} f_a^k - \sum_{a \in \omega^+(v)} f_a^k = \begin{cases} d^k & \text{si } v = t_k, \\ -d^k & \text{si } v = s_k, \\ 0 & \text{sinon.} \end{cases} & \forall v \in N\ \forall k \in \mathcal{K} \quad (a) \\ \displaystyle\sum_{k \in \mathcal{K}} f_a^k \leqslant c_a + \varepsilon & \forall a \in A \quad (b) \\ f_a^k \geqslant 0 & \forall a \in A\ \forall k \in \mathcal{K} \quad (c) \\ \varepsilon \geqslant 0 & (d) \end{cases} \tag{1.19}
$$

Lorsque l'écoulement du flot n'est pas contraint (variables f_a^k réelles), la résolution du problème peut se faire par n'importe quelle méthode de programmation linéaire. En revanche, si les contraintes sur le flot aboutissent à une modélisation par variables discrètes (cas, par exemple, du mono-routage), le problème est connu pour être un problème difficile.

D'autre part, cette modélisation devient difficilement exploitable en pratique lorsque le nombre de paires origine-destination est grand. Pour contourner ce problème de taille, d'autres formulations ont été proposées. En particulier, la formulation arc-chemin est régulièrement utilisée. Elle considère les chemins empruntés pour acheminer la demande. Ainsi, pour chaque paire origine-destination k, l'ensemble des chemins élémentaires reliant s_k à t_k est noté $\mathcal{P}_k$. Puis, à chaque chemin p de $\mathcal{P}_k$ est associée une variable f_p^k représentant la quantité de flot empruntant le chemin p pour la paire origine-destination k. De plus, nous noterons δ_a^p le vecteur indicateur qui vaut 1 si le chemin p passe par l'arc a et 0 sinon. Le problème de multiflot compatible se modélise alors par le programme linéaire (MFC_2) :

$$(\text{MFC}_2) \begin{cases} \min \ \varepsilon & & & \\ s.c. & & & \\ \quad \displaystyle\sum_{p \in \mathcal{P}_k} f_p^k = d_k & \forall k \in \mathcal{K} & (a) \\ \quad \displaystyle\sum_{k \in \mathcal{K}} \sum_{p \in \mathcal{P}_k} \delta_a^p f_p^k \leqslant c_a + \varepsilon & \forall a \in A & (b) \\ \quad f_p^k \geqslant 0 & \forall k \in \mathcal{K} \ \forall p \in \mathcal{P}_k & (c) \\ \quad \varepsilon \geqslant 0 & & (d) \end{cases} \tag{1.20}$$

Il est à noter que l'on peut facilement calculer les valeurs du multiflot arc-sommet à partir du multiflot arc-chemin par la formule 1.21.

$$f_a^k = \sum_{p \in \mathcal{P}_k} \delta_a^p f_p^k \qquad \forall a \in A \quad \forall k \in \mathcal{K} \tag{1.21}$$

Paradoxalement, ce modèle comporte un nombre de variable encore plus grand que le modèle arc-sommet. Cependant, il est plus facilement exploitable en pratique. En effet, le principe de génération de colonnes permet de se contenter d'un nombre restreint de variables et de « générer » des variables supplémentaires en cas de besoin pour atteindre l'optimum.

1.5 Conclusion

Nous avons présenté dans ce chapitre les problématiques liées à la conception de réseaux haut-débit et plus particulièrement au protocole MPLS. Ce protocole apporte des améliorations significatives au niveau des architectures de réseaux et de la gestion de la qualité de service. Nous avons également exposé un état de l'art des problèmes d'optimisation dans les réseaux de télécommunication et des méthodes de résolution. Enfin nous avons discuté d'un modèle de routage multicouche dans un réseau MPLS sur WDM développé avec nos partenaires du projet PRESTO. Ce modèle est le point de départ des travaux de thèse détaillés dans les chapitres suivants.

En particulier, nous nous sommes intéressé à la contrainte de limitation du nombre de LSP et à son influence sur des problèmes de routage classiques comme le problème du flot maximal. Cette contrainte correspond à la notion de flot k-séparable (cf. [BKS05]) que nous détaillons dans le chapitre suivant.

Nous y étudions également la relation entre ce type de flot et les coupes du graphe. Enfin nous présentons plusieurs modèles pour le problème du flot k-séparable maximal.

CHAPITRE 2

MODÈLES POUR LES FLOTS k-SÉPARABLES

Le support d'un flot est l'ensemble des moyens mis en œuvre pour transporter le flot d'informations ou de biens sur le réseau. Il peut prendre des formes diverses, unitaires comme un paquet IP portant une unité d'information, ou collectif comme un camion véhiculant plusieurs unités de marchandises, ou un LSP diffusant plusieurs paquets. Ainsi, les contraintes sur le support de flot peuvent revêtir divers aspects, compliquant plus ou moins les problèmes de flot traités.

Les caractéristiques recherchées d'un support de flot sont souvent liées à la topologie empruntée par le flot et induisent parfois des problèmes topologiques anciens. Ainsi certains problèmes de sécurisation dans les réseaux considèrent des supports de flot en forme de chemins arcs-disjoints (chemins n'ayant aucun arc en commun). Le théorème de Menger, énoncé en 1927, montre justement que le nombre de chemins arcs-disjoints entre deux nœuds d'un graphe non orienté est égal à la cardinalité de la plus petite coupe.

Plus récemment, Kleinberg [Kle96] a introduit la notion de multiflots entiers (*unsplittable flow*) comme généralisation des chemins disjoints. Il s'agit de flots qui ne se séparent pas et qui traversent donc le réseau le long d'un unique chemin. De nombreux problèmes munis de cette contrainte ont été étudiés dans la littérature. Des résultats théoriques ont été formulés (notamment sur la complexité) et des approches algorithmiques développées (algorithmes d'approximation ou algorithmes exacts).

Enfin, dans [BKS05], les auteurs généralisent cette notion de flots entiers pour router les flots sur des supports constitués d'un nombre maximal k de chemins. Le cas $k = 1$ correspond aux flots entiers. Dans le cas général, ils nomment les flots respectant cette limite flots k-séparables (*k-splittable flow*). Cette notion permet de modéliser la limitation du nombre de LSP dans le cadre d'un routage MPLS. Cette contrainte peut s'appliquer à tout type de problème de flot. Par souci de clarté, nous nous attacherons ici à présenter notre étude sur le problème du flot maximal avec une unique paire origine-destination. Nous garderons néanmoins une approche suffisamment générale pour qu'elle puisse s'étendre à d'autres problèmes (autres fonctions objectifs, plusieurs paires origine-destination, ...).

Dans une première partie, nous présenterons les définitions précises relatives à ce problème. Nous détaillerons également certaines propriétés importantes sur cette classe de flot. Dans les sections suivantes, nous présenterons les avantages et les inconvénients des modèles mathématiques que nous avons développés.

2.1 Définitions et propriétés

La notion de flots k-séparables a été introduite dans [BKS05] comme une généralisation des flots entiers (*Unsplittable Flow*). Nous souhaitons donner dans cette section un cadre théorique précis à cette notion. En s'appuyant sur le problème du flot k-séparable maximum (*k-splittable Maximum Flow Problem*, KMFP), nous présenterons quelques propriétés de cette classe de flots, notamment une extension des relations de dualité entre flots et coupes. Enfin, nous introduirons les notations communes utilisées dans les différents modèles présentés dans la suite de ce chapitre.

2.1.1 Définitions

Comme présenté dans la section 1.4, un flot sur un graphe $G = (N, A)$ est défini comme une application de l'ensemble des arcs E vers $\mathbb{R}$ vérifiant la loi de conservation de flot (dite aussi loi de Kirchoff). Ainsi l'ensemble des flots, noté $\mathcal{F}^A$, forme un sous-ensemble de $\mathbb{R}^A$, l'ensemble des applications de A vers $\mathbb{R}$.

Cette formulation des flots modélise l'agrégation des flux sur chaque arc et ne prend pas en compte le parcours des paquets dans le réseau. Ce parcours représente le support du flot, c'est-à-dire l'« objet » qui permet au flot de traverser le réseau. Par exemple, pour une réseau MPLS, ce sont les LSP qui servent de support aux paquets. Ainsi, la gestion des contraintes sur le support de flot ne peut pas se faire en utilisant uniquement cette formulation agrégée des flots. Il est nécessaire d'étendre cette représentation des flots à des structures plus complexes qu'un arc seul. Dans notre cas, les composantes indivisibles du flot que sont les paquets parcourent le réseau sur des chemins. Cependant, dans d'autres problèmes, le support de flot peut prendre d'autres formes (cycles, arbres, ...). Un flot peut donc être considéré comme l'agrégation de plusieurs flots circulant sur un même réseau.

L'opération inverse, nommée décomposition, consiste à déterminer un ensemble de flots, sur des supports particuliers, constituant, par leur agrégation, un flot donné sur les arcs. Le théorème de décomposition de flot est un résultat classique de théorie des graphes (cf. [AMO93]).

Théorème 2.1 (Théorème de décomposition de flot)**.** *Un flot dans un graphe peut être décomposé en au plus $(m + n)$ chemins élémentaires et cycles élémentaires (où m et n sont les nombres d'arcs et de sommets). Un chemin (resp. cycle) élémentaire est un chemin (resp. cycle) ne passant pas plus d'une fois par le même sommet.*

Ce théorème de décomposition de flot montre que la décomposition d'un flot en chemins et cycles élémentaires est une opération facile. Cependant, la décomposition n'est pas unique et parmi toutes celles possibles, il est naturel de chercher celle qui nécessite le moins de chemins. Dans les problèmes de télécommunications, il est rare qu'une circulation (un flot sur un cycle) ait une signification pour le problème réel. Par exemple, dans un problème de routage, il est inconcevable qu'un paquet tourne en rond au milieu du réseau. Les flots solutions se décomposent donc uniquement en chemins. Le problème de minimisation du support de flot consiste à chercher une décomposition du flot composée d'un nombre minimal de chemins. Nous définissons alors comme suit la largeur d'un flot.

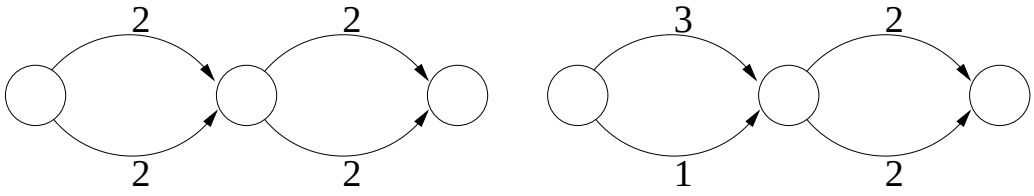
Définition 2.1 (largeur)**.** *Soit F un flot sur un graphe G. La largeur $w(F)$ du flot F (*flow width*) est le nombre minimal de chemins tels que l'agrégation des flots de chaque chemin donne exactement F.*

Le problème général de minimisation du support d'un flot, c'est-à-dire le calcul de sa largeur, est un problème NP-difficile (voir [Vat04]). Cependant, la largeur est traitée différemment dans le problème du flot k-séparable maximum. L'objectif n'est pas de calculer un flot d'une largeur minimale, mais plutôt de maintenir la largeur en dessous d'un seuil donné.

Définition 2.2 (flot k-séparable [BKS05])**.** *Un flot k-séparable est défini comme étant un flot pouvant être routé par un ensemble d'au plus k chemins. Sa largeur est donc majorée par la donnée k.*

Nous nous intéressons donc dans ce problème aux flots k-séparables routant une quantité maximale de données entre une source et une destination. Il est évident qu'une circulation ne peut en rien améliorer une solution. Nous ne considérerons donc que le sous-ensemble de $\mathcal{F}^A$ constitué des flots décomposables en chemins élémentaires uniquement (sans cycle). Pour une paire origine-destination donnée, nous noterons $\mathcal{P}$ l'ensemble des chemins élémentaires de la source à la destination. Ainsi, ces flots peuvent être décrits par une de leurs décompositions, donc par une application de $\mathcal{P}$ dans $\mathbb{R}$.

Les flots k-séparables forment alors le sous-ensemble $\mathcal{F}_k^{\mathcal{P}}$ des applications de $\mathcal{P}$ dans $\mathbb{R}$ dont au plus k composantes strictement positives. Ainsi, ils forment un sous-ensemble $\mathcal{F}_k^A$ de $\mathcal{F}^A$, image de $\mathcal{F}_k^{\mathcal{P}}$ par l'agrégation. Par cette opération, les flots sur les chemins se mélangent sur leurs arcs en commun. Il est alors impossible, dans le cas général, de déterminer si un flot est k-séparable à partir de la seule connaissance des arcs utilisés. C'est la distribution du flot sur les arcs qui détermine sa largeur. Par exemple, sur la figure 2.1(a), le flot est clairement 2-séparable. Avec les mêmes arcs mais une distribution différente (figure 2.1(b)), la nature du flot est totalement différente.



(a) Flot 2-séparable (b) Flot 3-séparable

FIG. 2.1 – Flots k-séparables.

2.1.2 Propriétés

Nous rappelons ici les résultats de complexité et d'approximation du problème présentés par Baier, Köhler et Skutella dans [BKS05]. Le théorème 2.2 prouve que le problème du flot k-séparable maximum est NP-difficile.

Théorème 2.2. *La résolution d'instances du problème du flot 2-séparable maximum avec un rapport d'approximation strictement meilleur que $\frac{2}{3}$ est NP-difficile.*

Démonstration. La preuve de ce théorème proposée par Baier, Köhler et Skutella s'appuie sur une réduction au problème NP-difficile SAT. À partir d'une instance du problème SAT composée de n variables $x_1, \ldots, x_n$ et m clauses $C_1, \ldots, C_m$, on construit un graphe de taille linéaire en n et m avec une source s et une destination t (voir figure 2.2).

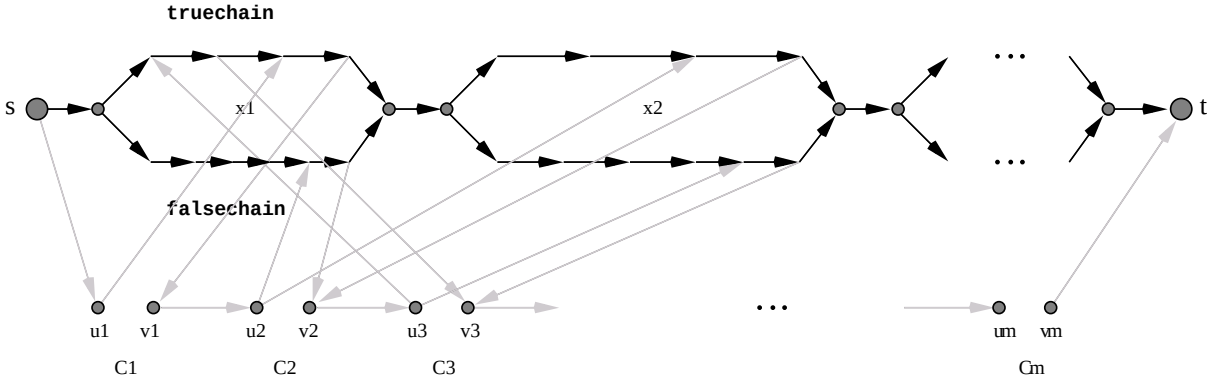


FIG. 2.2 – Réduction du problème du flot k-séparable maximum au problème SAT.

Ce graphe est construit en deux parties. La première, en noir sur la figure 2.2, représente les littéraux. À chaque variable x_i correspond deux chemins parallèles. Un des chemins est appelé « chemin vrai » (*true-chain*) et l'autre « chemin faux » (*false-chain*) de x_i. Le « chemin vrai » (resp. « chemin faux ») est composé de deux arcs pour chaque clause où x_i apparaît niée (resp. non-niée). Chaque paire de chemins forme un sous-graphe correspondant à une variable. Puis ces sous-graphes sont reliés en série par des « ponts » entre eux et avec la source et la destination. Chaque arc de cette première partie du graphe a une capacité de 2. Ainsi, un chemin portant un flot de valeur 2 et empruntant le « chemin vrai » (resp. « chemin faux ») de x_i sera interprété comme une affectation de la valeur vrai (resp. faux) à la variable x_i.

La deuxième partie représente les clauses. L'idée est de construire un second chemin en utilisant les arcs définis précédemment, et qui sera bloqué si la valuation des variables donnée par le premier chemin n'est pas satisfaisante. Pour chaque clause C_j, on considère deux sommets u_j et v_j reliés aux arcs représentant les littéraux de la clause. Tous les arcs ajoutés, en gris sur la figure 2.2, ont une capacité de 1. Notons que les clauses doivent utiliser les arcs des « chemins vrai/faux » dans l'ordre inverse de leur progression dans ce deuxième chemin. Par exemple, sur la figure 2.2, la clause C_3 utilise un arc du « chemin vrai » de la variable x_1 précédant celui utilisé par la clause C_1. Dans le cas inverse, une solution pourrait router du flot de la clause C_1 à la clause C_3 en « sautant » la clause C_2.

Ainsi pour router un flot maximum sur 2 chemins, il faut pouvoir router 2 unités sur un premier chemin sur le « sous-graphe des variables » (en noir sur la figure 2.2) puis une unité de flot sur un second chemin utilisant les arcs du « sous-graphe des clauses » (en gris sur la figure 2.2) . Pour cela, il doit exister pour chaque clause C_j, un arc correspondant à un littéral permettant de relier u_j à v_j, c'est-à-dire que l'arc ne porte pas de flot du premier chemin et donc l'affectation de la valeur à la variable par le premier chemin satisfait le littéral et donc la clause.

Ainsi, un flot 2-séparable maximum route 3 unités de flot si la formule est satisfiable et 2 unités sinon. Ceci démontre que le problème du flot 2-séparable maximum est NP-difficile, comme le problème SAT, et que tout algorithme polynomial a un rapport d'approximation d'au plus $\frac{2}{3}$. □

L'algorithme d'approximation proposé par Baier, Köhler et Skutella dans [BKS05] utilise une spécification des flots k-séparables aux propriétés remarquables définie comme suit.

Définition 2.3. *Un flot uniforme exactement k-séparable est un flot routé sur exactement k chemins et où chaque chemin porte la même quantité de flot.*

Ces auteurs proposent un algorithme polynomial pour résoudre le problème du flot uniforme exactement k-séparable maximum et montrent qu'une solution optimale de ce problème est une solution du problème du flot k-séparable maximum dont la valeur vaut au moins la moitié de l'optimum. Cet algorithme se base sur le théorème 2.3 qui étend le théorème du flot maximum / coupe minimale. Ce théorème nécessite la définition de la capacité k-uniforme d'une coupe.

Définition 2.4. *La capacité k-uniforme $c_k(C)$ d'une coupe C est la plus grande quantité de flot pouvant être répartie sur les arcs de C en exactement k blocs uniformes. Si la capacité d'un arc le permet, celui-ci peut accueillir plusieurs de ces blocs.*

Théorème 2.3. *La valeur du flot uniforme exactement k-séparable maximum est égal à la capacité k-uniforme minimale sur les coupes.*

On retrouve avec ce résultat une relation forte entre flots et coupes. Celle-ci fera l'objet d'une étude détaillée et étendue aux flots k-séparables dans la section suivante.

2.1.3 Relation avec les coupes

Dans cette partie, nous présentons une réflexion sur le rapport entre les flots simples (une seule paire origine-destination) et les coupes. Nous nous attachons notamment à soulever une analogie entre les flots fractionnaires classiques, les flots monoroutés, qui sont routés sur un seul chemin, et les flots k-séparables, qui sont routés sur au plus k chemins. Nous considérons un réseau $G = (N, A)$ où chaque arc $a \in A$ est muni d'une capacité $u_a > 0$. Dans la suite, le terme de coupe désignera sans ambiguïté une (s, t)-coupe où s est la source et t la destination d'une unique paire origine-destination.

Définition 2.5 (Répartition)**.** *Une **répartition** d'un flot de valeur f sur un sous-ensemble C de A est une séparation du flot en un ensemble $(f_1, f_2, \cdots, f_h)$ de valeurs positives réparties sur les arcs de C.*

Une répartition est donc caractérisée par une séparation $(f_1, f_2, \cdots, f_h)$ du flot et une fonction indicatrice δ de $\{1, \cdots, h\} \times C$ dans $\{0, 1\}$. δ_i^a vaut 1 si la valeur f_i est associée à l'arc a. L'ensemble des f_i et la fonction δ doivent vérifier les relations suivantes.

$$\sum_{i=1}^{h} f_i = f \tag{2.1}$$

$$\sum_{a \in C} \delta_i^a = 1 \qquad \forall i = 1, \cdots, h \tag{2.2}$$

Nous distinguerons les quatre types de répartition suivants qui diffèrent sur la forme de la séparation :

1. *Une **répartition fractionnaire** sépare le flot de valeur f sans contrainte particulière.*
2. *Une **répartition monoroutée** sépare le flot de valeur f en une unique valeur (égale à f).*

3. *Une **répartition uniforme exactement** k**-séparable** sépare le flot de valeur f en exactement k valeurs identiques (égales à $\frac{f}{k}$).*
4. *Une **répartition** k**-séparable** sépare le flot de valeur f en au plus k valeurs (qui peuvent être différentes).*

Définition 2.6 (Répartition valide)**.** *Une répartition d'un flot f sur un sous-ensemble d'arcs C est dite **valide** si elle respecte les capacités des arcs de C.*

Ainsi, la séparation $(f_1, f_2, \cdots, f_h)$ et la fonction indicatrice δ caractérisant la répartition doivent vérifier la relation suivante.

$$\sum_{i=1}^{h} \delta_i^a f_i \leqslant u_a \qquad \forall a \in C \tag{2.3}$$

Nous souhaitons maintenant montrer que l'existence d'un flot, quelle que soit sa nature, est lié non seulement à sa répartition (de même nature que le flot) sur les coupes mais également à un lien de « compatibilité » entre les coupes. En effet, la répartition d'un flot sur une coupe correspond aux chemins qui portent ce flot. Ces chemins définissent une décomposition du flot sur l'ensemble du réseau, donc sur chaque coupe. Ainsi, les répartitions du flot sur les coupes doivent avoir la même séparation.

Cependant, il est intéressant de remarquer que ce lien de compatibilité entre des répartitions fractionnaires est facile à obtenir puisqu'il n'y a pas de limite dans la séparation. Deux répartitions d'un même flot définissent aisément, à l'aide d'éventuels fractionnements supplémentaires, une séparation commune sans modifier la distribution du flot sur les arcs des coupes correspondantes. De même, les séparations des répartitions entières et des répartitions uniformes exactement k-séparables sont fixées et donc identiques sur chaque coupe. Il n'y a donc, là non plus, aucun problème de compatibilité entre les répartitions.

Dans le cas des répartitions k-séparables, le problème est bien plus complexe car les séparations disposent à la fois d'une limite sur leur cardinalité et d'une liberté sur chaque valeur. Ainsi, deux séparations d'une même quantité de flot peuvent ne pas accepter de séparation commune respectant la limite sur le nombre de valeurs. Par exemple, les séparations $(2, 2)$ et $(3, 1)$ sont compatibles dans le cas fractionnaire car elles peuvent être décrites par la même séparation $(2, 1, 1)$. Mais elles ne sont pas compatibles dans le cas 2-séparable.

On constate alors, pour les trois premiers types de répartitions, que la compatibilité entre les répartitions ne dépend que de la quantité de flot totale. Cette grandeur, réelle, définit un ordre total entre les répartitions. Pour les répartitions k-séparables, la compatibilité dépend également du k-uplet définissant la séparation (avec éventuellement des composantes nulles). Ces k-uplets définissent un ordre partiel entre les répartitions. La nature des problèmes de flots k-séparables revêt alors une complexité plus importante.

Nous présentons maintenant dans un cadre plus formel les idées que nous venons d'évoquer en les appliquant aux différents types de flots.

Propriété 2.1. *Un flot de valeur f peut être routé de s à t si et seulement si, pour toute s,t-coupe, il existe une répartition fractionnaire valide de f.*

Démonstration. Démontrons chaque partie de l'équivalence.

1. Supposons qu'il existe un flot F routant f de s à t. Pour toute coupe C, la répartition caractérisée par :
   - la séparation indexée par les arcs de C : $f_a = \frac{F_a}{\sum_{a \in C} F_a} f \ \forall a \in C$,
   - la fonction indicatrice δ_i^a valant 1 si $i = a$ et 0 sinon,

   est valide car, pour tout a de C, $\sum_{i \in C} \delta_i^a f_i = f_a = \frac{f}{\sum_{a \in C} F_a} F_a \leqslant F_a \leqslant u_a$.
2. Supposons qu'il n'existe pas de flot routant f de s à t en respectant les capacités. Considérons alors un flot F routant f de s à t sans respecter les capacités et minimisant la congestion maximale, i.e. $\max_{a \in A} F_a/u_a$. Cette congestion maximale est donc strictement supérieure à 1.

   Considérons l'ensemble C des arcs dont la congestion est supérieure ou égale à 1. S'il existait un chemin de s à t ne passant par aucun arc de C, alors on pourrait dérouter du flot par ce chemin et faire diminuer la congestion maximale, ce qui contredirait la définition de F. Nous en déduisons donc que C est une coupe.

   De plus, comme la congestion maximale est strictement supérieure à 1, il vient :

   $$\sum_{a \in C} u_a < \sum_{a \in C} F_a = f$$

   Donc il existe bien une coupe sur laquelle il n'est pas possible d'établir une répartition fractionnaire valide de f. □

Notons $\mathcal{C}_{s,t}$ l'ensemble des (s,t)-coupes. Le corollaire suivant s'ensuit :

Corollaire 2.1. *Le flot maximal f^* pouvant être routé entre la source et la destination est égal à la capacité de la coupe minimale :*

$$f^* = \min_{C \in \mathcal{C}_{s,t}} \sum_{a \in C} u_a \tag{2.4}$$

Ce corollaire est un résultat classique de théorie des graphes mais cette présentation permet de faire l'analogie avec les autres types de flot.

Propriété 2.2. *Un flot monorouté de valeur f peut être routé de s à t si et seulement si, pour toute coupe, il existe une répartition monoroutée valide de f.*

Démonstration. Démontrons chaque partie de l'équivalence.

1. Supposons qu'il existe un chemin P routant f de s à t. Pour toute coupe C, il existe au moins un arc $a_p \in C \cap P$. La répartition monoroutée (séparation $f_1 = f$), caractérisée par la fonction indicatrice δ_1^a valant 1 si $a = a_p$ et 0 sinon, est valide.
2. Supposons qu'il n'existe pas de chemin routant f de s à t en respectant les capacités. Alors l'ensemble des arcs de capacité strictement inférieure à f est une coupe (ensemble non minimal).

   Donc il existe bien une coupe sur laquelle il n'est pas possible d'établir une répartition monorouté valide de f. □

Le corollaire suivant s'ensuit :

Corollaire 2.2. *Le flot monorouté maximal f^* pouvant être routé entre la source et la destination est égal au plus petit maximum des capacités des arcs d'une coupe.*

$$f^* = \min_{C \in \mathcal{C}_{s,t}} \max_{a \in C} u_a \tag{2.5}$$

Propriété 2.3. *Un flot uniforme exactement k-séparable de valeur f peut être routé de s à t si et seulement si, pour toute coupe, il existe une répartition uniforme exactement k-séparable valide de f.*

Démonstration. Démontrons chaque partie de l'équivalence.

1. Supposons qu'il existe k chemins p_1, ..., p_k routant chacun $\frac{f}{k}$ de s à t. Pour toute coupe C, il existe au moins un arc $a_k \in C \cap p_k$ pour tout chemin p_k. La répartition uniforme exactement k-séparable (séparation $f_1 = \cdots = f_k = \frac{f}{k}$), caractérisée par la fonction indicatrice δ_i^a valant 1 si $a = a_i$ et 0 sinon, est valide.
2. Supposons qu'il n'existe pas d'ensemble de k chemins routant chacun $\frac{f}{k}$ de s à t en respectant les capacités. Il existe donc un nombre de chemins limite $k' \in \{0, \ldots, k-1\}$ tel que :
   - il existe un ensemble de k' chemins routant chacun $\frac{f}{k}$,
   - il n'existe pas un ensemble de $k'+1$ chemins routant chacun $\frac{f}{k}$.

   Considérons un ensemble de k' chemins $\{p_1, \ldots, p_{k'}\}$ routant chacun $\frac{f}{k}$ et permettant de router un maximum de flot sur un $(k'+1)^{\text{ième}}$ chemins. Considérons alors l'ensemble C des arcs de capacité résiduelle ($u_a - \sum_{i=1}^{k'} \delta_a^{p_i} \frac{f}{k}$) strictement inférieure à $\frac{f}{k}$. Comme il n'existe pas $k'+1$ chemins routant chacun $\frac{f}{k}$, C est une coupe (ensemble non minimal). S'il existait une répartition uniforme exactement $(k'+1)$-séparable sur C, cela contredirait la définition des k' chemins $\{p_1, \ldots, p_{k'}\}$. Il n'existe donc pas de répartition uniforme exactement k-séparable de f sur C. □

Le corollaire suivant s'ensuit :

Corollaire 2.3. *Le flot uniforme exactement k-séparable maximal f^* pouvant être routé entre la source et la destination est égal à la plus petite capacité uniforme exactement k-séparable d'une coupe.*

Ce résultat a été démontré dans [BKS05] sous une forme différente. Cette présentation permet de faire l'analogie avec les autres types de flot.

Propriété 2.4. *Un flot k-séparable de valeur f peut être routé de s à t si et seulement si il existe un ensemble de répartitions k-séparables sur les coupes ayant la même séparation du flot.*

Démonstration. Démontrons chaque partie de l'équivalence.

1. Supposons qu'il existe k chemins $p_1, \ldots, p_K$ routant f de s à t. Notons $f_1, \ldots, f_k$ les quantités de flot routées par chaque chemin. Pour toute coupe C, il existe un arc $a_i \in C \cap p_i$ pour tout chemin p_i. La répartition k-séparable caractérisée par :
   - la séparation $(f_1, \ldots, f_k)$,
   - la fonction indicatrice δ_i^a valant 1 si $a = a_i$ et 0 sinon,

est valide.

2. Supposons qu'il n'existe pas de flot k-séparable routant f de s à t en respectant les capacités. Alors, quelque soit la séparation $(f_1, \ldots, f_k)$ de f, il existe un nombre de chemins limite $k' \in \{0, \ldots, k-1\}$ tel que :
   - il existe un ensemble de k' chemins routant respectivement f_1, ..., $f_{k'}$,
   - il n'existe pas un ensemble de $k'+1$ chemins routant respectivement f_1, ..., $f_{k'+1}$.

   Considérons un ensemble de k' chemins $\{p_1, \ldots, p_{k'}\}$ routant respectivement f_1, ..., $f_{k'}$ et permettant de router un maximum de flot sur un $(k'+1)^{\text{ième}}$ chemins. Considérons alors l'ensemble C des arcs de capacité résiduelle ($u_a - \sum_{i=1}^{k'} \delta_a^{p_i} f_i$) strictement inférieure à $f_{k'+1}$. Comme il n'existe pas $k'+1$ chemins routant respectivement f_1, ..., $f_{k'+1}$, C est une coupe (ensemble non minimal). S'il existait une répartition $(k'+1)$-séparable sur C, cela contredirait la définition des k' chemins $\{p_1, \ldots, p_{k'}\}$. Il n'existe donc pas de répartition k-séparable de f sur C.

   Ainsi pour toute séparation $(f_1, \ldots, f_k)$ de f, il existe au moins une coupe sur laquelle aucune répartition k-séparable de cette séparation n'est possible. □

2.1.4 Notations communes aux différents modèles

Dans les sections suivantes, nous présentons des modèles sous forme de programmes linéaires du problème du flot k-séparable maximal (KMFP). Par souci de clarté, nous proposons d'introduire ici les notations communes à chaque modèle.

Soit $G = (N, A)$ un graphe orienté constitué d'un ensemble N de n nœuds et d'un ensemble A de m arcs. Chaque arc $a \in A$ est muni d'une capacité $u_a \geqslant 0$. Nous considérons une unique paire origine-destination (s, t) du flot à router sur G. Pour faciliter la généralisation des modèles à plusieurs paires origine-destination, généralement notées k, nous notons H le nombre maximal de chemins élémentaires utilisés pour porter le trafic. Le problème du flot k-séparable maximal (KMFP) est de trouver un flot maximal porté par au plus H chemins élémentaires.

Notons $\mathcal{P}$ l'ensemble des chemins élémentaires entre s et t. Pour chaque chemin p, $u_p = \min_{a \in p} u_a$ désigne sa capacité. On note δ_a^p la fonction indicatrice du passage du chemin p par l'arc a.

Nous utilisons principalement deux types de variables :

- Les **variables de flots** représentent la quantité de flot circulant sur un arc, variables x_a, ou sur un chemin, variables x_p.
- Les **variables de support de flot** sont les variables de décision relatives au passage du flot sur un arc, variables y_a, ou sur un chemin, variables y_p.

2.2 Modèle arc-chemin basique

Le premier modèle est basé sur une formulation arc-chemin. La variable de flot $x_p \geqslant 0$ représente la quantité de flot portée par le chemin $p \in \mathcal{P}$ et y_p est la variable de décision associée. Le modèle arc-chemin s'écrit alors comme suit :

$$
(\text{KMFP}_1)\begin{cases}
\max \sum_{p\in\mathcal{P}} x_p & & \\
s.c. & & \\
\quad \sum_{p\in\mathcal{P}} \delta_a^p x_p \leqslant u_a & \forall a \in A & (a) \\
\quad x_p - u_p y_p \leqslant 0 & \forall p \in \mathcal{P} & (b) \\
\quad \sum_{p\in\mathcal{P}} y_p \leqslant H & & (c) \\
\quad x_p \geqslant 0 & \forall p \in \mathcal{P} & (d) \\
\quad y_p \in \{0,1\} & \forall p \in \mathcal{P} & (e)
\end{cases} \tag{2.6}
$$

La contrainte (2.6.a) est la contrainte de capacité. Elle limite le flot agrégé sur chaque arc par la capacité de l'arc. Le lien entre les variables de flot et les variables de décision se fait dans la contrainte (2.6.b). En effet, si un chemin p n'est pas choisi pour router du flot, la variable de flot associée doit être nulle. Dans le cas inverse, le flot routé sur p est tout de même limité par la capacité du chemin, même si cette contrainte est redondante avec les contraintes de capacité.

Ainsi, nous pouvons utiliser les variables de décision pour compter le nombre de chemins utilisés. Il vient naturellement la contrainte de largeur de flot (2.6.c) qui s'assure qu'au plus H chemins portent le flot. Notons enfin les contraintes d'intégrité (2.6.d) pour la positivité du flot, et (2.6.e) pour le caractère binaire des variables de décision.

La difficulté de ce modèle réside d'une part dans la présence de variables de décision y_p qui rendent le problème combinatoire, et d'autre part dans le nombre exponentiel de variables car lié au nombre de chemins. De plus, comme nous verrons dans le chapitre 3, ce modèle ne peut pas être utilisé de manière performante dans une méthode de résolution de type *Branch & Bound*. Des reformulations sont alors nécessaires.

Le modèle suivant sera une reformulation utilisant une approche arc-sommet. Cependant, la plupart des articles récents reprennent le modèle arc-chemin du fait de sa plus grande facilité de mise en œuvre [BKS05, BKP03, EJLW01, MTUP04].

2.3 Modèle arc-sommet

Plutôt que de choisir H chemins dans $\mathcal{P}$, chacun de ces H chemins peut être décrit comme un sous-problème de flot en utilisant une formulation arc-sommet. En effet, un chemin peut se construire à partir d'un flot entier de valeur 1, moyennant des contraintes adéquates. Ainsi, les arcs portant ce flot de la source à la destination constituent le chemin. Cette discrétisation impose un ordre sur les H chemins car chaque bloc de variables, associées à un chemin et au flot qu'il porte, sera repéré par un numéro de chemin h allant de 1 à H.

Pour chaque numéro de chemin $h = 1 \dots H$, la variable $x_a^h \geqslant 0$ correspond à la contribution au flot sur l'arc $a \in A$ par le chemin numéro h. La variable de décision $y_a^h \in \{0,1\}$ indique si le $h^{\text{ième}}$ chemin

passe par l'arc a ou non. Enfin, la variable $z_h \geqslant 0$ représente la quantité de flot porté par le chemin numéro h. Il peut être vu comme la contribution de ce chemin au flot sur un arc virtuel, de capacité infinie, reliant la destination à la source.

Notons $\omega^-(v)$ l'ensemble des arcs entrants du cocycle du sommet v et $\omega^+(v)$ l'ensemble des arcs sortants du cocycle. Alors, le modèle arc-sommet peut s'établir comme suit :

$$
(\text{KMFP}_2)\begin{cases}
\max \displaystyle\sum_{h=1}^{H} z_h & & & \\
s.c. & & & \\
\displaystyle\sum_{a\in\omega^-(v)} x_a^h - \sum_{a\in\omega^+(v)} x_a^h = \begin{cases} z_h & \text{si } v=t, \\ -z_h & \text{si } v=s, \\ 0 & \text{sinon.} \end{cases} & & \forall h=1\dots H,\ \forall v\in N & (a) \\
\displaystyle\sum_{a\in\omega^-(v)} y_a^h - \sum_{a\in\omega^+(v)} y_a^h = \begin{cases} 1 & \text{si } v=t, \\ -1 & \text{si } v=s, \\ 0 & \text{sinon.} \end{cases} & & \forall h=1\dots H,\ \forall v\in N & (b) \\
\displaystyle\sum_{h=1}^{H} x_a^h \leqslant u_a & & \forall a\in A & (c) \\
x_a^h - u_a y_a^h \leqslant 0 & & \forall h=1\dots H,\ \forall a\in A & (d) \\
\displaystyle\sum_{a\in\omega^-(v)} y_a^h \leqslant 1 & & \forall h=1\dots H,\ \forall v\in N & (e) \\
x_a^h \geqslant 0 & & \forall h=1\dots H,\ \forall a\in A & (f) \\
y_a^h \in \{0,1\} & & \forall h=1\dots H,\ \forall a\in A & (g) \\
z_h \geqslant 0 & & \forall h=1\dots H & (h)
\end{cases}
\tag{2.7}
$$

Ce modèle utilise deux blocs de contraintes de conservation de flot, pour les variables de flot d'une part (2.7.a), et pour les variables de support de flot d'autre part (2.7.b). Le respect des capacités est assuré par la contrainte (2.7.c). Elle limite le flot agrégé sur chaque arc à hauteur de la capacité correspondante. Le couplage entre le flot et son support est réalisé dans la contrainte (2.7.d). Celle-ci contraint le flot à circuler uniquement sur les arcs définis par le support. Si un arc est utilisé dans le support de flot, le flot est limité à la capacité de l'arc même si cette contrainte est redondante avec la contrainte de capacité (2.7.c). Les contraintes d'intégrité fixent les domaines de définition des variables de flot (2.7.f) et (2.7.h), et de support de flot (2.7.g).

La contrainte (2.7.e) est assez inhabituelle mais elle s'est rapidement avérée indispensable. En effet, dans ce modèle, le support de flot est défini comme un flot entier de valeur 1. Néanmoins, comme tout flot, il se décompose en chemins et circulations. Comme il est entier et de valeur 1, il se décompose en un seul chemin mais cette modélisation accepte des circulations, ce qui peut être préjudiciable.

Prenons l'exemple du modèle arc-sommet, similaire au nôtre, présenté dans [BHV00] pour un problème de flot entier où la demande est donnée et indivisible. Dans ce cas, les variables de flot et de support de flot sont étroitement liées puisque que le flot est égal à la demande sur le support et nul ailleurs. Ainsi, tout cycle dans le support de flot induit une circulation dans le flot. Dans le cas de ce problème, cette circulation dégrade la solution. Elle est donc naturellement éliminée. Dans d'autres problèmes, elle est simplement inutile car elle permet rarement d'améliorer la valeur de la solution.

Notre cas est différent car il y a un espace de liberté dans le lien entre les variables de flot et de support de flot. Le flot n'est pas obligé de suivre le support sur toute sa longueur. Il peut emprunter une partie d'un support puis bifurquer sur une autre. Ainsi, un cycle dans le support n'induit pas forcément une circulation dans le flot. Il peut servir au flot à bifurquer comme illustré sur la figure 2.3. Les arcs a représentés sur cette figure correspondent à des variables de décision $y_a^h = 1$ et, pour chacun, la valeur du flot est indiquée. Dans une telle situation, un cycle dans le support de flot (variables y^h) peut permettre de définir plus d'un chemin pour le flot avec les variables x^h. Cette bifurcation permet donc au flot de définir une meilleure solution, évidemment non valide.

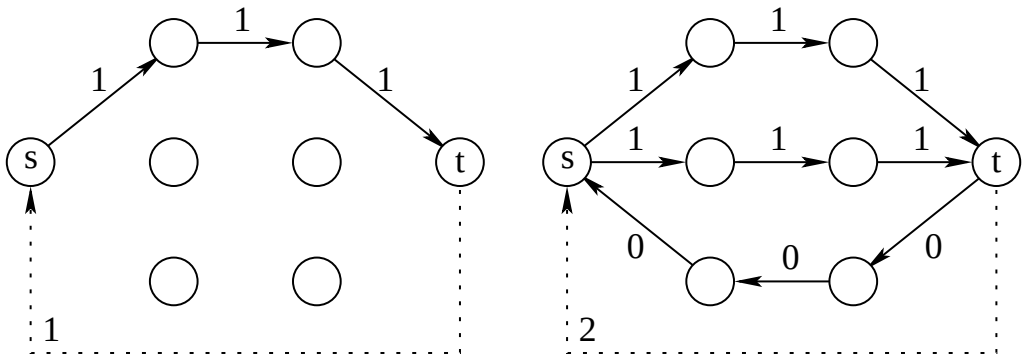


FIG. 2.3 – Influence d'un cycle dans la definition du support de flot y^p sur les variables de flot x^p.

La contrainte (2.7.e) permet de prévenir cette situation en forçant chaque chemin h à être élémentaire, c'est-à-dire d'éviter les cycles connectés au chemin. D'un autre coté, elle n'exclut pas les cycles déconnectés comme illustré sur la figure 2.4. Cependant, puisque de tels cycles ne permettent pas au flot de bifurquer, cette situation n'a pas besoin d'être interdite.

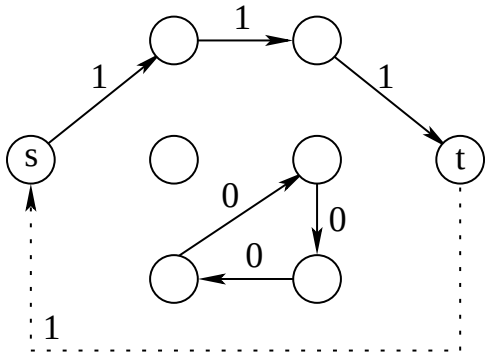


FIG. 2.4 – Influence d'un cycle déconnecté sur les variables de flot x^p.

Par rapport au modèle arc-chemin, ce modèle a l'avantage d'avoir un nombre de variables polynomial en fonction de la taille du graphe et du nombre maximum H de chemins. Cependant, la résolution de ce modèle reste très complexe étant donné le nombre encore important de variables de décision. D'autre part, la contrainte (2.7.e) permet d'assurer la validité du modèle dans cette forme discrète. Cependant, dans le cas fractionnaire obtenu par relaxation des contraintes d'intégrité des variables y_a^h, nous

retrouvons la même constatation. Les variables de support de flot, une fois relâchées, peuvent définir des chemins et des cycles connectés (avec des valeurs fractionnaires) permettant ainsi au flot de bifurquer. Ce point explique la différence de qualité entre ce modèle et le modèle suivant obtenu par décomposition de Dantzig-Wolfe.

Enfin, il est important de remarquer un inconvénient de ce modèle, étroitement lié à la structure de symétrie induite par l'attribution des variables de décision [SS01]. Toute permutation sur les numéros de chemins conserve la validité et la valeur de la solution. L'espace réalisable des solutions est donc inutilement grand puisqu'à chaque solution correspond autant de solutions équivalentes qu'il y a de permutations possibles sur les numéros de chemins. Il peut donc être intéressant d'ajouter une contrainte d'ordre sur les numéros de chemins pour casser la symétrie du modèle et diminuer la taille de l'espace réalisable. Ainsi, la contrainte 2.8 de *variable ordering* ordonne les chemins par flot décroissant : le chemin numéro h doit porter plus de flot que le chemin numéro $h+1$. Cette contrainte est suffisante dans la plupart des cas. Seuls les cas où la solution optimale comporte deux chemins portant la même quantité de flot ne peuvent pas être départagés.

$$z_{h+1} - z_h \leqslant 0 \qquad \forall h = 1 \cdots H-1 \tag{2.8}$$

2.4 Modèle arc-chemin par décomposition de Dantzig-Wolfe

2.4.1 Méthode de décomposition de Dantzig-Wolfe

Le schéma de décomposition de Dantzig-Wolfe a été proposé en 1960 dans [DW60] et constitue maintenant une approche classique pour résoudre des programmes linéaires de grande taille. Il s'avère surtout efficace pour des programmes linéaires fortement structurés comme, par exemple, avec des matrices de contraintes diagonales par bloc (cf. par exemple [Las70, Min83]).

L'idée générale est de considérer le polyèdre associé à une partie des contraintes. L'espace réalisable est donc inclus dans ce polyèdre et une solution réalisable s'exprime en fonction de ses points extrêmes et de ses rayons extrêmes. Cette formulation d'une solution réalisable est reportée dans la partie restante des contraintes et dans la fonction objectif. Le nouveau modèle ainsi obtenu est constitué d'un nombre réduit de contraintes mais d'un nombre potentiellement exponentiel de variables. Il est donc nécessaire d'utiliser la méthode de génération de colonnes afin de ne traiter qu'une partie des variables, celles susceptibles d'intervenir dans la solution optimale.

Formalisons cette idée en considérant un programme linéaire, noté (P), dans lequel nous distinguons deux ensembles de contraintes.

$$\text{(P)} \begin{cases} \max\ cx \\ s.c. \\ \qquad Ax \leqslant a \quad (a) \\ \qquad Bx \leqslant b \quad (b) \\ \qquad x \geqslant 0 \quad (c) \end{cases} \tag{2.9}$$

Le bloc de contraintes $Bx \leqslant b$ et $x \geqslant 0$ définit un polyèdre qui comporte un nombre fini de points extrêmes, notés x^i avec $i = 1, \cdots, I$, et de rayons extrêmes, notés r^j avec $j = 1, \cdots, J$. D'après le théorème de Minkowski, tout point x de ce polyèdre peut s'écrire comme une combinaison convexe des points extrêmes et une combinaison conique des rayons extrêmes.

$$\begin{aligned} x \quad &= \quad \sum_{i=1}^{I} \lambda_i x^i + \sum_{j=1}^{J} \nu_j r^j \\ &\text{avec} \quad \sum_{i=1}^{I} \lambda_i = 1 \text{ et } \lambda_i \geqslant 0 \quad \forall i = 1, \cdots, I \\ &\text{et} \quad \nu_j \geqslant 0 \quad \forall j = 1, \cdots, J \end{aligned} \tag{2.10}$$

Il est alors possible d'écrire (P) sous la forme d'un nouveau programme linéaire appelé **programme maître** et noté (PM).

$$\text{(PM)} \begin{cases} \max\ c \left(\sum_{i=1}^{I} \lambda_i x^i + \sum_{j=1}^{J} \nu_j r^j \right) & & \\ s.c. & & \\ \quad A \left(\sum_{i=1}^{I} \lambda_i x^i + \sum_{j=1}^{J} \nu_j r^j \right) \leqslant a & & (a) \\ \quad \sum_{i=1}^{I} \lambda_i = 1 & & (b) \\ \quad \lambda_i \geqslant 0 & \forall i = 1, \cdots, I & (c) \\ \quad \nu_j \geqslant 0 & \forall j = 1, \cdots, J & (d) \end{cases} \tag{2.11}$$

Pour procéder à la génération de colonnes, on considère un programme maître comportant uniquement les variables associées à un sous-ensemble de points et rayons extrêmes. Ce programme maître est appelé programme maître restreint, noté (PMR). En le résolvant, on obtient une solution optimale et la valeur correspondante des variables duales, notée π pour les contraintes (2.11.a) et π_0 pour la contrainte (2.11.b). La solution optimale du (PMR) est également solution optimale du programme maître si aucune variable hors base de (PM) ne peut améliorer cette solution. Pour s'en assurer, il suffit de calculer les coûts réduits des variables correspondant à des points ou des rayons extrêmes n'intervenant pas dans le (PMR). Si tous ces coûts réduits sont négatifs ou nuls, la solution est bien optimale pour le programme maître. S'il existe un ou plusieurs coûts réduits strictement positifs, une ou plusieurs des variables correspondantes doivent être ajoutées au (PMR) (cf. figure 2.5).

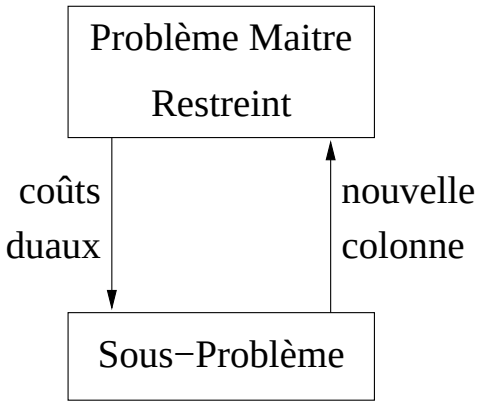


FIG. 2.5 – Principe de la méthode de génération de colonnes.

Ainsi, pour un programme maître restreint donné, il s'agit de déterminer la variable de coût réduit maximal. Cette variable correspond à un point extrême du polyèdre des contraintes (2.9.b). Le problème à résoudre, appelé problème auxiliaire ou sous-problème, se modélise par le programme linéaire (SP).

$$\text{(SP)} \begin{cases} \max\ (c - \pi A)x \\ s.c. \\ \qquad Bx \leqslant b \qquad (a) \\ \qquad x \geqslant 0 \qquad (b) \end{cases} \tag{2.12}$$

La méthode de génération de colonnes basique consiste donc à résoudre itérativement un problème maître restreint puis un problème auxiliaire afin, soit d'enrichir le problème restreint d'une nouvelle variable et de réitérer, soit de déterminer la solution optimale du programme maître (pour plus de développements sur les méthodes de génération de colonnes cf. [Las70, BJN$^+$98]).

2.4.2 Application au flot maximum k-séparable

Dans [AdC03], les auteurs proposent des modèles pour le problème de multiflot entier (*Unsplittable Multicommodity Flow Problem*). En particulier, ils proposent un modèle arc-chemin obtenu par décomposition de Dantzig-Wolfe d'un modèle arc-sommet. Nous pouvons procéder de la même manière sur notre modèle arc-sommet (KMFP$_2$).

Pour cela, nous considérons le polyèdre des flots décrit par la contrainte de conservation de flot (2.7.a). Comme la quantité de flot à router n'est pas limitée, le polyèdre ne possède pas de points extrêmes. Chaque flot peut s'exprimer comme combinaison conique des rayons extrêmes de ce polyèdre. Ces rayons extrêmes correspondent aux chemins élémentaires et aux cycles élémentaires.

Notons $\mathcal{P}$ l'ensemble des chemins élémentaires pour la paire origine-destination donnée et $\mathcal{C}$ l'ensemble des cycles du graphe. Pour chaque chemin p (respectivement chaque cycle c) élémentaire, le

rayon extrême correspondant est représenté par le vecteur $\delta^p \in \{0,1\}^m$ (respectivement δ^c) indiquant les arcs composant le chemin (respectivement le cycle).

Ainsi pour chaque numéro de chemin h, le flot x^h sur les arcs se décompose selon l'équation 2.13.

$$x_a^h = \sum_{p \in \mathcal{P}} x_p^h \delta_a^p + \sum_{c \in \mathcal{C}} x_c^h \delta_a^c \qquad \forall a \in A \quad \forall h = 1, \ldots, H \tag{2.13}$$

Or il est clair que, pour le problème étudié, les cycles ne peuvent pas améliorer la solution. Toute solution peut donc se ramener à un solution équivalente dont les variables de cycles x_c^h sont nulles. Nous pouvons donc les supprimer du modèle.

Nous pouvons procéder de la même manière avec les variables y^h décrivant le support de flot car elles sont également contraintes à la conservation du flot (2.7.b). Notons néanmoins que la quantité de flot à router est fixée et égale à 1. Les chemins élémentaires constituent donc non plus des rayons mais des points extrêmes. Les variables y^h se décomposent donc en une combinaison convexe sur les chemins et une combinaison conique sur les cycles. De plus, la contrainte sur les cocycles (2.7.e) implique que seuls des cycles disjoints du chemin actif peuvent être utilisés. Or ils ne peuvent pas améliorer la solution et les variables correspondantes peuvent donc également être supprimées du modèle. Enfin, le caractère discret des variables de support de flot doit être reporté sur les variables du programme maître.

Ainsi, nous remplaçons dans le modèle (KMFP$_2$) les variables de flot x_a^h et de support de flot y_a^h par leurs décompositions sur les chemins élémentaires (2.14) et (2.15). Les variables de flot x_p^h du programme maître représentent alors la contribution au flot sur le chemin physique p du chemin actif numéro h. Chaque indice représentant un chemin, il est important de distinguer les chemins physiques élémentaires $p \in \mathcal{P}$, constitués d'arcs du graphe, et les chemins actifs qui sont des associations entre un chemin physique et une quantité de flot, numérotés dans ce modèle de 1 à H. Les variables de support de flot y_p^h correspondent donc aux variables de décision associant un chemin physique p et un chemin actif numéroté h. Ainsi, pour toute solution réalisable, chacun des H chemins actifs ne contribue qu'au flot du chemin physique correspondant. Mais il faut imaginer que les contraintes d'intégrités vont être relâchées pour résoudre ce modèle dans sa forme fractionnaire. Et dans ce cas, un chemin actif n'a de chemin que le nom puisqu'il peut alors se répartir sur plusieurs chemins physiques.

$$x_a^h = \sum_{p \in \mathcal{P}} \delta_a^p x_p^h \qquad \forall a \in A \quad \forall h = 1, \ldots, H \tag{2.14}$$

$$y_a^h = \sum_{p \in \mathcal{P}} \delta_a^p y_p^h \qquad \forall a \in A \quad \forall h = 1, \ldots, H \tag{2.15}$$

avec

$$\begin{aligned} x_p^h &\geqslant 0 && \forall p \in \mathcal{P} \quad \forall h = 1, \ldots, H \\ y_p^h &\in \{0,1\} && \forall p \in \mathcal{P} \quad \forall h = 1, \ldots, H \\ \text{et } \sum_{p \in \mathcal{P}} y_p^h &= 1 && \forall h = 1, \ldots, H \end{aligned}$$

Il reste à reporter cette décomposition dans les contraintes restantes du modèle (KMFP$_2$) pour définir le programme maître. Ainsi la contrainte de capacité (2.7.c) devient :

$$\sum_{h=1}^{H}\sum_{p\in P}\delta_a^p x_p^h \leqslant u_a \qquad \forall a \in A \tag{2.16}$$

De même, la contrainte (2.7.d) de liaison entre le flot et le support de flot s'écrit sous la forme 2.17.

$$\sum_{p\in\mathcal{P}}\delta_a^p(x_p^h - u_a y_p^h) \leqslant 0 \qquad \forall a \in A \quad \forall h = 1,\ldots,H \tag{2.17}$$

Or chaque chemin actif numéroté h est associé à au plus un chemin physique p_h. En effet, le caractère discret des variables de support de flot et la combinaison convexe sur les chemins assurent que y_p^h vaut 1 si $p = p_h$ et 0 sinon. Ainsi, la relation 2.17 impliquent le système d'inégalités 2.18 et 2.19.

$$\sum_{p\in\mathcal{P}}\delta_a^p x_p^h \leqslant 0 \qquad \forall a \notin p_h \quad \forall h = 1,\ldots,H \tag{2.18}$$

$$\sum_{p\in\mathcal{P}}\delta_a^p x_p^h \leqslant u_a \qquad \forall a \in p_h \quad \forall h = 1,\ldots,H \tag{2.19}$$

Puis, il découle de 2.18 et 2.19 que tout chemin différent de p_h ne peut pas router de flot et que le flot routé par p_h est limité à $u_p = \min_{a\in p_h} u_a$, la capacité du chemin. Et tout ceci se généralise par la contrainte 2.20.

$$x_p^h - u_p y_p^h \leqslant 0 \qquad \forall p \in \mathcal{P} \quad \forall h = 1,\ldots,H \tag{2.20}$$

Enfin, la contrainte (2.7.e) sur les cocycles entrants devient :

$$\sum_{a\in\omega^-(v)}\sum_{p\in\mathcal{P}}\delta_a^p y_p^h \leqslant 1 \qquad \forall v \in V \quad \forall h = 1,\ldots,P \tag{2.21}$$

Or tous les chemins passent par le cocycle entrant de la destination t. Donc la contrainte 2.21 pour $v = t$ inclut toutes les autres et équivaut à la contrainte 2.22. Celle-ci signifie que pour chaque numéro h de chemin actif, il correspond au plus un chemin de $\mathcal{P}$.

$$\sum_{p\in\mathcal{P}} y_p^h \leqslant 1 \qquad \forall h = 1,\ldots,H \tag{2.22}$$

Cette contrainte est incluse dans la relation entre les variables y_p^h relative à la combinaison convexe des points extrêmes. Cependant, le problème originel limite le nombre de chemins à au plus H. Il ne demande pas explicitement de les utiliser tous. Le modèle (KMFP_2), pour simplifier, oblige l'utilisation

d'exactement H chemins mais certains peuvent ne pas porter de flot. Ainsi, la contrainte 2.22 correspond mieux au problème. Nous la conservons donc pour ce modèle.

En outre, pour la fonction objectif, les variables z_h sont simplement remplacés par la somme sur $\mathcal{P}$ des variables de flot x_p^h. Ainsi, en appliquant une décomposition de Dantzig-Wolfe au modèle (KMFP$_2$), nous avons défini un nouveau modèle arc-chemin. Il diffère du modèle (KMFP$_1$) puisqu'il se fonde sur une discrétisation des variables de flot et de décision. Ce modèle s'établit de la façon suivante :

$$\text{(KMFP}_3\text{)} \begin{cases} \max \displaystyle\sum_{h=1}^{H} \sum_{p \in \mathcal{P}} x_p^h & & \\ s.c. & & \\ \quad \displaystyle\sum_{h=1}^{H} \sum_{p \in \mathcal{P}} \delta_a^p x_p^h \leq u_a & \forall a \in A & (a) \\ \quad x_p^h - u_p y_p^h \leqslant 0 & \forall h = 1 \dots H,\ \forall p \in \mathcal{P} & (b) \\ \quad \displaystyle\sum_{p \in \mathcal{P}} y_p^h \leqslant 1 & \forall h = 1 \dots H & (c) \\ \quad x_p^h \geqslant 0 & \forall h = 1 \dots H,\ \forall p \in \mathcal{P} & (d) \\ \quad y_p^h \in \{0,1\} & \forall h = 1 \dots H,\ \forall p \in \mathcal{P} & (e) \end{cases} \tag{2.23}$$

Cette formulation nécessite encore plus de variables que le premier modèle arc-chemin puisque les blocs de variables de flot et de support de flot ont été dupliqués H fois. Néanmoins il nous permet de mettre en œuvre dans le chapitre 3 des algorithmes performants.

D'autre part, les considérations de symétrie soulevées pour le modèle (KMFP$_2$) sont toujours valables. Considérons un ensemble de H chemins $(p_1, p_2, \cdots, p_H)$ constituant une solution optimale. Toute permutation de cet ensemble donne une autre solution optimale. Comme n'importe quel chemin peut être affecté à n'importe quelle position dans la solution de routage, cela peut potentiellement mener à un grand nombre de solutions « identiques ». Nous ajoutons donc également pour le modèle (KMFP$_3$) les contraintes de *variable ordering* (2.24).

$$\sum_{p \in \mathcal{P}} x_p^{h+1} - \sum_{p \in \mathcal{P}} x_p^h \leqslant 0 \qquad \forall h = 1 \cdots H-1 \tag{2.24}$$

2.4.3 Comparaison avec les modèles précédents

En comparaison avec le modèle (KMFP$_1$), le modèle (KMFP$_3$) n'a plus besoin de restriction sur le nombre de chemins actifs puisque c'est implicitement supposé à travers la discrétisation des variables. Cependant, une étape supplémentaire d'affectation des numéros aux chemins est requit par la contrainte (2.23.c). D'autre part, il est à noter qu'une solution du modèle (KMFP$_3$) peut toujours être transposée en une solution du modèle (KMFP$_1$) en utilisant les relations suivantes :

$$x_p = \sum_{h=1}^{H} x_p^h \quad \forall p \in \mathcal{P} \tag{2.25}$$

$$y_p = \max_{h=1,\ldots,H} y_p^h \quad \forall p \in \mathcal{P} \tag{2.26}$$

En fait, la relation entre ces deux modèles est même plus forte, comme le montre la propriété suivante.

Propriété 2.5. *Les modèles (KMFP$_1$) et (KMFP$_3$) sont équivalents.*

Démonstration. Soit $(\overline{x}_p^h, \overline{y}_p^h)$ une solution fractionnaire réalisable du modèle (KMFP$_3$). Les variables agrégées s'expriment de la façon suivante :

$$x_p = \sum_{h=1}^{H} \overline{x}_p^h \qquad \forall p \in \mathcal{P}$$

$$y_p = \sum_{h=1}^{H} \overline{y}_p^h \qquad \forall p \in \mathcal{P}$$

La solution (x_p, y_p) vérifie toutes les contraintes du modèle (KMFP$_1$).

Soit $(\overline{x}_p, \overline{y}_p)$ une solution fractionnaire réalisable du modèle (KMFP$_1$). Les variables désagrégées s'expriment de la façon suivante :

$$x_p^h = \frac{\overline{x}_p}{H} \qquad \forall p \in \mathcal{P} \quad \forall h = 1, \cdots, H$$

$$y_p^h = \frac{\overline{y}_p}{H} \qquad \forall p \in \mathcal{P} \quad \forall h = 1, \cdots, H$$

La solution (x_p^h, y_p^h) vérifie toutes les contraintes du modèle (KMFP$_3$).

Il existe donc une bijection entre les espaces réalisables des deux modèles. □

D'un autre coté, la comparaison avec le modèle (KMFP$_2$) aboutit à une situation différente.

Propriété 2.6. *Toute solution fractionnaire du modèle (KMFP$_3$) induit une solution fractionnaire du modèle (KMFP$_2$). La réciproque est fausse.*

Démonstration. Soit $(\overline{x}_p^h, \overline{y}_p^h)$ une solution fractionnaire réalisable du modèle (KMFP$_3$). Les variables sur les arcs s'expriment de la façon suivante :

$$x_a^h = \sum_{p \in \mathcal{P}} \delta_a^p x_p^h \qquad \forall a \in A \quad \forall h = 1, \cdots, H$$
$$y_a^h = \sum_{p \in \mathcal{P}} \delta_a^p y_p^h \qquad \forall a \in A \quad \forall h = 1, \cdots, H$$

La solution (x_a^h, y_a^h) vérifie toutes les contraintes du modèle (KMFP$_2$).

La réciproque est fausse. En effet, tout flot sur les arcs peut se décomposer en un ensemble de chemins élémentaires et de cycles élémentaires (cf. Ahuja et al [AMO93]). Or le modèle (KMFP$_2$) utilise une formulation qui ne peut pas interdire les cycles. Ainsi beaucoup de solutions réalisables du modèle (KMFP$_2$) ne peuvent pas être transposées en solutions du modèle (KMFP$_3$).

Par exemple, sur la figure 2.6, la solution fractionnaire réalisable du modèle (KMFP$_2$) utilise des cycles de support pour augmenter le flot. Le graphe est constitué d'arcs (trait plein) de capacité 2 à l'exception des arcs (u, v) et (u', v'), de capacité 1. La solution du flot entier maximum (1-séparable) entre s et t est un chemin ((s, u, v, t) ou (s, u', v', t)) portant une unité de flot. En pointillé est indiquée la décomposition en deux chemins et deux cycles du support de flot défini par la solution fractionnaire. Les valeurs en noir correspondent aux variables de flot x_a^h et les valeurs en gris aux variables de support de flot y_a^h. Les cycles permettent donc d'utiliser toute la capacité des arcs (u, v) et (u', v') sans être obliger de router du flot sur les arcs (v, u) et (v', u'). Ainsi cette solution fractionnaire est bien réalisable et porte un total de 2 unités de flot. Elle ne peut clairement pas être transposée en une solution du modèle (KMFP$_3$).

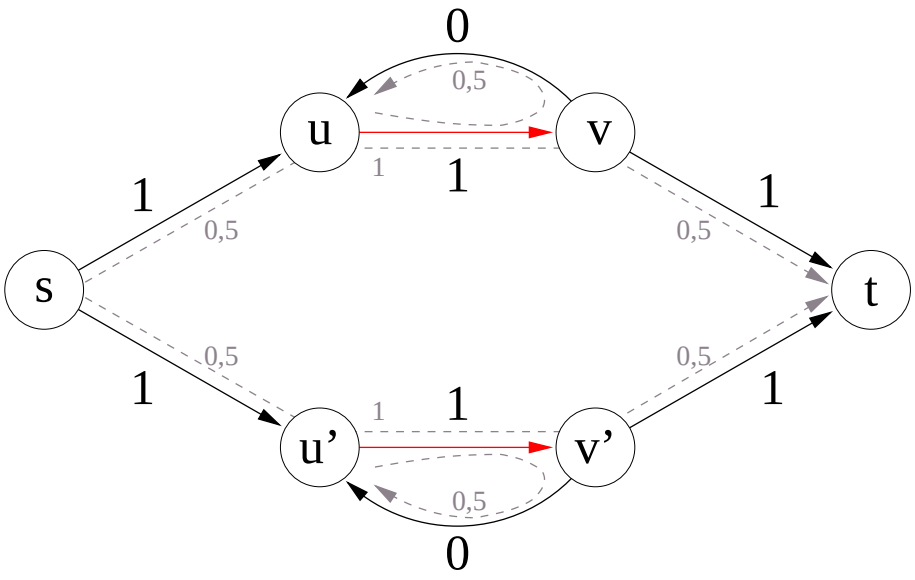


FIG. 2.6 – Utilisation de cycles de support pour augmenter le flot.

En supprimant les cycles, le modèle (KMFP$_3$) améliore nettement le modèle (KMFP$_2$) dans le sens où son espace réalisable contient moins de solutions. □

CHAPITRE 3

RÉSOLUTION PAR LA MÉTHODE DU BRANCH AND PRICE

L'application de la méthode du *Branch and Price* est une façon populaire et efficace de résoudre des programmes linéaires mixtes (variables fractionnaires et entières) comportant un grand nombre de variables. Elle consiste à utiliser la méthode de génération de colonnes à l'intérieur d'un algorithme de séparations et évaluations (*Branch and Bound*). Le principe du Branch and Bound a été présenté en 1960 par Land et Doig dans [LD60]. Peu avant, Ford et Fulkerson ([FF58]) ont proposé une résolution par génération de colonnes de problèmes de multiflots. Puis Danzig et Wolfe ([DW60]) ont développé cette idée dans le cadre de leur schéma de décomposition.

Cette méthode a été utilisée pour la première fois par Gilmore et Gomory dans le cadre d'une heuristique efficace pour résoudre le *Cutting Stock problem* [GG61, GG63]. La génération de colonnes ne tarde pas à trouver des applications dans de nombreux domaines. De nos jours, elle fait partie des méthodes prépondérantes pour traiter des modèles avec un grand nombre de variables.

Dans [DSD84], Desrosiers *et al.* introduisent une méthode embarquant la génération de colonne dans le cadre d'un Branch and Bound spécifique pour la programmation linéaire, afin de résoudre un problème de tournées de véhicules avec contraintes de fenêtres de temps. Ce principe trouvera de nombreuses applications dans la conception d'algorithmes exacts pour des programmes linéaires en nombres entiers [LD05]. Plus récemment, Barnhart *et al.* [BJN$^+$98] et Vanderbeck et Wolsey [VW96] ont décrit des algorithmes génériques pour résoudre des programmes linéaires en nombres entiers par génération de colonnes. Des études ont également porté sur des problématiques liées à l'implémentation de tels algorithmes comme, par exemple, les problèmes de stabilisation de la génération de colonnes [dMVDH99, Van05].

Dans un premier temps, nous détaillerons le principe de la méthode du Branch and Price dans un cadre général. Puis nous appliquerons ce principe au problème de flot k-séparable maximum modélisé dans le chapitre 2. Nous développerons chaque étape de la méthode : la relaxation linéaire, le sous-problème de génération de colonnes, les différentes règles de branchement, l'initialisation de l'algorithme et plusieurs améliorations. Enfin, nous analyserons les résultats numériques obtenus pour les différentes variantes de notre algorithme.

3.1 Principe général de la méthode du Branch and Price

3.1.1 Méthode de séparations et évaluations

La méthode de séparations et évaluations a été développée pour résoudre à l'optimum des problèmes discrets. Elle consiste en une exploration de l'espace des solutions privilégiant les parties « prometteuses ». L'exploration s'effectue par partitionnement de l'espace des solutions (généralement, on se limite à deux parties). Cette procédure s'appelle « séparation » ou « branchement » et peut être répétée récursivement sur chacune des parties ainsi définies. Cette succession de branchements forme naturellement une structure d'arbre, appelée « arbre de recherche » ou « arbre de décision », où chaque nœud correspond à une partie de l'espace des solutions. Si la procédure d'évaluation est considérée seule, l'exploration doit parcourir l'ensemble des solutions possibles (les feuilles de l'arbre de décision) qui est généralement beaucoup trop grand pour qu'une telle exploration soit acceptable.

Pour éviter l'exploration de la totalité de l'arbre de décision, il est nécessaire de pouvoir déterminer rapidement, pour une partie donnée de l'espace des solutions, une borne inférieure et supérieure de la valeur optimale. Ainsi, dans le cas d'une minimisation, il est possible de supprimer du parcours de l'arbre toutes les parties dont la borne inférieure est supérieure à la meilleure borne supérieure trouvée. En effet, cela signifie que cette partie de contient pas de solution optimale. De plus, la borne inférieure permet d'orienter la recherche vers les parties les plus « prometteuses ». Ainsi, une règle de parcours classique consiste à parcourir les nœuds de l'arbre de décision par borne inférieure croissante. L'algorithme s'arrête lorsque la borne inférieure du nœud étudié est supérieure ou égale à la meilleure borne supérieure trouvée. La meilleure solution trouvée est alors forcément optimale.

Ce parcours classique de l'arbre de décision est le plus utilisé en pratique. Notons toutefois que d'autres parcours sont possibles. Un parcours en profondeur d'abord a notamment fait l'objet de plusieurs études. Il consiste à choisir systématiquement un nœud fils du nœud traité comme nœud suivant. Ainsi, ce parcours permet d'obtenir rapidement des solutions entières fournissant des bornes supérieures de qualité. On peut alors espérer réduire rapidement les branches de l'arbre de décision à explorer. Il existe aussi des stratégies mixtes cherchant à profiter des qualités des différentes stratégies utilisées.

Les principales difficultés de l'application d'une méthode de séparations et évaluations résident généralement, d'une part, dans le choix des méthodes de calcul de bornes et, d'autre part, dans le choix des branchements à effectuer. En effet, il est important de pouvoir calculer des bornes inférieures et supérieures suffisamment proches de la valeur optimale afin de limiter au maximum l'exploration de l'arbre de décision. Cependant, si l'exploration reste de taille importante, il est nécessaire que le calcul des bornes ne prenne pas trop de temps. Ainsi, le choix des méthodes de calcul des bornes se portera principalement sur celles offrant un compromis entre la rapidité d'exécution et la qualité des bornes proposées.

D'autre part, Ryan et Foster [RF81] ont montré toute l'importance du choix des branchements en proposant une stratégie de branchement pour des problèmes de partitionnement (*Set Partitioning Problem*). Considérons un ensemble J d'objets possédant un coût c_j, $j \in J$, et un ensemble I de sous-ensembles J_i de J, $i \in I$. Le problème consiste à sélectionner des objets tels que, pour chaque sous-ensemble J_i, un et un seul objet soit sélectionné, tout en minimisant la somme des coûts des objets sélectionnés. En associant à chaque objet $j \in J$ une variable de décision $x_j \in \{0, 1\}$, le problème peut être modélisé par le programme linéaire en nombres entiers suivant :

$$\text{(SPP)} \begin{cases} \min \sum_{j \in J} c_j x_j \\ s.c. \\ \qquad \sum_{j \in J_i} x_j = 1 \quad \forall i \in I \quad (a) \\ \\ \qquad x_j \in \{0,1\} \quad \forall j \in J \quad (b) \end{cases} \tag{3.1}$$

L'approche classique de Dakin [Dak65] pour partitionner l'espace des solutions est de considérer une variable x_j et de la fixer soit à 0 soit à 1. Ainsi la partition est constituée de l'ensemble des solutions vérifiant $x_j = 0$ et de l'ensemble des solutions vérifiant $x_j = 1$. Fixer x_j à 1 est une contrainte forte car elle oblige à annuler toutes les variables en conflit avec x_j dans les sous-ensembles J_i contenant l'objet j. Cette contrainte réduit donc grandement l'ensemble des solutions possibles. Au contraire, la contrainte fixant x_j à 0 laisse beaucoup plus de liberté car elle n'apporte que peu de restrictions dans les contraintes (3.1.a). C'est ce déséquilibre que Ryan et Foster ont voulu réduire pour diminuer la taille de l'arbre de décision à explorer.

Leur stratégie de branchement se base sur la proposition suivante :

Proposition 3.1. *Soit A une matrice 0-1. Si une solution de base du système $Ax = 1$ est fractionnaire, i.e. au moins une composante de x est fractionnaire, alors il existe deux lignes r et s telles que la somme des variables communes soit fractionnaire :*

$$0 < \sum_{j : a_{rj} = a_{s_j} = 1} x_j < 1 \tag{3.2}$$

Ainsi, le branchement suivant est valide.

$$(\sum_{j \in J_r \cap J_s} x_j = 1) \quad \text{vs.} \quad (\sum_{j \in J_r \cap J_s} x_j = 0) \tag{3.3}$$

La branche de gauche signifie que les deux sous-ensembles J_r et J_s sont couverts par la même variable. La branche de droite signifie qu'ils sont couverts par des variables différentes. Toutefois, il est à noter que le terme de branchement de type Ryan et Foster est souvent employé dans la littérature pour évoquer tous branchements susceptibles de proposer un meilleur équilibrage de l'arbre de décision que l'approche classique. Par exemple, pour le problème précédent, un branchement possible consiste à partitionner un sous-ensemble J_i en deux parties égales J_i^1 et J_i^2 [AW00]. Ainsi le branchement suivant est valide :

$$(\sum_{j \in J_i^1} x_j = 0) \quad \text{vs.} \quad (\sum_{j \in J_i^2} x_j = 0) \tag{3.4}$$

Notons enfin que d'autres types de branchements ont été étudiés. Par exemple, dans [AW00], les

auteurs proposent des stratégies de branchement sur des combinaisons linéaires à coefficients entiers de variables.

3.1.2 Application aux programmes linéaires en nombres entiers

L'application de la méthode de séparations et évaluations aux programmes linéaires en nombres entiers (PLNE) revêt certaines spécificités que nous nous proposons d'exposer dans cette partie. Considérons pour cela le PLNE suivant.

$$\text{(PLNE)} \begin{cases} \min\ cx \\ s.c. \\ \qquad Ax \leqslant b \qquad (a) \\ \\ \qquad x \in \mathbb{Z}^n \qquad (b) \end{cases} \tag{3.5}$$

Un calcul classique d'une borne inférieure consiste à relâcher les contraintes d'intégrité (3.5.b) pour considérer uniquement des variables fractionnaires. La résolution de la relaxation linéaire (RL) ainsi obtenue est généralement rapide et la valeur de solution optimale fractionnaire est une borne inférieure de la solution optimale entière. Cependant, la « qualité » de cette borne inférieure dépend fortement du modèle. Pour estimer cette qualité, on présente généralement l'écart relatif, appelé *gap*, entre la valeur de la solution optimale fractionnaire et celle de la solution optimale entière pour l'espace de solution tout entier (nœud racine de l'arbre de décision). Plus cet écart est faible, plus le calcul de la borne est de bonne qualité puisqu'il permet alors de réduire fortement la taille de l'arbre à explorer.

$$\text{(RL)} \begin{cases} \min\ cx \\ s.c. \\ \qquad Ax \leqslant b \qquad (a) \\ \\ \qquad x \in \mathbb{R}^n \qquad (b) \end{cases} \tag{3.6}$$

D'autre part, la solution optimale fractionnaire peut se révéler être une solution entière. La valeur de cette solution est alors une borne supérieure pour la partie de l'espace des solutions considérée. Elle peut servir à élaguer certaines branches de l'arbre de décision.

De plus, la solution fractionnaire calculée par résolution de la relaxation linéaire sert généralement à définir un branchement à effectuer. Par exemple, dans le cas classique du branchement de Dakin, on choisit une variable x_j ayant un valeur x_j^* fractionnaire. Puis, le branchement s'effectue à partir des parties entières inférieure et supérieure (cf. figure 3.1) :

$$(x_j \leqslant \lfloor x_j^* \rfloor) \quad \text{vs.} \quad (x_j \geqslant \lceil x_j^* \rceil) \tag{3.7}$$

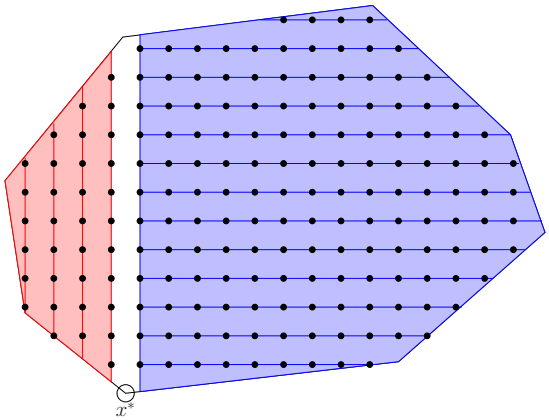


FIG. 3.1 – Séparation à partir de la solution fractionnaire x^*.

Ainsi l'application de la méthode de séparations et évaluation aux programmes linéaires en nombres entiers peut être représentée par la résolution de la relaxation linéaire et une procédure de branchement (figure 3.2). Cependant, pour de nombreux modèles, l'écart entre les solutions optimales fractionnaire et entière oblige à effectuer une exploration importante de l'arbre de décision. Pour réduire cet écart et donc l'exploration de l'arbre de décision, de nombreuses études ont portés sur des approches de types coupes et branchements (*Branch and Cut*).

Cette approche s'appuie sur l'analyse polyédrale pour définir des classes d'inégalités valides pour l'enveloppe convexe des solutions entières réalisables. Si toutes les facettes de l'enveloppe convexe pouvaient être ajoutées au programme linéaire, la relaxation linéaire donnerait directement la solution optimale entière. Cependant, ces facettes sont généralement très nombreuses et difficilement identifiables de manière ad hoc. De plus, la plupart sont inutiles pour déterminer l'optimum. L'idée est alors d'essayer d'identifier des inégalités violées à partir d'une solution optimale fractionnaire. L'ajout successif de ces inégalités au programme linéaire permet alors de réduire l'écart entre les optima entiers et fractionnaires.

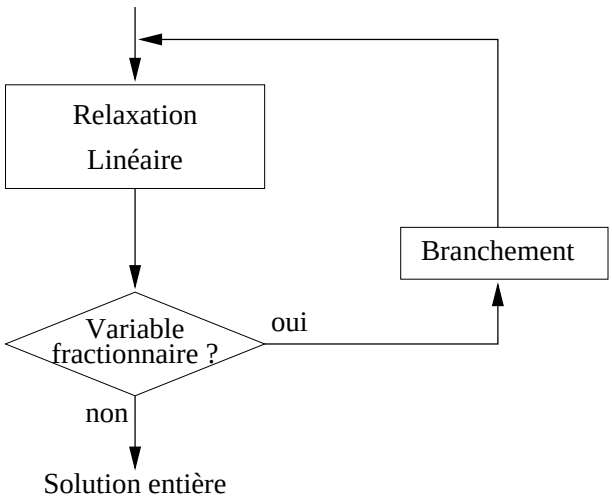


FIG. 3.2 – Méthode de séparations et évaluations.

3.1.3 Extension à la génération de colonnes

La méthode du Branch and Price est une généralisation du Branch and Bound autorisant l'utilisation de la génération de colonnes pour résoudre la relaxation linéaire. Plusieurs raisons conduisent à considérer des formulations avec un grand nombre de variables. En particulier, des techniques de reformulation telles que la décomposition de Dantzig-Wolfe aboutissent fréquemment à des modèles réduisant l'écart entre les solutions optimales entières et fractionnaires au prix d'un accroissement du nombre de variables. Les méthodes de génération de colonnes et de coupes sont donc des méthodes complémentaires pour renforcer la relaxation linéaire. Ces deux approches peuvent également être combinées au sein d'une méthode appelée *Branch and Cut and Price*.

À première vue, un Branch and Price semble être une simple combinaison d'un Branch and Bound embarquant une génération de colonne (cf. figure 3.3). Il existe cependant des difficultés dans la mise en œuvre d'une telle combinaison. En particulier, les branchements classiques peuvent complètement changer la nature du sous-problème de génération de colonnes. Si celui-ci devient trop compliqué à résoudre, cette approche perd tout son intérêt. Ainsi, la principale difficulté dans l'application d'une méthode de Branch and Price revient à trouver des branchements compatibles avec le sous-problème de génération de colonnes.

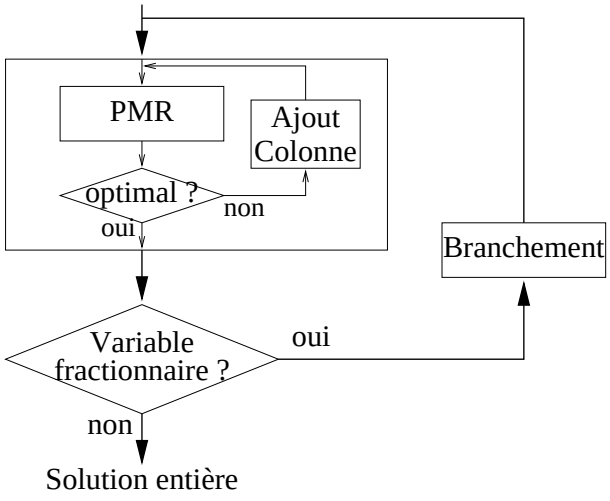


FIG. 3.3 – Principe général de la méthode du Branch and Price.

3.2 Génération de colonnes

Comme nous le détaillons dans notre présentation de la décomposition de Dantzig-Wolfe (cf. section 2.4.1), la génération de colonnes est une méthode fréquemment utilisée pour résoudre des programmes linéaires avec un grand nombre de variables (colonnes). Elle est basée sur la connaissance implicite de l'ensemble $\mathcal{X}$ des variables. À chaque itération, un programme maître restreint (PMR) est résolu puis la résolution d'un sous-problème (SP) permet de mettre à jour le (PMR) ou de passer à l'itération suivante. Le (PMR) consiste en une restriction de $\mathcal{X}$ à un sous-ensemble réalisable de variables $S \subset \mathcal{X}$. Après résolution à l'optimum du (PMR), il est nécessaire de savoir s'il existe ou non une variable de $\mathcal{X} \setminus S$

permettant d'améliorer la valeur de la fonction objectif. Ceci est effectué à travers la procédure de *pricing* (SP) qui consiste à utiliser les informations duales du (PMR) pour calculer le coût réduit le plus violé d'une variable de $\mathcal{X} \setminus S$. Si le coût réduit est améliorant, la variable associée est ajoutée au (PMR) pour les itérations suivantes. Sinon, c'est la preuve qu'il n'existe aucune variable améliorante. La génération de colonnes est arrêtée et la solution optimale du (PMR) est également optimale pour le problème.

3.2.1 Relaxation linéaire

Pour appliquer la méthode du Branch and Price sur le modèle (KMFP$_3$), il convient, dans un premier temps, de relâcher les contraintes d'intégrité. Cependant, l'application de la génération de colonnes n'est pas immédiate. Les contraintes de couplage (2.23.b) impose la génération d'une nouvelle contrainte à chaque génération d'un variable correspondant à un nouveau couple (h, p). De nouvelles variables duales apparaissent et modifient la structure du sous-problème. Néanmoins, la propriété suivante va permettre de simplifier la relaxation linéaire et de résoudre ce problème.

Propriété 3.1. *Il existe au moins une solution optimale de la relaxation linéaire de (KMFP$_3$) pour laquelle les contraintes de couplage (2.23.b) sont saturées.*

Démonstration. Soit (x^*, y^*) une solution optimale. Soit $\overline{y_p^h} = x_p^{h*}/u_p$ si $u_p > 0$ et 0 sinon, pour tout p de $\mathcal{P}$ et pour tout $h = 1 \dots H$. Alors les contraintes de couplage (2.23.b) implique que $\overline{y_p^h} \leqslant y_p^{h*}$, pour tout p de $\mathcal{P}$ et pour tout $h = 1 \dots H$. Donc $\sum_{p\in\mathcal{P}} \overline{y_p^h} \leqslant \sum_{p\in\mathcal{P}} y_p^{h*} \leqslant 1$ pour tout $h = 1 \dots H$ (contraintes (2.23.c)). Alors $(x^*, \overline{y})$ est une solution réalisable et fournit la même valeur que (x^*, y^*). Donc, $(x^*, \overline{y})$ est également une solution optimale. □

Ainsi, les contraintes de couplage peuvent être éliminées et les variables de décision y peuvent être remplacées avec les variables de flot x. Ceci mène à une version simplifiée de la relaxation linéaire, incluant les contraintes de variable-ordering :

$$
(\mathrm{RL}_3)\begin{cases}
\max \displaystyle\sum_{h=1}^{H}\sum_{p\in\mathcal{P}} x_p^h & & & \\
s.c. & & & \\
\quad \displaystyle\sum_{h=1}^{H}\sum_{p\in\mathcal{P}} \delta_a^p x_p^h \leqslant u_a & \forall a \in A & (a) & \\
\quad \displaystyle\sum_{p\in\mathcal{P}} \frac{x_p^h}{u_p} \leqslant 1 & \forall h = 1 \dots H & (b) & \\
\quad \displaystyle\sum_{p\in\mathcal{P}} x_p^{h+1} - \sum_{p\in\mathcal{P}} x_p^h \leqslant 0 & \forall h = 1 \dots H-1 & (c) & \\
\quad x_p^h \geqslant 0 & \forall h = 1 \dots H \quad \forall p \in \mathcal{P} & (d) &
\end{cases} \tag{3.8}
$$

3.2.2 Sous-problème

Après résolution du programme maître restreint, les informations duales sont envoyées au sous-problème. Notons respectivement $\pi_a \geqslant 0$, $\lambda^h \geqslant 0$ et $\nu^h \geqslant 0$ les variables duales associées aux contraintes primales $(3.8.a)$, $(3.8.b)$ et $(3.8.c)$ de la relaxation linéaire (RL_3). Alors, le sous-problème (SP) consiste à trouver une variable maximisant le coût réduit. Il peut être décomposé en H sous-problèmes indépendants (SP^h). Chacun se réduit à un problème de chemin optimal [MPRD99] pour un numéro de chemin h. Le coût réduit est donné par

$$\overline{c_p^h} = 1 - \sum_{a \in A} \delta_a^p \pi_a - \frac{\lambda^h}{u_p} - (\nu^{h-1} - \nu^h) \tag{3.9}$$

Le calcul du chemin élémentaire de coût maximal ne se réduit pas à un simple problème de plus court chemin puisque le coût réduit implique une combinaison de deux variables duales, π et λ (le coût associé aux contraintes de variable-ordering ne dépend que du numéro h et non des arcs du chemin). Ainsi, pour un numéro de chemin h donné, le sous-problème correspondant s'établit comme suit :

$$(\text{SP}^h) : \min_{p \in \mathcal{P}} w = \sum_{a \in A} \delta_a^p \pi_a + \frac{\lambda^h}{u_p} - 1 + (\nu^{h-1} - \nu^h) \tag{3.10}$$

Il est intéressant de remarquer qu'en l'absence d'une des deux variables duales, π ou λ, le problème correspond à un problème simple et classique de chemin optimal :

- Si la fonction à minimiser est $\sum_{a \in A} \delta_a^p \pi_a$, il s'agit du problème de plus court chemin dont les longueurs sur les arcs sont données par les variables duales π_a. De plus, les longueurs étant positives, l'algorithme de Dijkstra peut s'appliquer.
- Si la fonction à minimiser est $\frac{\pi^h}{u_p}$, il s'agit du problème de chemin de capacité maximale. Il consiste à maximiser l'arc de plus faible capacité emprunté par le chemin. Comme le problème de plus court chemin, il peut être résolution par un algorithme à label (*labelling algorithm*).

Plus précisément, pour tout problème de chemin optimal, Martins *et al.* [MPRD99] définissent le principe d'optimalité faible comme le fait qu'il existe un chemin optimal composé de sous-chemins optimaux. Puis, ils montrent que ce principe est une condition nécessaire et suffisante pour l'application de n'importe quel algorithme à label (par exemple, les algorithmes classiques de *label setting / label correcting* pour le problème de plus court chemin).

Malheureusement, notre sous-problème (SP) ne vérifie pas ce principe d'optimalité faible, comme illustré sur la figure 3.4. Pour chaque arc a, le premier nombre réfère à sa variable duale π_a tandis que le second réfère à sa capacité u_a. Dans cet exemple, la variable λ vaut 4. Ainsi le plus court chemin de s à v utilise l'arc a_2 puisque :

- Pour l'arc a_1, $\pi_1 + \frac{\lambda}{u_1} = 1 + \frac{4}{1} = 5$.
- Pour l'arc a_2, $\pi_2 + \frac{\lambda}{u_2} = 2 + \frac{4}{2} = 4$.

En revanche, le plus court chemin de s à t passe par l'arc a_1 puisque :

- Par l'arc a_1, $\pi_1 + \pi_3 + \frac{\lambda}{\min\{u_1,u_3\}} = 1 + 1 + \frac{4}{1} = 6$.
- Par l'arc a_2, $\pi_2 + \pi_3 + \frac{\lambda}{\min\{u_2,u_3\}} = 2 + 1 + \frac{4}{1} = 7$.

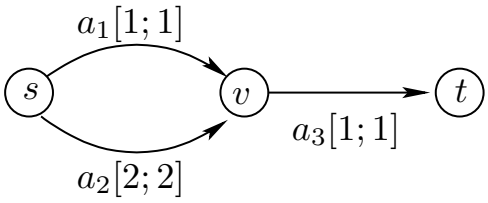


FIG. 3.4 – Échec du principe d'optimalité faible

Ainsi, aucun algorithme à label ne peut être utilisé pour calculer la solution optimale [MPRD99] et un algorithme spécifique doit être conçu.

Le principe de l'algorithme vient du constat suivant : étant donné la fonction objectif à minimiser dans (SP^h), un chemin est « meilleur » qu'un autre s'il a soit une plus petite longueur $\sum_{a\in A} \delta_a^p \pi_a$, soit une plus grande capacité u_p. Ainsi, en partant d'un plus court chemin (pour les longueurs π), un « meilleur » chemin pour la fonction objectif a donc forcément une capacité strictement plus grande. Notons u la capacité du plus court chemin. Il est possible de calculer un nouveau plus court chemin parmi les chemins de capacité strictement supérieure à u. Pour cela, il suffit de supprimer du graphe les arcs de capacité inférieure ou égale à u. Puis, en itérant ce raisonnement, on calcule des plus courts chemins sur un graphe contenant de moins en moins d'arcs puisque la capacité « limite » u augmente. L'algorithme s'arrête alors lorsqu'il n'y a plus de chemin entre s et t. Le calcul du chemin optimal p^* pour (SP^h) s'établit comme suit :

Algorithme 3.1 Calcul du chemin optimal pour (SP^h)

Soit $A' = A$
Soit $p^* = \emptyset$, $u^* = 0$, $w^* = \infty$
Répéter
  calculer un plus court chemin p de s à t sur A'
  Si $p = \emptyset$ **alors**
    arrêter
  else
    Soit u la capacité du chemin p et w son coût
    Si $w < w^*$ **alors**
      $p^* \leftarrow p$
      $u^* \leftarrow u$
      $w^* \leftarrow w$
    Fin Si
    $A' \leftarrow A' \setminus \{a \in A' \mid u_a \leqslant u\}$
  Fin Si
Jusqu'à $A' = \emptyset$
retourner (p^*, u^*, w^*)

Propriété 3.2. *L'algorithme 3.1 s'exécute en un temps polynomial.*

Démonstration. Il n'y a pas d'arc de coût négatif puisque $\pi_a \geqslant 0, \forall a \in A$. Ainsi, il n'existe aucun cycle de coût négatif et n'importe quel algorithme polynomial de plus court chemin peut être utilisé. Ensuite, à chaque itération de l'algorithme, au moins un arc est supprimé de l'ensemble A'. Ainsi, il y a au plus m itérations, c'est-à-dire m calculs de plus court chemin. □

La résolution de la relaxation linéaire par génération de colonne s'arrête alors dès que $w^* \leqslant 0$. Si la solution est fractionnaire, il convient de passer à la phase de branchement de l'algorithme de Branch and Price.

3.3 Règles de branchement

Une fois la relaxation linéaire résolue par génération de colonne, si la solution obtenue est fractionnaire, il est nécessaire de procéder à la séparation de l'espace des solutions. Cependant, dans le cadre du Branch and Price, il est impératif de veiller à ne pas modifier la procédure de génération de colonnes.

Ainsi, le branchement classique de Dakin [Dak65] sur les variables y_p^h ne peut pas être appliqué. D'une part, une branche impose au chemin p d'être le chemin actif numéro h, i.e. $y_p^h = 1$, et d'autre part, l'autre branche interdit au chemin p d'être le chemin actif numéro h, i.e. $y_p^h = 0$. La première branche est facile à gérer puisqu'il devient inutile de générer des chemins supplémentaires pour le numéro h. En revanche, pour la deuxième branche, rien ne garantit que la solution du sous-problème de chemin optimal ne sera pas justement p. Il est donc nécessaire dans ce cas d'utiliser un algorithme de type k plus courts chemins pour éviter les chemins « interdits ». Ceci complique considérablement la procédure de génération de colonnes.

En général, on constate qu'il est plus simple de ne pas modifier la structure du problème de pricing en basant le branchement sur les variables de la formulation originale et non sur les variables de la reformulation pour la génération de colonnes [DDSS95, BHV00]. Considérons alors le branchement de Dakin sur les variables de flot du modèle arc-sommet y_a^h. Sur une branche, le chemin numéro h est obligé de passer par l'arc a, i.e. $y_a^h = 1$, et sur l'autre branche, le chemin numéro h ne doit pas passer par l'arc a, i.e. $y_a^h = 0$. Cette fois, la deuxième branchement est facilement compatible avec la génération de colonnes puisqu'il suffit d'« interdire » l'arc a pour le chemin numéro h. En revanche, la première branche détruit la structure du sous-problème. En effet, elle oblige le sous-problème a généré un chemin optimal passant par l'arc $a = (i, j)$, ce qui peut être résolu par le calcul de deux sous-chemins optimaux : de s à i et de j à t. Mais après plusieurs branchements de ce type, le chemin solution du sous-problème a obligation de passer par un ensemble d'arcs. Il est alors impossible de résoudre le sous-problème de manière efficace.

Ryan et Foster [RF81] ont été parmi les premiers à proposer une règle de branchement susceptible d'améliorer l'exploration de l'espace des solutions. Au lieu de séparer l'ensemble des solutions en fonction d'une seule variable, ils considèrent, pour deux lignes données du problème maître, d'une part, les colonnes couvrant les deux lignes, et d'autre part, les colonnes ne couvrant qu'une seule des deux lignes. Ce principe n'est pas applicable pour notre problème car il complexifie grandement le sous-problème.

3.3.1 Branchement classique

Plus récemment, Barnhart *et al.* [BHV00] ont proposé une généralisation de la règle de Ryan et Foster pour un problème de flot entier (flot routé sur un seul chemin). Elle se base sur l'analyse des variables de flot agrégé $x_a^h = \sum_p \delta_a^p x_p^h, \forall a \in A$. Pour un numéro de chemin h donné, la solution est fractionnaire si le flot concernant ce numéro h ne peut pas être porté par un seul chemin. Cela signifie qu'il existe au moins un nœud, appelé nœud de divergence, à partir duquel le flot se sépare pour suivre plusieurs routes distinctes.

Ainsi, un nœud de divergence est un nœud $d \in N$ tel le flot agrégé arrive d'un seul arc et sort par plusieurs arcs (cf. figure 3.5). Pour un numéro de chemin h donné, une divergence se produit lorsque les variables de support de flot y_p^h sont fractionnaires. Sur la figure 3.5, ces fractions sont respectivement $7/10$, $1/10$ et $2/10$. Ainsi, la solution utilise une partie fractionnaire de chaque chemin p.

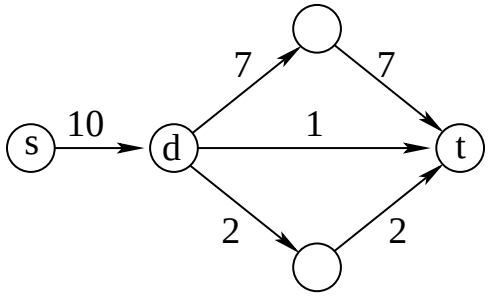


FIG. 3.5 – Divergence du flot agrégé x^h au nœud d.

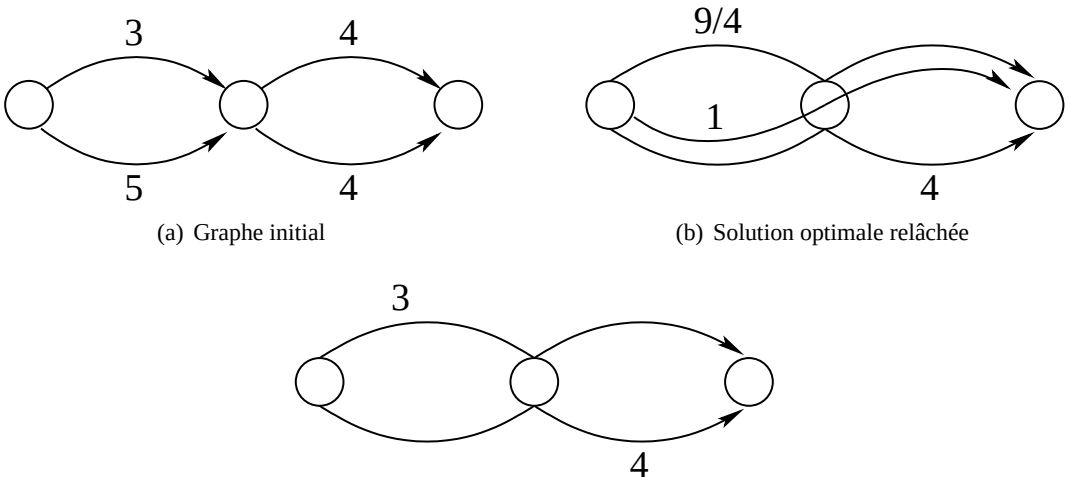
Notons $\omega^+(d)$ l'ensemble des arcs sortants du cocycle du nœud d. Soient ω_1 et ω_2 deux sous-ensembles formant une partition de $\omega^+(d)$ telle que chaque sous-ensemble contienne au moins un arc portant une quantité strictement positive de flot. Considérons ensuite $P_1 \subset \mathcal{P}$, respectivement $P_2 \subset \mathcal{P}$, l'ensemble des chemins passant par le nœud d et empruntant un arc de ω_1, respectivement ω_2. Puisque le flot x^h doit être entier (i.e. inséparable), il ne peut emprunter à la fois un arc de l'ensemble ω_1 et un arc de ω_2. Alors, le branchement suivant est valide :

$$(\sum_{p \in P_1} y_p^h = 0) \quad \text{vs.} \quad (\sum_{p \in P_2} y_p^h = 0) \tag{3.11}$$

Sur la première branche, on interdit les chemins empruntant un arc de ω_1 et sur la deuxième branche, on interdit ceux empruntant un arc de ω_2. Ceci revient à interdire certains arcs pour le sous-problème de génération de colonnes. Comme nous l'avons vu précédemment, l'interdiction d'un ou plusieurs arcs, analogue à la suppression de ces arcs du graphe pour un sous-problème donné, ne modifie pas la structure du sous-problème. Cependant, il est à noter que ce branchement n'oblige en rien le passage par un des sous-ensembles ω_1 ou ω_2. Ainsi, les solutions potentielles dont le chemin numéro h ne passe pas par le nœud d se retrouvent de chaque coté du branchement. D'un point de vue théorique, ce dédoublement n'empêche en rien la convergence de l'algorithme. Néanmoins, d'un point de vue pratique, il peut entrainer l'exploration de plusieurs branches équivalentes de l'arbre de décision. Ceci reste malheureusement inévitable.

Pour une meilleure efficacité, le branchement doit être appliqué sur le premier nœud de divergence. D'autre part, il est important d'essayer d'équilibrer les deux sous-ensembles ω_1 et ω_2 aussi bien au niveau de leur cardinalité (nombre d'arcs) que de la somme des fractions y_p^h correspondantes.

Il est à noter que cette règle de branchement ne peut pas être facilement adaptée au premier modèle (KMFP$_1$) car plusieurs chemins peuvent être nécessaires au nœud de divergence. Cette situation est illustrée sur la figure 3.6. La figure 3.6(a) représente un graphe où sont rapportées les capacités. Nous souhaitons transporter une quantité maximale de flot de s à t en utilisant au plus $H = 2$ chemins. La figure 3.6(b) présente la solution optimale de la relaxation linéaire du modèle (KMFP$_1$). Toutes les capacités sont vérifiées par le flot agrégé et la somme $\sum_{p\in\mathcal{P}} y_p = \sum_{p\in\mathcal{P}} x_p/u_p = 3/4 + 1/4 + 1 = 2$ satisfait à la contrainte du nombre limite H de chemins actifs. Clairement, cette solution utilises trois chemins et aucun branchement du type défini précédemment ne peut être appliqué. La solution optimale entière est indiquée sur la figure 3.6(c).



(a) Graphe initial

(b) Solution optimale relâchée

(c) Solution optimale entière

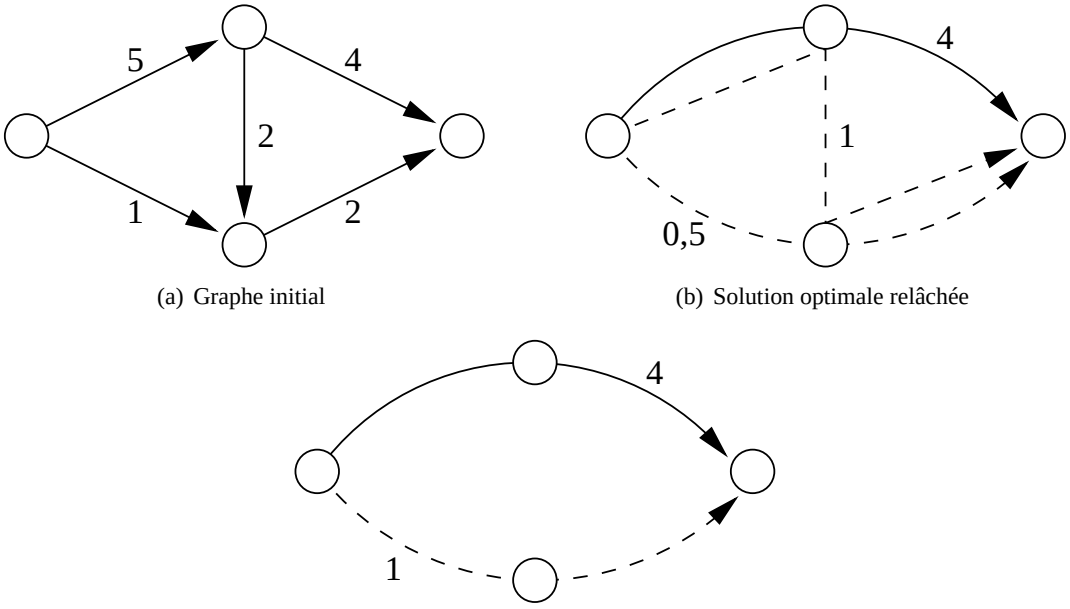
FIG. 3.6 – Divergence du flot agrégé pour le modèle (KMFP$_1$).

3.3.2 Branchement alternatif

Un des inconvénients de la précédente règle de branchement est la nécessité de combiner plusieurs branchements successifs avant d'obtenir des contraintes efficaces. En partant du constat que certains arcs seulement limitent le flot (les goulots d'étranglement), nous proposons une autre règle de branchement basée sur la relation entre le flot transporté par des chemins h empruntant un même arc et la capacité de cet arc.

Pour illustrer cette situation, considérons le graphe de la figure 3.7(a) sur lequel sont rapportées les capacités. Nous souhaitons transporter une quantité maximale de flot de s à t en utilisant au plus $H = 2$ chemins. La solution optimale de la relaxation linéaire (LR$_3$) est représentée sur la figure 3.7(b). Elle est constitué d'un seul chemin de capacité 4 pour $h = 1$ (trait plein) et de deux chemins de capacités 2 et 1 pour $h = 2$ (trait pointillé). L'arc de capacité 5 empêche d'utiliser pleinement les chemins de capacité 4 et 2. Mais une unité de flot transportée sur le chemin de capacité 2 représente une fraction de $\frac{1}{2}$ pour

$h = 2$. La solution optimale peut donc encore utiliser le chemin de capacité 1 pour router $\frac{1}{2}$ unité de flot. On constate alors que pour $h = 2$ le flot route $1,5$ unités de flot alors qu'il emprunte un arc de capacité 1. La solution optimale entière est indiquée sur la figure 3.7(c).



(a) Graphe initial

(b) Solution optimale relâchée

(c) Solution optimale entière

FIG. 3.7 – Illustration de l'application du branchement alternatif.

Considérons un arc $a \in A$ et $S \subset \{1, \cdots, H\}$ le sous-ensemble des numéros de chemins h empruntant l'arc a, i.e. $\sum_{p \in \mathcal{P}} \delta_a^p x_p^h > 0$, $\forall h \in S$. Alors, soit tous les numéros de chemins h de S empruntent l'arc a ou alors au moins l'un d'entre eux n'utilise pas cet arc a. Plus formellement, la règle de branchement s'établit comme suit :

$$(\sum_{h \in S} y_a^h = |S|) \quad \text{vs.} \quad (\sum_{h \in S} y_a^h \leqslant |S| - 1) \tag{3.12}$$

La première branche implique que tout le flot pour les numéros h de S passe par l'arc a. Or, nous avons vu qu'imposer le passage par un arc modifie la structure de la procédure de génération de colonnes. C'est pourquoi il est plus intéressant de contraindre la quantité de flot routé pour ces chemins.

$$(\sum_{h \in S} y_a^h = |S|) \quad \Rightarrow \quad (\sum_{h \in S} \sum_{p \in \mathcal{P}} x_p^h \leqslant u_a) \tag{3.13}$$

Il n'y a pas équivalence entre les deux contraintes puisque les flots correspondant aux numéros de chemins de S peuvent être limités à la capacité de l'arc a sans que tous empruntent cet arc. De toute façon, l'idée de ce branchement est bien de contraindre le flot de chaque numéro de chemin à respecter la capacité des arcs qu'il emprunte. D'autre part, il est à noter que cette restriction est plus forte que la contrainte de capacité (2.23.a) :

$$\sum_{h\in S}\sum_{p\in\mathcal{P}}\delta_a^p x_p^h \leqslant \sum_{h\in S}\sum_{p\in\mathcal{P}} x_p^h \leqslant u_a \tag{3.14}$$

Ce branchement ajoute donc une nouvelle contrainte dans le programme maître restreint. Une nouvelle variable duale doit être prise en compte dans la génération de colonnes. Elle altère alors la structure du sous-problème et le rend difficile à résoudre. Ainsi, une solution consiste à utiliser la formulation arc-sommet pour les contraintes relatives aux branchements et à lier les variables arc-sommet avec les variables arc-chemin. Cette approche est basée sur la formulation *Explicit Master* dans le *Robust Branch and Cut and Price* proposé par Fukasawa et al. [FLL$^+$06]. Le programme maître restreint s'écrit alors :

$$(\mathrm{RL}_4)\begin{cases} \max \displaystyle\sum_{h=1}^{H}\sum_{p\in\mathcal{P}} x_p^h & & & \\ s.c. & & & \\ \quad \displaystyle\sum_{p\in\mathcal{P}}\delta_a^h x_p^h - x_a^h \leqslant 0 & \forall h = 1\dots H & \forall a\in A & (a) \\ \quad \displaystyle\sum_{h=1}^{H}\sum_{p\in\mathcal{P}}\delta_a^p x_p^h \leqslant u_a & \forall a\in A & & (b) \\ \quad \displaystyle\sum_{p\in\mathcal{P}}\frac{x_p^h}{u_p} \leqslant 1 & \forall h = 1\dots H & & (c) \\ \quad x_p^h \geqslant 0 & \forall h = 1\dots H & \forall p\in\mathcal{P} & (d) \\ \quad x_a^h \geqslant 0 & \forall h = 1\dots H & \forall a\in A & (e) \end{cases} \tag{3.15}$$

Le branchement peut alors être appliqué sur les variables x_a^h tandis que la génération de colonnes traitent les variables x_p^h. Ces deux types de variables sont liés par la contrainte (3.15.a). Les variables duales μ_a^h associées à cette contrainte s'ajoutent simplement aux variables duales π_a sur les arcs. La structure du sous-problème n'est donc pas modifié.

Appliqué aux variables sur les arcs, la première branche devient :

$$(\sum_{h\in S}\sum_{p\in\mathcal{P}} x_p^h \leqslant u_a) \quad \Leftrightarrow \quad (\sum_{h\in S}\sum_{a'\in\omega^+(s)} x_{a'}^h \leqslant u_a) \tag{3.16}$$

L'autre branche porte sur les variables de support de flot y_p^h. Cependant, après relaxation des contraintes d'intégrité, elles se plaquent sur le rapport x_p^h/u_p. La contrainte associée à cette branche n'est alors pas assez restrictive. Il est préférable de se ramener à l'interdiction de l'arc a pour un numéro de chemin h donné. Ceci est possible car la contrainte relative à cette deuxième branche est équivalente à interdire l'arc a à au moins un numéro de chemin de S. Elle peut alors être reformulée comme suit :

$$(\sum_{h\in S} y_a^h \leqslant |S| - 1) \quad \Leftrightarrow \quad (\exists h\in S \mid y_a^h = 0) \tag{3.17}$$

Ce branchement revient à créer autant de branches que de numéros de chemins h dans S. Il est probable que la multiplicité de ces branches restreigne l'efficacité de l'exploration de l'arbre de décision. C'est pourquoi nous nous limiterons à un ensemble S réduit à un unique numéro de chemin. La règle de branchement s'établit alors sous une forme simplifiée :

$$(\sum_{a' \in \omega^+(s)} x^h_{a'} \leqslant u_a) \quad \text{vs.} \quad (y^h_a = 0) \tag{3.18}$$

Ce branchement ne peut s'appliquer que si le flot fractionnaire relatif à un numéro de chemin h route une quantité plus grande que la capacité d'un arc a qu'il emprunte. Sinon, le branchement classique doit être utiliser.

3.4 Initialisation et améliorations

3.4.1 Initialisation

Pour pouvoir appliquer la méthode de génération de colonnes, il est nécessaire de disposer d'une base valide initiale. Il faut donc générer préalablement un certain nombre de variables ou colonnes permettant de satisfaire les contraintes de la relaxation linéaire. Dans le cadre du problème de flot maximal étudié ici, les contraintes sont telles qu'une base vide suffit (ce sont les variables d'écart qui constituent la base initiale). Il n'y a alors pas de problème pour initier la génération de colonnes. Cependant, pour d'autres problèmes, par exemple lorsqu'une demande fixe doit être routée, le problème peut s'avérer complexe à résoudre.

D'autre part, il est usuel de rechercher une solution entière initiale avant d'entamer l'algorithme du Branch and Price. Cette solution entière n'est pas indispensable mais elle fournit une borne supérieure qui peut permettre de limiter l'exploration de l'arbre de décision. Suivant le problème étudié, la recherche d'une telle solution peut être complexe. Vu la faible importance que cette borne supérieure peut avoir sur le Branch and Price (cela dépend du problème et de la qualité de la borne), il est important de privilégier, dans ce calcul d'une solution entière initiale, la rapidité d'exécution à la qualité de la solution. C'est pourquoi ce calcul utilise principale des heuristiques ou des algorithmes d'approximation polynomiaux.

Dans le cadre du problème du flot k-séparable maximal, nous avons utilisé l'algorithme d'approximation proposé par Baier *et al.* [BKS05]. Cet algorithme polynomial résout à l'optimum le problème du flot maximal uniforme exactement k-séparable (k chemins routant chacun une fraction $\frac{1}{k}$ du flot total). Par ailleurs, Baier *et al.* montrent que la solution optimale obtenue pour ce problème route une quantité de flot supérieure à la moitié de l'optimum du problème de flot k-séparable maximal.

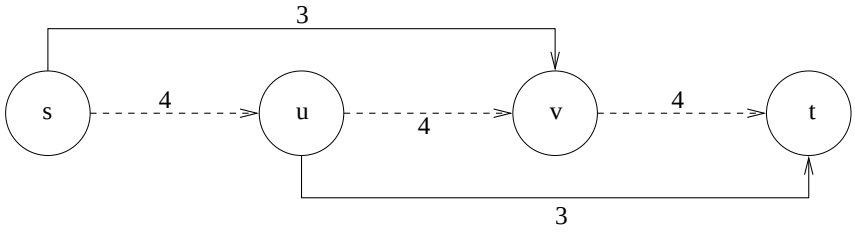
Cette borne d'approximation est relativement faible du fait de la contrainte d'uniformité du flot sur chaque chemin. Cependant, les chemins constituant cette solution peuvent servir de support de flot pour le calcul d'une meilleure solution entière sans cette contrainte d'uniformité. Une fois le support de flot fixé, l'optimisation des flots sur chaque chemin est assez simple et rapide à effectuer.

L'algorithme de Baier *et al.* est similaire à l'algorithme de Ford & Fulkerson pour le problème de flot

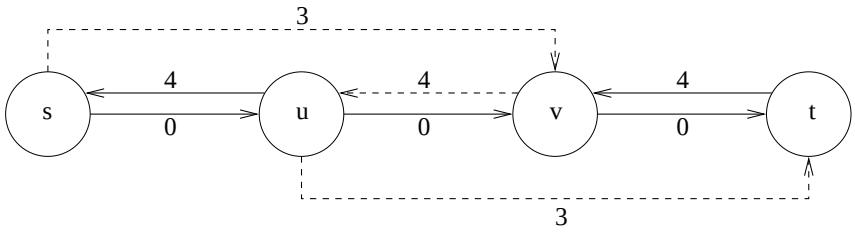
maximal. Il consiste à chercher une chaîne augmentante dans un graphe résiduel. À chaque itération i, la chaîne augmentante permet de déduire i chemins routant chacun une quantité de flot égale à la capacité de la chaîne.

Cependant, la contrainte d'uniformité se retrouve dans les capacités résiduelles. En effet, considérons $P_1, \cdots, P_i$ les i chemins résultants des itérations précédentes et f_i la quantité commune de flot qu'ils routent. Pour chaque arc a de A, notons q_a^i le nombre de chemins parmi $P_1, \cdots, P_i$ passant par a. Du fait de la contrainte d'uniformité des flots routés sur chaque chemin, si un nouveau chemin passe par un arc a, la quantité de flot routé par chaque chemin est majorée par $\frac{u_a}{q_a^i+1}$ qui est donc la capacité résiduelle de l'arc a.

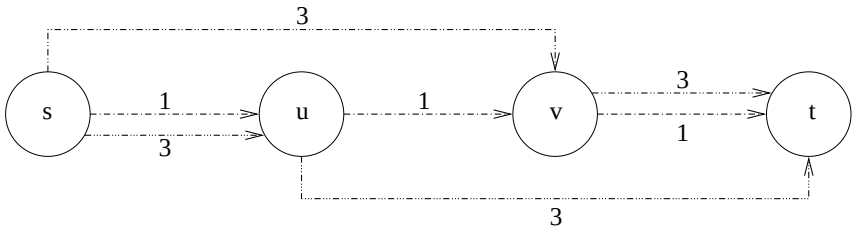
De plus, il faut considérer un arc inverse à a dans le graphe résiduel. Si la chaîne augmentante emprunte cet arc inverse, cela signifie que le flot sur l'arc a doit être diminué. Cela induit une décomposition en chemins illustrée sur la figure 3.8. La figure 3.8(a) présente la chaîne augmentante initiale (s, u, v, t) de capacité 4. Puis, figure 3.8(b), la chaîne augmentante dans le graphe résiduel résultant (s, v, u, t), de capacité 3, utilise l'arc inverse de l'arc (u, v). La figure 3.8(c) décrit la décomposition en chemins induite par cette chaîne : les chemins (s, v, t) et (s, u, t) routent chacun 3 unités de flot et le chemin (s, u, v, t) ne route plus qu'une unité de flot.



(a) Chaîne augmentante dans le graphe initial



(b) Chaîne augmentante dans le graphe résiduel



(c) Décomposition en chemins

FIG. 3.8 – Décomposition en chemins à partir d'une chaîne augmentante dans un graphe résiduel.

Ainsi, si la chaîne augmentante emprunte un arc inverse et route une quantité de flot strictement plus

grande que f_i, le nombre total de chemins après décomposition serait strictement plus grand que $i+1$. La capacité de l'arc inverse dans le graphe résiduel est donc égale à f_i.

La chaîne augmentante est alors obtenue par calcul du chemin de capacité maximale dans le graphe résiduel. La capacité de ce chemin indique la nouvelle quantité de flot f_{i+1} commune aux $i+1$ chemins qu'il reste à déterminer. Pour cela, on envoie f_{i+1} unités de flot sur chaque chemin $P_1, \cdots, P_i$ et sur la chaîne augmentante. Par définition du graphe résiduel, le flot ainsi obtenu est réalisable dans le graphe G. De plus, il est f_{i+1} intégral (le flot sur chaque arc est un multiple entier de f_{i+1}) et route une quantité totale de flot égale à $(i+1)f_{i+1}$. Ainsi, on obtient un nouvel ensemble de $i+1$ chemins routant chacun f_{i+1} unités de flot. Cet ensemble de chemins forme un flot uniforme exactement $(i+1)$-séparable maximal. Alors, au bout de k itérations, l'algorithme a bien calculé un flot uniforme exactement k-séparable maximal.

L'algorithme de Baier *et al.* fournit donc une solution entière initiale pour notre algorithme de Branch and Price. La propriété suivante propose un résultat théorique sur la qualité d'approximation de cet algorithme pour le problème du flot k-séparable maximal.

Propriété 3.3. *La quantité totale de flot routé par un flot uniforme exactement k-séparable maximal est supérieure à la moitié du flot routé par un flot k-séparable maximal.*

Démonstration. Considérons un flot k-séparable maximal F^* de valeur OPT. Notons D la valeur $\frac{OPT}{2k}$.

Pour chaque chemin p de F^*, routant f_p unités de flot, considérons $\lfloor \frac{f_p}{D} \rfloor$ copies de p routant chacun D unités de flot. Le flot F' ainsi défini est réalisable puisque pour chaque chemin p de F^* le flot total routé sur celui-ci est plus petit que f_p. De plus, comme le nombre de chemins de $F*$ est majoré par k, nous obtenons l'inégalité suivante :

$$\sum_{p \in F^*} \lfloor \frac{f_p}{D} \rfloor \geq \sum_{p \in F^*} (\frac{f_p}{D} - 1) = \frac{OPT}{D} - k = k \tag{3.19}$$

Il existe donc k chemins parmi ceux de F' routant une quantité totale de flot $kD = \frac{OPT}{2}$. □

Comme défini dans [Lap00], le coefficient d'approximation ρ d'un algorithme est un majorant de l'erreur relative entre la solution fournie par l'algorithme et la solution optimale du problème. L'algorithme de Baier *et al.* est donc au moins un algorithme de $\frac{1}{2}$-approximation. Il est possible que le coefficient d'approximation soit meilleur mais nous n'en avons pas la démonstration. Cependant, le graphe de la figure 3.9 permet de démontrer un minorant pour ce coefficient d'approximation. Ce graphe est constitué de :

- quatre noeuds s, u, v et t,
- k arcs de capacité k entre s et u,
- un arc de capacité k entre u et v,
- k arcs de capacité k entre v et t,
- $k-1$ arcs de capacité 1 entre s et t.

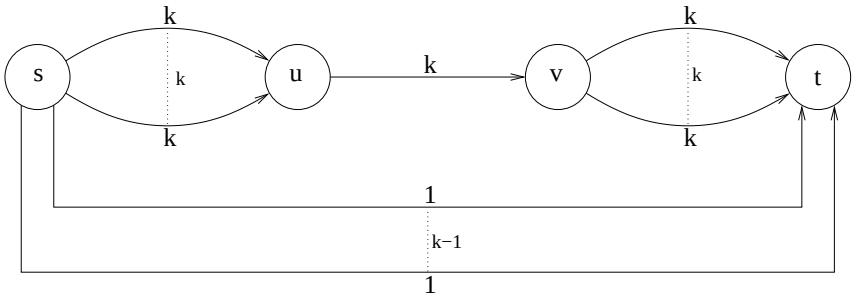


FIG. 3.9 – Démonstration d'un minorant du coefficient d'approximation de l'algorithme de Baier *et al.*.

Sur le graphe de la figure 3.9, la solution optimale du problème du flot k-séparable maximal est clairement constituée d'un chemin (s, u, v, t) routant k unités de flot et de $k-1$ chemins (s, t) routant chacun une unité de flot. Ainsi, la valeur de la solution optimale est de $2k-1$. Pour le problème du flot uniforme exactement k-séparable maximal, une solution optimale est forcément constitué de k chemins routant chacun une unité de flot, soit un total de k unités de flot. L'erreur relatif est donc dans ce cas égale à $\frac{k-1}{2k-1}$. D'où l'encadrement suivant du coefficient d'approximation :

$$\frac{k-1}{2k-1} \leq \rho \leq \frac{1}{2} \tag{3.20}$$

Le coefficient d'approximation est relativement faible mais nous constaterons, dans la section 3.5 sur les résultats numériques, que la post-optimisation sur les chemins fournis par cet algorithme donne souvent une solution de bonne qualité mais rarement optimale.

3.4.2 Pool de colonnes

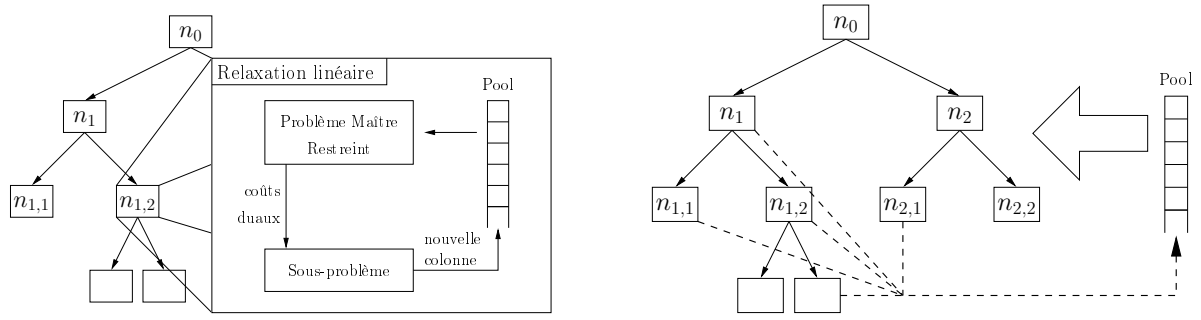
Dans le cadre d'un Branch and Price, chaque nœud de l'arbre de décision correspond à une relaxation linéaire sur une partie de l'espace des solutions. La relaxation linéaire est résolue par génération de colonnes : le Problème Maître Restreint résout le programme linéaire sur un sous-ensemble de variables puis il transmet les coûts duaux au sous-problème qui, en retour, fournit une nouvelle colonne.

Seuls les variables en base, c'est-à-dire ici les chemins actifs, sont nécessaires pour effectuer le branchement. Nous pouvons donc nous permettre de conserver en mémoire uniquement les colonnes correspondant aux variables en base. Cependant, une variable peut très bien entrer de nouveau en base après en être sortie. Il faudrait alors la générer de nouveau ce qui semble être une perte de temps. Supprimer une variable hors base déjà générée ne se justifie que lorsqu'il y a beaucoup de variables générées ce qui peut ralentir la résolution du programme maître restreint. De plus, techniquement, la suppression d'une variable générée n'est pas négligeable en terme de temps de calcul. C'est pourquoi toutes les colonnes générées sont stockées dans un pool.

La figure 3.10 illustre deux stratégies pour la gestion du pool de colonnes. La première, figure 3.10(a), considère un pool local à chaque relaxation linéaire, c'est-à-dire à chaque nœud de l'arbre de décision. Il n'y a donc aucun échange d'informations entre les nœuds et chaque relaxation linéaire repart de zéro.

Cette stratégie a l'avantage de ne considérer que les colonnes intéressantes pour la partie de l'espace des solutions correspondant au nœud traité. De plus, un pool de taille réduite peut permettre des calculs plus rapide pour la résolution du PMR. Cependant, il est fort probable qu'avec cette stratégie de nombreuses colonnes doivent être générées plusieurs fois dans différents nœuds de l'arbre de décision.

La stratégie illustrée par la figure 3.10(b) considère un pool global à tous les nœuds de l'arbre de décision. Ainsi, chaque relaxation linéaire entame sa résolution avec un sous-ensemble de colonnes déjà générées. Cette stratégie a l'avantage d'éviter de générer plusieurs fois la même colonne, d'où un gain de temps évident. Cependant, au fil de l'exploration de l'arbre de décision, le pool ne cesse de grossir ce qui peut ralentir la résolution du PMR. Différentes approches sont alors possibles pour réduire la taille du pool. Par exemple, un compteur peut être affecté à chaque colonne, similaire à un *Time To Live*. Ce compteur décroît à chaque résolution du PMR et n'est réinitialisé que lorsque la variable correspondante est en base. Si le compteur s'annule, la colonne est supprimée du pool car elle semble inutile. Avec cette approche, certaines colonnes peuvent être générée plusieurs fois mais de façon beaucoup moins fréquente qu'avec la stratégie du pool local. Cependant, cette procédure n'est pas négligeable en temps de calcul. Dans notre cas, nous n'avons pas jugé suffisant le gain de temps obtenu par la réduction du pool pour justifier la mise en place d'une telle procédure.



(a) Pool local (b) Pool global.

FIG. 3.10 – Gestion du pool de colonnes.

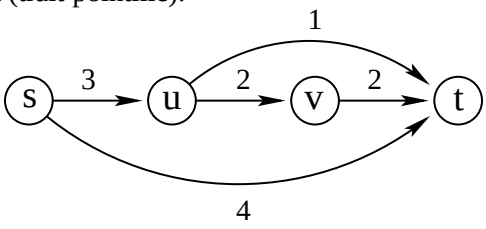
3.4.3 Variable ordering

Comme nous l'avons expliqué à propos des modèles ($KMFP_2$) et ($KMFP_3$), la discrétisation des variables a engendrée une symétrie dans les modèles. Un grand nombre de solutions sont alors équivalentes à une permutation près. Pour réduire l'espace des solutions, nous avons introduit les contraintes de variable ordering qui induisent une hiérarchie entre les variables. De plus, nous avons constaté que les coûts duaux associés à ces contraintes ne modifient pas la structure du sous-problème.

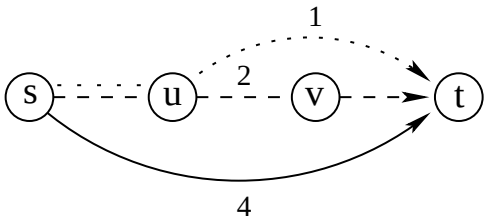
Cependant, dans certains cas, les contraintes de variable ordering complexifient la résolution du problème. C'est notamment le cas lorsque la valeur de la solution optimale est égale au flot maximal pouvant être routé entre la source et la destination (la contrainte sur la largeur du flot devient inutile). Cette situation est illustré sur la figure 3.11. Le graphe et ses capacités sont indiqués sur la figure 3.11(a). Le flot maximal pouvant circuler de s à t est évidemment de 7. Ce flot peut être décomposé en 3 chemins : (s, t) route 4 unités de flot, (s, u, v, t) route 2 unités de flot et (s, u, t) route une unité de flot. Numériquement, pour $H = 3$, la résolution de la relaxation linéaire (RL_3) sans les contraintes de variable ordering (VO)

donne directement une solution entière optimale (cf. figure 3.11(b)). En revanche, avec les contraintes de variable ordering, la solution fractionnaire est de même valeur mais elle est beaucoup plus fractionnée que la solution précédente (cf. figure 3.11(c)) :

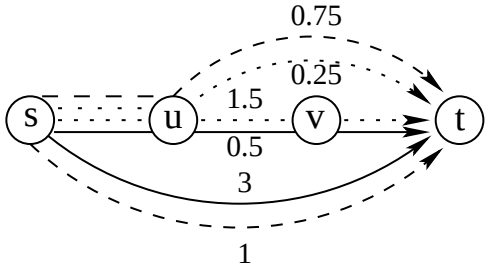
- Pour $h = 1$, le flot est décomposé en deux chemins (s, t) et (s, u, v, t) routant respectivement 3 et 0.5 unités de flot (trait plein).
- Pour $h = 2$, le flot est décomposé en deux chemins (s, t) et (s, u, t) routant respectivement 1 et 0.75 unités de flot (trait tirets).
- Pour $h = 3$, le flot est décomposé en deux chemins (s, u, t) et (s, u, v, t) routant respectivement $0, 25$ et 1.5 unités de flot (trait pointillé).



(a) Le graphe et ses capacités



(b) Sans les contraintes VO, $H = 3$



(c) Avec les contraintes VO, $H = 3$

FIG. 3.11 – Impact des contraintes de variable ordering sur la solution fractionnaire.

Cette solution fractionnaire est valide mais elle oblige a effectué des branchements pour atteindre une solution entière. La raison de ce problème se trouve évidemment dans les contraintes de variable ordering et se comprend mieux en faisant le constat suivant. En sommant les flots routés pour chaque numéro de chemin h, on constate que les flots pour $h = 2$ et $h = 3$ routent chacun un total de 1.75 unités de flot. Ainsi, la saturation des contraintes de variable ordering est à l'origine du fractionnement de la solution.

Ce problème n'a malheureusement pas de solution immédiate. Dans de futurs travaux, nous pourrions étudier ce problème plus en détail. Par exemple, Sherali et Smith [SS01] proposent de résoudre la relaxation linéaire de la racine de l'arbre de décision sans les contraintes de variable ordering puis de les ajouter dans les relaxations linéaires suivantes. Sans résoudre totalement le problème, cette stratégie pourrait s'avérer intéressante dans certains cas.

3.5 Résultats numériques

Pour comparer l'efficacité des différentes stratégies que nous proposons, nous avons utilisé deux types d'instances. Un premier ensemble a été généré par le générateur de « *Transit Grid* » développé par G. Waissi[1]. La topologie de ces instances (cf. figure 3.12) a été conçue pour tester l'efficacité d'algorithmes pour le problème du flot maximal. Cette structure en forme de grille induit une multitude de chemins élémentaires reliant la source à la destination, ce qui assure une certaine complexité à notre problème. Le dimensionnement des instances est effectué aléatoirement en choisissant les capacités des arcs dans l'intervalle $[1; 1000]$. Pour juger le passage à l'échelle de nos algorithmes, nous avons utilisé des graphes de 10 à 100 nœuds. Pour chaque taille de graphes, 10 instances ont été générées. Elles permettent d'analyser l'influence de la topologie et du dimensionnement du réseau sur les résultats.

D'autre part, nous avons utilisé le générateur aléatoire de graphe développé par B. Bachelet[2]. Pour ces instances, la topologie et le dimensionnement sont générés aléatoirement. Les tailles des graphes utilisés se répartissent de 5 à 20 nœuds.

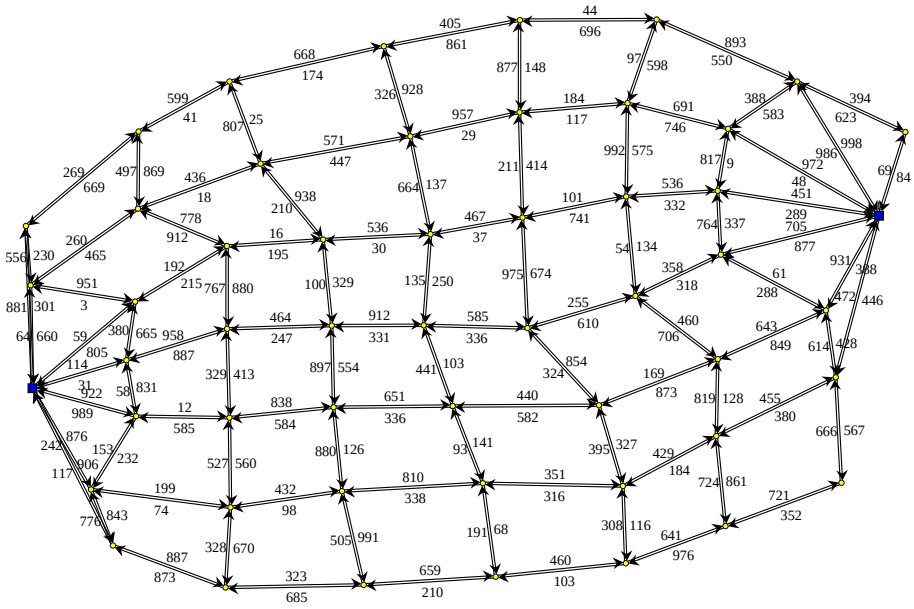


FIG. 3.12 – *Transit Grid* avec 52 nœuds et 198 arcs.

L'éclectisme des résultats obtenus perturbe l'interprétation des calculs statistiques que nous pouvons effectuer. On peut effectivement se poser la question de la pertinence d'une moyenne lorsque l'écart-type est très important. C'est pourquoi nous avons préféré présenter dans un premier temps les résultats obtenus pour un ensemble d'instances représentatives des différents cas de figure et problèmes rencontrés. Ensuite, nous discuterons quelques résultats statistiques sur l'ensemble des instances présentées.

Pour chaque table (table 3.1 à table 3.4) la colonne « graphe » indique le nom de l'instance ainsi que les nombres de nœuds et d'arcs. Le nombre maximum H de chemins est rapporté dans la deuxième

[1] `http://www.informatik.uni-trier.de/~naeher/Professur/research/`
[2] `http://www.nawouak.net/?doc=bpp_library+ch=build_graph+lang=en`

colonne. La colonne z^* donne la valeur optimale si elle a été calculée par au moins un de nos algorithmes dans le temps limite imposé (une heure). Sinon, le signe "$\geqslant$" indique qu'il ne s'agit que d'une borne inférieure (meilleure solution entière trouvée). En comparaison, nous rapportons dans la colonne « BKS » la valeur de la solution initiale calculée par l'algorithme de Baier, Kölher et Skutella [BKS05]. Comme le temps CPU pour cet algorithme est marginal par rapport aux différentes stratégies étudiées, il n'est pas présenté.

Dans les tables 3.1 à 3.4, nous comparons les temps CPU obtenu pour les stratégies suivantes (le symbole "-" indique que l'exécution a été arrêtée au bout d'une heure) :

C : le modèle arc-sommet ($KMFP_2$) est résolu par le solveur CPLEX 8.0,

BB : le modèle arc-sommet ($KMFP_2$) est résolu par Branch and Bound en utilisant la règle de branchement de Barnhart *et al.* [BHV00],

BP : le modèle arc-chemin ($KMFP_3$) est résolu par Branch and Price en utilisant la règle de branchement de Barnhart *et al.* et une gestion locale du pool de colonnes,

BP-P : similaire à BP mais avec une gestion globale du pool,

BP-VP : similaire à BP-P mais en ajoutant les contraintes de variable ordering,

BP-VP2 : similaire à BP-VP mais en utilisant également la règle de branchement alternative,

Les deux dernières colonnes comparent le gap à la racine de l'arbre de décision entre le modèle arc-sommet ($KMFP_2$) et le modèle arc-chemin ($KMFP_3$). Ce gap représente l'écart relatif entre la solution optimale fractionnaire et la solution optimale entière du problème. Ainsi, plus le gap est faible, plus il est probable que l'exploration de l'arbre de décision soit courte. Cependant, pour le calculer, il est nécessaire de disposer de la valeur de la solution optimale. Le symbole "-" indique alors que le gap correspondant n'a pas pu être calculé.

Dans un premier temps, nous constatons que le gap pour le modèle arc-chemin est relativement faible, généralement de l'ordre de quelques pour cents avec un maximum à 11.98%. Au contrainte, le gap pour le modèle arc-sommet vaut très souvent plusieurs dizaines de pour cents. La principale explication vient du problème avec les cycles de support de flot illustré sur la figure 2.6. En relâchant les variables 0-1, la solution fractionnaire peut avoir des cycles décrit par les variables de support de flot y_a^h. Ces cycles aident artificiellement à augmenter la quantité de flot qui peut être routé. Le modèle arc-chemin est obtenu par décomposition de Dantzig-Wolfe et en supprimant volontairement les variables relatives aux cycles. Le modèle ($KMFP_3$) ne porte alors plus que sur les chemins élémentaires. La solution fractionnaire ne peut alors pas créer de tels cycles de support de flot. La différence au niveau des gaps entre les deux modèles explique en grande partie pourquoi les approches par Branch and Price semblent dominées la résolution par CPLEX et par Branch and Bound au niveau du temps CPU.

On observe également que les différentes améliorations proposées – gestion globale du pool de colonnes, contraintes de variable ordering – semblent être relativement efficace en pratique sauf dans certains cas particulier. Notamment, lorsque la limite sur le nombre de chemin est suffisante pour router tout le flot maximal, les algorithmes embarquant les contraintes de variable ordering sont souvent en retrait au niveau du temps CPU. D'autre part, on relève pour la plupart des instances et des stratégies une augmentation du temps CPU en fonction du nombre maximum H de chemins, puis une brusque chute lorsque H est tel que la solution optimale est égale au flot maximal.

instance		valeurs		temps CPU (s)						gap (%)	
graphe	H	z^*	BKS	C	BB	BP	BP-P	BP-VP	BP-VP2	$KMFP_2$	$KMFP_3$
tg10-2	1	389.00	389.00	0.01	0.01	**0.00**	**0.00**	**0.00**	**0.00**	25.13	0.00
12-38	2	557.00	557.00	0.18	0.93	0.21	0.11	0.06	**0.02**	26.59	11.98
	3	716.00	557.00	1.44	104.74	9.65	4.27	0.21	**0.02**	12.15	6.18
	4	815.00	716.00	0.03	7.48	**0.02**	0.21	0.06	0.07	0.00	0.00
	∞	815.00									
tg10-3	1	189.00	189.00	0.02	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	52.13	0.00
12-38	2	350.00	350.00	0.51	0.25	**0.01**	**0.01**	**0.01**	0.02	39.66	3.41
	3	466.00	442.00	3.04	21.82	0.40	0.09	0.05	**0.04**	19.66	6.92
	4	558.00	558.00	70.97	586.99	2.98	0.69	0.12	**0.07**	3.79	1.95
	5	580.00	558.00	0.11	0.04	**0.00**	**0.00**	0.01	0.04	0.00	0.00
	∞	580.00									
tg10-9	1	501.00	501.00	0.03	0.01	**0.00**	**0.00**	**0.00**	**0.00**	32.58	0.00
12-38	2	935.00	935.00	0.23	0.11	**0.00**	**0.00**	**0.00**	**0.00**	32.77	0.00
	3	1173.00	1051.00	1.95	9.08	0.51	0.15	0.06	**0.05**	22.68	1.84
	4	1356.00	1289.00	13.09	930.27	21.99	11.51	**0.23**	3.46	10.61	4.09
	5	1460.00	1364.00	288.64	-	-	1419.06	11.29	**0.54**	-	2.78
	6	1517.00	1364.00	5.52	82.11	3.57	**0.09**	5.88	0.68	0.00	0.00
	∞	1517.00									
tg20-2	1	385.00	385.00	0.02	0.13	**0.00**	**0.00**	**0.00**	**0.00**	18.43	0.00
22-72	2	643.00	643.00	0.38	112.14	**0.01**	**0.01**	**0.01**	**0.01**	24.62	0.00
	3	832.00	643.00	1.94	-	0.05	**0.01**	**0.01**	0.05	-	0.00
	4	853.00	832.00	0.38	2.43	1.26	**0.01**	104.60	0.22	0.00	0.00
	∞	853.00									
tg40-1	1	517.00	517.00	1.16	0.05	**0.01**	**0.01**	**0.01**	**0.01**	21.31	0.00
42-152	2	750.00	520.00	788.84	-	0.04	0.02	**0.01**	0.04	-	0.06
	3	908.00	750.00	-	-	2066.70	-	303.87	**20.18**	-	2.59
	4	994.00	918.00	-	-	-	-	-	**90.29**	-	1.00
	5	1004.00	918.00	28.09	-	0.17	**0.01**	183.20	-	-	0.00
	∞	1004.00									
tg40-5	1	487.00	487.00	0.23	0.07	**0.00**	0.01	0.01	0.01	22.70	0.00
42-152	2	828.00	828.00	2.87	-	23.29	17.87	4.19	**0.14**	-	4.94
	3	1062.00	828.00	-	-	-	-	137.41	**0.36**	-	0.28
	4	1078.00	1062.00	11.88	271.55	**0.02**	**0.02**	100.26	0.98	0.00	0.00
	∞	1078.00									
tg40-8	1	454.00	454.00	0.31	0.52	0.01	0.01	**0.00**	0.01	27.93	0.00
42-152	2	775.00	705.00	16.42	-	0.05	**0.01**	0.02	0.03	-	0.00
	3	991.00	858.00	-	-	1138.19	2148.15	332.33	**0.85**	-	2.56
	4	1085.00	1067.00	32.26	-	-	1739.14	**17.72**	-	-	0.00
	∞	1085.00									
tg40-10	1	142.00	142.00	1.71	26.32	**0.00**	**0.00**	**0.00**	0.01	72.28	0.00
42-152	2	278.00	278.00	-	-	**0.01**	**0.01**	**0.01**	0.02	-	0.00
	3	410.00	410.00	-	-	**0.01**	**0.01**	**0.01**	0.02	-	0.00
	4	509.00	509.00	-	-	**0.01**	**0.01**	**0.01**	0.03	-	0.00
	5	602.00	553.00	-	-	0.14	**0.01**	0.88	0.44	-	0.00
	6	691.00	642.00	-	-	0.06	**0.01**	1.43	0.91	-	0.00
	7	769.00	720.00	-	-	0.17	**0.01**	0.63	2.07	-	0.00
	8	804.00	769.00	22.75	-	0.29	**0.09**	-	9.21	-	0.00
	∞	804.00									

TAB. 3.1 – Temps CPU pour de petites instances de type transit grid.

Dans la table 3.1, on constate que la stratégie BP-P obtient les meilleurs temps pour les instances "tg20-2" et "tg40-10" quelque soit la valeur de H. Ces deux instances sont également celles dont le gap du modèle ($KMFP_3$) est toujours nul quelque soit H, ce qui révèle sûrement une topologie particulière. Un gap nul signifie que les solutions optimales fractionnées et entière ont la même valeur. On retrouve donc le problème de saturation des contraintes de variable ordering évoqué à la section 3.4.3. Ces contraintes ont tendance à donner des solutions relâchées plus fractionnaires, obligeant l'algorithme a effectué plus de branchements. Ce problème soulève un manque de robustesse de la part des stratégies utilisant ces contraintes de variable ordering. Cependant, on peut imaginer que ces cas sont plutôt rares en pratique.

Pour les instances de la table 3.1, il semble que le branchement alternatif (BP-VP2) fournisse de bons résultats. Il en résulte souvent le meilleur temps ou un temps du même ordre que pour le branchement classique seul (BP-VP). Parfois le temps CPU correspondant est bien plus faible que celui de l'algorithme BP-VP : "tg40-1" ($H = 4$), "tg40-5" ($H = 3$), "tg40-8" ($H = 3$).

instance		valeurs		temps CPU (s)						gap (%)	
graphe	H	z^*	BKS	C	BB	BP	BP-P	BP-VP	BP-VP2	$KMFP_2$	$KMFP_3$
tg50-2	1	336.00	336.00	12.39	0.56	0.01	**0.00**	0.01	0.01	63.77	0.00
52-198	2	652.00	652.00	-	-	1.78	0.26	0.36	**0.20**	-	1.69
	3	900.00	900.00	-	-	-	-	641.34	**4.52**	-	6.44
	4	1147.00	1134.00	-	-	-	-	818.97	**28.70**	-	4.09
	5	1342.00	1329.00	-	-	-	-	-	**138.51**	-	2.47
	6	$\geqslant$1394.00	1329.00	-	-	-	-	-	-	-	-
	∞	1719.00									
tg50-5	1	562.00	562.00	91.08	0.20	**0.00**	0.01	0.01	0.01	33.84	0.00
52-198	2	965.00	902.00	-	-	0.09	**0.02**	0.04	0.26	-	2.08
	3	1343.00	1087.00	-	-	1.23	0.21	**0.10**	0.80	-	1.85
	4	1596.00	1165.00	-	-	-	-	**83.80**	-	-	6.53
	5	$\geqslant$1781.00	1480.00	-	-	-	-	-	-	-	-
	∞	2507.00									
tg50-10	1	399.00	399.00	0.07	0.01	**0.00**	**0.00**	**0.00**	0.01	28.88	0.00
52-198	2	734.00	734.00	1.50	-	**0.01**	**0.01**	**0.01**	0.02	-	0.00
	3	899.00	734.00	219.69	-	**0.01**	**0.01**	**0.01**	0.09	-	0.00
	4	1031.00	734.00	-	-	**0.04**	**0.04**	0.13	0.18	-	0.00
	5	1153.00	899.00	-	-	0.45	**0.08**	0.18	1.51	-	0.00
	6	1270.00	899.00	-	-	1.34	**0.09**	0.62	3.62	-	0.00
	7	1376.00	1031.00	-	-	0.85	**0.06**	19.86	9.57	-	0.00
	8	1403.00	1153.00	-	-	4.57	**0.61**	452.23	35.66	-	0.00
	9	1430.00	1267.50	63.00	-	18.55	**0.54**	1922.68	23.48	-	0.00
	∞	1430.00									
tg60-3	1	776.00	776.00	0.24	0.03	**0.00**	0.01	0.01	0.01	4.53	0.00
62-242	2	1168.00	986.00	-	-	0.24	0.13	**0.06**	0.43	-	2.86
	3	1480.00	1216.00	-	-	584.87	106.12	**1.49**	3.68	-	3.06
	4	$\geqslant$1681.00	1528.00	-	-	-	-	-	-	-	-
	∞	2739.00									
tg60-6	1	347.00	347.00	-	1.81	0.01	0.01	**0.00**	0.01	52.55	0.00
62-242	2	676.00	676.00	-	-	**0.01**	**0.01**	**0.01**	0.02	-	0.00
	3	983.00	983.00	-	-	**0.01**	**0.01**	**0.01**	0.04	-	0.00
	4	1251.00	1208.00	-	-	40.00	1.66	2.34	**1.59**	-	0.00
	5	1510.00	1476.00	-	-	-	-	**15.10**	103.28	-	0.00
	6	$\geqslant$1512.00	1476.00	-	-	-	-	-	-	-	-
	∞	2070.00									

TAB. 3.2 – Temps CPU pour des instances de taille moyenne de type transit grid.

Pour les instances de la table 3.2, les constatations générales semblent se confirmer. Les temps CPU ont également tendance à augmenter en fonction de la taille des instances. On retrouve des instances dont la topologie et/ou le dimensionnement assurent un gap nul pour le modèle ($KMFP_3$) : "tg50-10" et "tg60-6". On constate également que la largeur du flot maximal augmente aussi avec la taille des instances. En revanche, les stratégies BP-VP et BP-VP2 sont plus difficiles à comparer. Pour l'instance "tg50-2", l'algorithme BP-VP2 trouve l'optimum dans des temps beaucoup plus rapides que l'algorithme BP-VP et pour l'instance "tg50-5", c'est l'inverse.

Enfin, pour les instances de la table 3.3, le problème est de plus en plus dur à résoudre et peu d'algorithmes arrivent encore à trouver l'optimum malgré un nombre maximum de chemins relativement faible ($H = 4$). La règle de branchement alternative donne ici de bons résultats et le gap du modèle ($KMFP_3$) est toujours aussi faible.

instance		valeurs		temps CPU (s)						gap (%)	
graphe	H	z^*	BKS	C	BB	BP	BP-P	BP-VP	BP-VP2	$KMFP_2$	$KMFP_3$
tg70-8	1	515.00	515.00	4.64	3.55	**0.01**	**0.01**	**0.01**	**0.01**	6.68	0.00
72-268	2	928.00	928.00	4.87	-	**0.01**	**0.01**	0.02	0.02	-	0.00
	3	1144.00	928.00	-	-	96.04	85.65	1.46	**0.51**	-	0.13
	4	1147.00	1144.00	51.03	-	1.94	**0.07**	-	-	-	0.00
	∞	1147.00									
tg80-1	1	549.00	549.00	-	3.47	**0.01**	**0.01**	**0.01**	**0.01**	27.31	0.00
82-322	2	984.00	984.00	-	-	56.39	12.00	7.79	**0.87**	-	5.02
	3	1411.00	1321.00	-	-	-	-	451.69	**1.54**	-	3.85
	4	$\geqslant$1589.00	1589.00	-	-	-	-	-	-	-	-
	∞	2797.00									
tg80-6	1	474.00	474.00	-	9.49	**0.01**	**0.01**	**0.01**	**0.01**	50.76	0.00
82-322	2	833.00	833.00	-	-	1.00	0.23	0.13	**0.10**	-	0.45
	3	1160.00	1139.00	-	-	167.95	332.63	41.21	**18.35**	-	1.77
	4	1429.00	1235.00	-	-	-	-	2474.14	**173.52**	-	2.98
	5	$\geqslant$1656.00	1480.00	-	-	-	-	-	-	-	-
	∞	2445.00									
tg100-2	1	530.00	530.00	-	7.27	**0.01**	**0.01**	**0.01**	**0.01**	42.76	0.00
102-400	2	1007.00	1007.00	-	-	0.02	**0.01**	0.02	0.02	-	0.00
	3	1407.00	1336.00	-	-	76.85	20.41	5.98	**0.46**	-	0.14
	4	1768.00	1664.00	-	-	-	-	**56.22**	1951.96	-	0.32
	5	$\geqslant$1711.00	1711.00	-	-	-	-	-	-	-	-
	∞	3519.00									
tg100-9	1	424.00	424.00	-	680.71	**0.01**	**0.01**	**0.01**	**0.01**	49.73	0.00
102-400	2	845.00	845.00	-	-	**0.02**	**0.02**	**0.02**	**0.02**	-	0.00
	3	1234.00	1199.00	-	-	-	2060.47	600.63	**0.55**	-	0.10
	4	1600.00	1570.00	-	-	-	-	-	**26.41**	-	0.41
	5	$\geqslant$1905.00	1905.00	-	-	-	-	-	-	-	-
	∞	3271.00									

TAB. 3.3 – Temps CPU pour des instances de grande taille de type transit grid.

Pour les instances aléatoires de la table 3.4, les phénomènes décris précédemment se répètent :

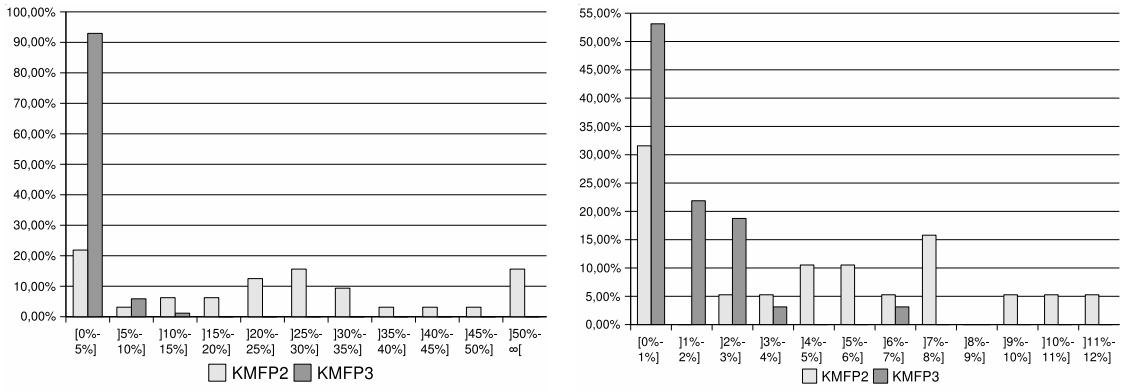
- L'instance "rand5-35" présente un gap nul pour le modèle ($KMFP_3$) quelque soit la valeur de H ce qui est fatal aux stratégies utilisant les contraintes de variable ordering.
- Pour les instances "rand10-45" et "rand15-60", c'est l'algorithme BP-VP qui se comporte le mieux, au détriment du branchement alternatif.
- L'algorithme BP-VP2 est en revanche bien plus performant sur l'instance "rand20-140".

Il est donc toujours aussi difficile de juger l'apport effectif de la règle de branchement alternative. En revanche, sur ces instances, on constate que les gaps pour le modèle ($KMFP_2$) sont moins élevés que dans le cas des transit grids. De plus, la largeur du flot maximal est également plus grande (pour une taille de graphe comparable) ce qui nous permet de tester nos algorithmes pour des valeurs de H plus importantes.

instance		valeurs		temps CPU (s)						gap (%)	
graphe	H	z^*	BKS	C	BB	BP	BP-P	BP-VP	BP-VP2	$KMFP_2$	$KMFP_3$
rand5-35	1	66.00	66.00	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	10.09	0.00
5-35	2	128.00	128.00	0.02	0.02	0.01	**0.00**	**0.00**	**0.00**	11.87	0.00
	3	182.00	182.00	0.02	0.54	**0.00**	**0.00**	**0.00**	**0.00**	9.08	0.00
	4	223.00	204.00	0.07	19.04	0.07	**0.02**	0.03	0.04	7.19	0.00
	5	262.00	243.00	0.05	221.62	0.09	**0.01**	0.05	0.08	5.59	0.00
	6	297.00	278.00	0.19	1186.40	0.22	**0.05**	0.09	0.12	4.15	0.00
	7	326.00	297.00	0.41	0.06	0.19	**0.03**	0.12	0.26	0.00	0.00
	∞	326.00									
rand10-45	1	73.00	73.00	0.01	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	2.25	0.00
10-45	2	142.00	142.00	0.08	0.05	**0.00**	**0.00**	**0.00**	**0.00**	4.05	0.58
	3	209.00	209.00	0.53	0.52	0.03	**0.01**	**0.01**	0.04	5.43	0.57
	4	260.00	260.00	1.39	12.53	0.51	**0.09**	0.12	1.63	6.87	1.58
	5	306.00	306.00	7.26	147.83	6.87	1.04	**0.28**	2.85	7.42	1.83
	6	345.00	321.00	118.78	-	202.89	29.01	**0.93**	13.79	-	2.36
	7	381.00	360.00	1285.61	-	-	-	**4.81**	59.60	-	2.54
	8	413.00	368.00	-	-	-	-	**17.59**	379.34	-	2.93
	9	429.00	381.00	-	-	-	-	**2554.28**	-	-	6.21
	10	$\geqslant$417.00	417.00	-	-	-	-	-	-	-	-
	∞	498.00									
rand15-60	1	86.00	86.00	0.01	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.00	0.00
15-60	2	163.00	163.00	0.08	0.02	**0.00**	**0.00**	**0.00**	0.01	0.61	0.00
	3	221.00	221.00	0.24	0.47	**0.00**	**0.00**	**0.00**	0.01	3.37	0.44
	4	248.00	229.00	1.97	28.51	5.43	1.26	**0.12**	0.76	7.62	2.53
	5	268.00	229.00	12.91	-	87.58	20.94	**0.92**	3.18	-	2.86
	6	287.00	229.00	72.99	-	-	2616.90	**2.07**	5.90	-	2.74
	7	295.00	229.00	1960.18	-	-	-	**46.63**	2080.02	-	3.97
	8	$\geqslant$301.00	256.00	-	-	-	-	-	-	-	-
	∞	310.00									
rand20-140	1	81.00	81.00	0.01	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.46	0.00
20-140	2	158.00	158.00	0.02	0.02	**0.00**	**0.00**	**0.00**	**0.00**	0.30	0.00
	3	228.00	228.00	0.30	0.03	**0.00**	**0.00**	0.01	0.01	0.38	0.00
	4	253.00	235.00	14.90	-	91.92	323.10	7.48	**1.52**	-	1.81
	5	274.00	235.00	230.60	-	-	-	-	**4.80**	-	1.86
	6	294.00	236.00	-	-	-	-	-	**5.81**	-	1.78
	7	311.00	236.00	-	-	-	-	-	**19.25**	-	1.81
	8	319.00	236.00	-	-	-	-	-	**2042.98**	-	1.60
	9	$\geqslant$325.00	261.00	-	-	-	-	-	-	-	-
	10	327.00	261.00	725.65	-	49.66	**6.16**	-	-	-	0.00
	∞	327.00									

TAB. 3.4 – CPU times for random digraphs.

Pour mettre en valeur la différence entre les gaps des modèles ($KMFP_2$) et ($KMFP_3$), la figure 3.13 présentent leur distribution sous la forme d'un histogramme. Pour chaque modèle et pour chaque intervalle considéré, nous rapportons le poucentage d'instances dont le gap du modèle est compris dans cet intervalle. Ainsi, nous constatons le phénomène évoqué précédemment : pour le modèle arc-chemin, le gap est généralement très faible alors que pour le modèle arc-sommet, il est plus réparti avec des valeurs qui peuvent être très importantes, notamment pour les transit grids. Pour les instances aléatoires, les gaps sont plus faibles mais le phénomène se confirme.

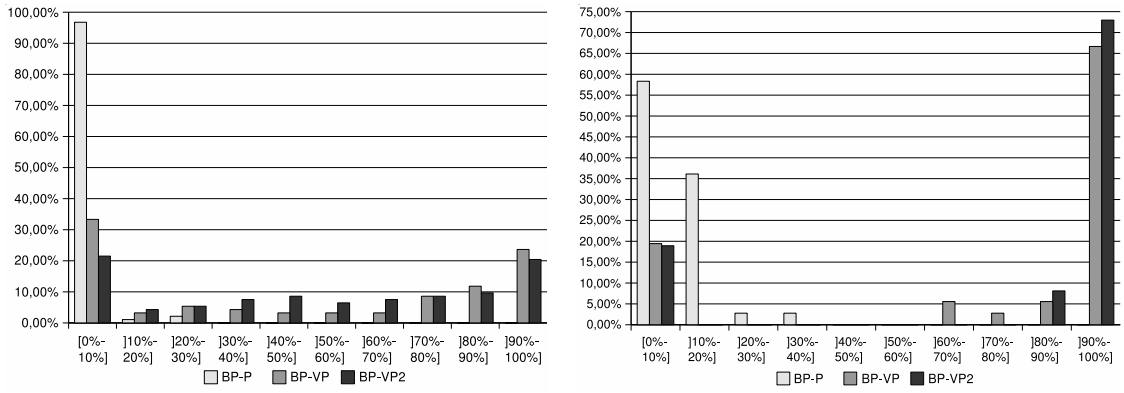


(a) transit grids (b) instances aléatoires

FIG. 3.13 – Distribution des gaps pour les modèles ($KMFP_2$) et ($KMFP_3$).

Un autre cas de figure que nous évoquions dans l'analyse des résultats est le problème engendré par la saturation des contraintes de variable ordering. Cette saturation induit des solutions plus fractionnées de la relaxation linéaire. Des branchements sont alors nécessaires pour trouver une solution entière de même valeur. Pour confirmer cette supposition, nous souhaiterions compter pour chaque instance le pourcentage de relaxations linéaires aboutissant à une saturation de ces contraintes. Mais pour comparer avec l'algorithme BP-P qui n'utilisent pas ces contraintes, nous avons préféré compter le pourcentage de solutions fractionnaires dont au moins deux flots de numéros h successifs routent la même quantité de flot. Ce pourcentage est représentatif du caractère fractionnaire des solutions de la relaxation linéaire et donc de la saturation des contraintes de variable ordering pour les stratégies BP-VP et BP-VP2.

La figure 3.14 représente la distribution de ce critère pour les algorithmes BP-P, BP-VP et BP-VP2. Pour chaque intervalle, nous rapportons le pourcentage d'instances pour lesquelles le critère choisi se situe dans cet intervalle.



(a) transit grids (b) instances aléatoires

FIG. 3.14 – Saturation des contraintes de variable ordering.

Ainsi, nous constatons que ce critère est très faible pour l'algorithme BP-P. Puis, en ajoutant sim-

plement les contraintes de variable ordering, ce critère devient beaucoup plus élevé. Le branchement alternatif ne modifie pas ce phénomène qui est donc bien inhérent à ces contraintes. Ceci explique en grande partie pourquoi ces deux algorithmes sont parfois très lent sur des instances alors que les autres, jugés moins performant, sont plus rapides, notamment pour les instances dont le gap du modèle ($KMFP_3$) est nul, c'est-à-dire que les solutions fractionnaires ont la même valeur que l'optimum entier.

Dans la table 3.5, nous souhaitons comparer nos différents algorithmes suivant des critères globaux sur l'ensemble des instances. Ainsi, la moyenne des temps CPU est un premier indicateur de la qualité d'un algorithme par rapport aux autres. Mais il reflète mal la distribution des temps CPU qui peuvent être très éloignés de cette moyenne, comme le confirme l'écart-type. C'est pourquoi nous avons également choisi de présenter le pourcentage d'instances pour lequel un algorithme donné à obtenu le meilleur temps (seul ou dans le même temps que d'autres algorithmes). Puis, en cas d'« échec », c'est-à-dire si l'algorithme ne réalise pas le meilleur temps, il est important de pouvoir comparer le temps CPU réalisé au meilleur temps. Comme les écarts peuvent être extrêmement important, l'écart relatif donne des valeurs difficilement exploitables. C'est pourquoi nous avons préféré calculer le ratio entre l'écart absolue (différence entre le temps CPU de l'algorithme considéré et le meilleur temps) et le temps CPU de l'algorithme considéré. Ainsi, ce ratio est compris entre 0 et 1. Il est proche de 0 si les deux temps sont également très proches et il est proche de 1 si, au contraire, le meilleur temps est négligeable par rapport au temps CPU de l'algorithme considéré.

De plus, il est intéressant de considérer ces critères uniquement dans les cas où le gap du modèle ($KMFP_3$) est strictement positif ("[gap > 0]"). Ainsi, on met de coté les cas patologiques où le gap est nul et les cas trop difficiles à résoudre où aucun algorithme n'a convergé dans le temps imparti (une heure), ne permettant donc pas de calculer le gap.

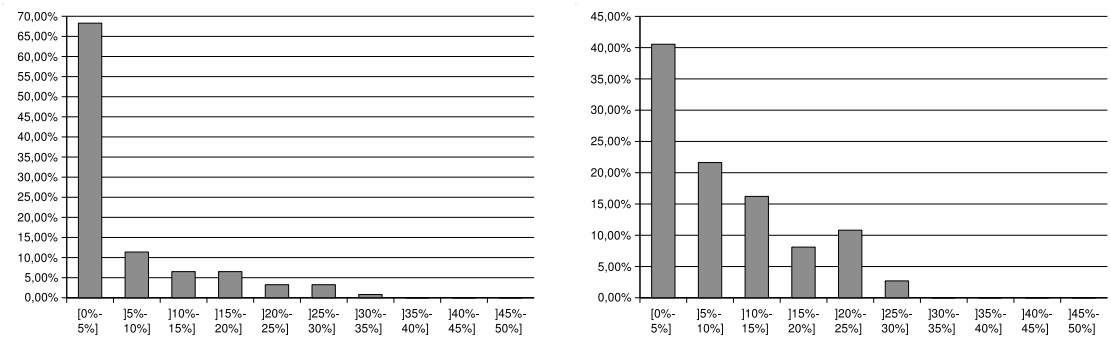
	Critère	C	BB	BP	BP-P	BP-VP	BP-VP2
transit grid	Moyenne temps CPU (s)	2443.44	2686.70	1557.11	1528.15	1330.11	**1250.91**
	Écart-type temps CPU (s)	1674.53	1550.46	1768.71	1752.73	1697.70	1707.59
	Moyenne temps CPU [gap > 0] (s)	2544.59	2886.49	1437.89	1388.49	508.34	**183.86**
	Pourcentage de meilleur temps (%)	30.89	31.71	56.10	**67.48**	59.35	61.79
	Pourcentage de meilleur temps [gap > 0] (%)	0.00	0.00	3.03	6.06	24.24	**72.73**
	Écart moyen en cas d'échec	0.95	0.95	0.87	0.75	0.71	**0.68**
	Écart moyen en cas d'échec [gap > 0]	1.00	1.00	0.97	0.94	0.75	**0.28**
instances aléatoires	Moyenne temps CPU (s)	1092.82	1795.07	1374.20	1345.91	1044.20	**806.00**
	Écart-type temps CPU (s)	1594.33	1791.46	1760.90	1735.33	1631.15	1446.57
	Moyenne temps CPU [gap > 0] (s)	1205.97	2410.55	1821.96	1766.24	946.40	**456.75**
	Pourcentage de meilleur temps (%)	3.13	12.50	31.25	56.25	**62.50**	40.63
	Pourcentage de meilleur temps [gap > 0] (%)	0.00	0.00	11.11	22.22	**66.67**	33.33
	Écart moyen en cas d'échec	0.66	0.73	0.67	0.64	**0.46**	0.53
	Écart moyen en cas d'échec [gap > 0]	1.00	1.00	0.89	0.78	**0.33**	0.67

TAB. 3.5 – Comparaison des différentes stratégies.

Pour les instances de type transit grid, on observe une progression au fil des améliorations successives et une domination de l'algorithme BP-VP2. Cependant, on peut remarquer que c'est l'algorithme BP-P qui obtient le plus de meilleurs temps sur l'ensemble des instances. En revanche, lorsqu'on ne considère que les instances pour lesquelles le gap du modèle ($KMFP_3$) est strictement positif, l'algorithme BP-VP2 parrait beaucoup plus performant que les autres.

Pour les instances aléatoire, les temps CPU moyens semble donner la faveur au branchement alternatif. Pourtant, c'est l'algorithme BP-VP qui obtient le plus de meilleurs temps et un meilleur écart moyen en cas d'échec. Ceci reflète la difficulté pour départager ces deux algorithmes et donc de juger du réel apport du branchement alternatif.

Enfin, il est intéressant d'analyser la qualité de la solution initiale calculée par l'algorithme de Baier, Kölher et Skutella. Pour cela, nous avons calculé l'écart relatif entre la solution initiale et la solution optimale (ou la meilleur solution entière). L'histogramme de la figure 3.15 illustre la distribution de cet écart relatif. On constate alors que cet écart est généralement très faible. Ceci nous conforte dans notre choix de cet algorithme pour calculer la solution initiale.



(a) transit grids (b) instances aléatoires

FIG. 3.15 – Qualité de la solution initiale.

CHAPITRE 4

APPLICATION AU PROBLÈME DU DÉLAI DE BOUT-EN-BOUT

Ce chapitre est consacré à l'utilisation des flots k-séparables dans la gestion de la qualité de service sous forme de délai de bout-en-bout. Traditionnellement, dans les problèmes d'optimisation dans les réseaux, la qualité de service est uniquement représentée par le délai moyen sur les arcs. Cependant, le délai réellement ressenti par les utilisateurs est lié au parcours des paquets d'un « bout » à l'autre du réseau. Avec le développement des réseaux MPLS, prendre en considération la mesure des délais de bout-en-bout de chaque chemin actif est devenu un des défis majeurs des réseaux haut débit.

L'idée principale que nous développons dans cette partie est d'utiliser des flots k-séparables pour modéliser ce délai de bout-en-bout sur les chemins actifs. Nous nous proposons alors de traiter un problème de flots k-séparables dont l'objectif est de minimiser le « pire » délai de bout-en-bout. Ce problème sera mentionné sous le terme de *k-splittable Delay Constrained Routing Problem* (k-DCRP). La solution d'un tel problème propose un compromis entre des contraintes de délai qui induisent généralement l'utilisation d'un grand nombre de chemins, et la contrainte sur le nombre de chemins actifs qui induit, au contraire, un délai élevé. En comparaison, les modèles classiques de routage utilisent une mesure du délai moyen induite par une approximation convexe de la congestion moyenne parmi tous les arcs portant du flot (cf. [FGK73]). Ainsi, ces modèles ne peuvent pas donner de garanties réalistes sur la qualité de services. Cependant, nous étendrons notre approche au délai moyen sous la forme du problème *k-splittable Minimum Average Delay Problem* (k-MADP) et nous montrerons que celui-ci peut être utilisé comme une approximation pour le k-DCRP.

Outre les travaux sur les flots k-séparables évoqués dans les chapitres précédents, nous pouvons citer l'analyse dans [DM07] d'une adaptation de la méthode de déviation de flot au problème de multiflot à coût convexe incluant un mécanisme de contrôle sur le nombre de chemins actifs. Comme les contraintes de délai de bout-en-bout sont non-linéaires et induisent également des difficultés combinatoires (choix des chemins actifs), à notre connaissance, peu de modèles ont été présentés. Les travaux les plus importants dans ce domaine ont été réalisés par Ben-Ameur et Ouorou [BaO06] qui ont défini et modélisé le *Delay Constrained Routing Problem*. Ils ont montré que ce problème est NP-difficile et ils ont proposé plusieurs bornes par convexification des contraintes de délai.

Dans un premier temps, nous proposerons une définition et une formulation mathématique du (k-DCRP). Puis, nous décrirons les méthodes utilisées dans notre algorithme de résolution qui combine des méthodes d'approximation au sein d'un Branch and Price. Ensuite, nous détaillerons la résolution du problème et l'application de ces différentes méthodes. Enfin, nous discuterons les résultats numériques obtenus en comparant la minimisation du délai de bout-en-bout (k-DCRP) et du délai moyen (k-MADP).

4.1 Formulation du problème

4.1.1 Définitions

Pour ce problème, nous ne considérons qu'une seule paire origine-destination à laquelle est associée une demande d à router entre la source s et la destination t. Notons $\mathcal{P}$ l'ensemble des chemins élémentaires entre s et t. Le *Delay Constrained Routing Problem* (DCRP) consiste à router la demande de façon à minimiser le délai de bout-en-bout maximal.

Sous l'hypothèse de Kleinrock (trafic poissonien, files d'attente M/M/1 indépendantes), le délai de bout-en-bout d'un paquet peut être estimé à partir des délais moyens sur les arcs empruntés par le paquet. Cette estimation représente une moyenne sur tous les paquets parcourant le même chemin dans le réseau sous des hypothèses très fortes. Elle n'est en aucun cas une garantie sur le délai de bout-en-bout maximal subi par les paquets. Cependant, cette estimation du délai de bout-en-bout semble être acceptée dans la littérature [BKP03, BaO06]. Ainsi, en notant λ la taille moyenne d'un paquet et $x_a \geqslant 0$ la quantité de flot sur l'arc $a \in A$, le délai de bout-en-bout θ_p, pour un chemin actif $p \in \mathcal{P}$ donné, est évalué par :

$$\theta_p = \sum_{a \in p} \frac{\lambda}{u_a - x_a} \tag{4.1}$$

Il est à noter que ce délai de bout-en-bout ne doit être évalué que pour les chemins actifs, c'est-à-dire ceux qui portent réellement le flot. En effet, la formulation de ce délai peut s'appliquer à n'importe quel chemin. Il est alors possible de trouver un chemin de s à t tel que la somme des délais sur les arcs de ce chemin soit supérieure à celle sur les chemins actifs. Comme aucun paquet n'emprunte ce chemin de bout-en-bout (sinon ce serait lui aussi un chemin actif), ce délai ne doit pas être pris en compte.

Plusieurs modèles sont possibles. Plutôt que modéliser le choix des chemins actifs dans l'ensemble $\mathcal{P}$, nous avons choisi de décrire chaque chemin actif comme un flot de valeurs 0-1 en utilisant une formulation arc-sommet. Une telle approche permet de mieux contrôler les délais de bout-en-bout de chaque chemin actif. Cependant, pour limiter la taille du modèle, il convient de limiter le nombre de chemins actifs. Comme nous l'avons vu dans les chapitres précédents, cette contrainte définit la notion de flot k-séparable. Soit H le nombre maximal de chemins actifs autorisés à porter le trafic. Ainsi le *k-splittable Delay Constrained Routing Problem* (k-DCRP) consiste à router la demande de façon à minimiser le délai de bout-en-bout maximal et avec au plus H chemins.

Ce problème (k-DCRP) est NP-difficile car c'est une généralisation du problème (DCRP) présenté par Ben-Ameur et Ouorou [BaO06]. Le (DCRP) est un problème de routage sous la forme d'un problème de multiflot de coût minimal avec une contrainte de délai de bout-en-bout sur les chemins actifs. Dans

[BaO06], les auteurs proposent des résultats sur la complexité de ce problème ainsi que des bornes inférieures et supérieures obtenues par convexification de la contrainte de délai.

Le (k-DCRP) est également une généralisation du problème du flot k-séparable maximal présenté par Baier, Köhler et Skutella [BKS05].

4.1.2 Modèle arc-sommet

Nous avons donc choisi de modéliser H chemins potentiellement actifs comme un sous-problème de flot entier. Pour chaque numéro de chemin $h = 1 \dots H$, $x_a^h \geqslant 0$ représente la variable de flot sur l'arc $a \in A$, $y_a^h \in \{0,1\}$ est la variable de décision associée et $d_h \geqslant 0$ est la quantité de flot associée. Notons $\omega^-(v)$ l'ensemble des arcs entrants du cocycle de v et $\omega^+(v)$ l'ensemble des arcs sortants du cocycle. Pour simplifier les notations, notons x_a le flot agrégé sur l'arc a, c'est-à-dire $x_a = \sum_{h=1}^{H} x_a^h$. Enfin, θ représente le délai de bout-en-bout maximal. Le modèle arc-sommet s'écrit alors comme suit :

$$
(k\text{-DCRP}) \left\{
\begin{array}{l}
\min\ \theta \\
s.c. \\
\displaystyle\sum_{a\in\omega^-(v)} x_a^h - \sum_{a\in\omega^+(v)} x_a^h = \begin{cases} -d_h & \text{si } v = s, \\ d_h & \text{si } v = t, \\ 0 & \text{sinon.} \end{cases} \quad \forall h = 1\dots H,\ \forall v \in N \quad (a) \\
\displaystyle\sum_{a\in\omega^-(v)} y_a^h - \sum_{a\in\omega^+(v)} y_a^h = \begin{cases} -1 & \text{si } v = s, \\ 1 & \text{si } v = t, \\ 0 & \text{sinon.} \end{cases} \quad \forall h = 1\dots H,\ \forall v \in N \quad (b) \\
x_a^h - u_a y_a^h \leqslant 0 \qquad \forall a \in A \quad \forall h = 1\dots H \quad (c) \\
\displaystyle x_a = \sum_{h=1}^{H} x_a^h \leqslant u_a \qquad \forall a \in A \quad (d) \\
\displaystyle\sum_{h=1}^{H} d_h = d \quad (e) \\
\displaystyle\sum_{a\in\omega^-(v)} y_a^h \leqslant 1 \qquad \forall v \in V \quad \forall h = 1\dots H \quad (f) \\
\displaystyle\sum_{a\in A} \frac{\lambda y_a^h}{u_a - x_a} \leqslant \theta \qquad \forall h = 1\dots H \quad (g) \\
\\
x_a^h \geqslant 0 \qquad \forall a \in A \quad \forall h = 1\dots H \quad (h) \\
y_a^h \in \{0,1\} \qquad \forall a \in A \quad \forall h = 1\dots H \quad (i) \\
d_h \geqslant 0 \qquad \forall h = 1\dots H \quad (j)
\end{array}
\right. \tag{4.2}
$$

Comme le modèle (KMFP$_2$), ce modèle utilise deux blocs de contraintes de conservation de flot, pour les variables de flot d'une part (4.2.a), et pour les variables de support de flot d'autre part (4.2.b). Puis les contraintes (4.2.c) assurent le couplage entre le flot et son support. Les contraintes (4.2.d) imposent le respect des capacités. La contrainte (4.2.e) est la contrainte de demande. Les contraintes (4.2.f) sont nécessaires pour éliminer les cycles connectés au chemin dans le support de flot (cf. figure 2.3). Enfin, les contraintes (4.2.g) sont les contraintes de délai de bout-en-bout utilisant la formulation (4.1) du délai.

La minimisation du délai induit une dispersion du flot sur le réseau la plus importante possible. L'objectif, ici, implique donc que chacune des H routes portera une fraction positive de la demande dans la solution optimale. D'autre part, l'utilisation de variables de flot x_a^h spécifiques pour chaque numéro de chemin h est nécessaire car une formulation avec uniquement les variables de flot agrégé x_a laisse libre le flot de « sauter » d'un chemin à un autre. Ceci violerait la contrainte sur le nombre maximum de chemins autorisés et ne permettrait pas l'estimation du délai de bout-en-bout pour ces flots.

Comme indiqué précédemment, la minimisation du délai moyen sur un flot k-séparable (k-MADP) peut fournir une approximation raisonnable même si certaines contraintes de délai de bout-en-bout peuvent être violées. Ce problème se modélise comme suit

$$\text{(k-MADP)} \begin{cases} \min\ \psi(x) = \displaystyle\sum_{a \in A} \frac{x_a}{u_a - x_a} \\ s.c. \\ \qquad \text{(4.2.a), (4.2.b), (4.2.c), (4.2.d), (4.2.e), (4.2.f), (4.2.h), (4.2.i), (4.2.j)} \end{cases} \tag{4.3}$$

Ce modèle reprend la modélisation précédente du (k-DCRP). Seule la contrainte non-linéaire de délai de bout-en-bout est absente du modèle du (k-MADP).

Ces deux modèles utilisent donc à la fois des variables discrètes et des contraintes non linéaire. Leur résolution nécessite la combinaison de plusieurs méthodes que nous détaillons dans la section suivante.

4.2 Présentation des méthodes utilisées

Dans la section suivante, nous décrivons comment résoudre le problème (k-DCRP) en utilisant un algorithme de type Branch and Price. Le sous-problème est résolu par un schéma d'approximation similaire aux idées proposées par Shahrokhi et Matula [SM90] et développées plus tard par Bienstock [Bie02]. Le cœur de ce schéma d'approximation est un problème de routage résolu par l'algorithme de déviation de flot. Enfin, dans notre cas, l'application de l'algorithme de déviation de flot nécessite la résolution d'un programme linéaire correspondant à la relaxation linéaire du problème du flot k-séparable de coût minimal, ce qui sera fait par génération de colonnes.

Avant de détailler l'enchaînement de ces différentes méthodes, nous proposons ici une description de chacune d'entre elles.

4.2.1 Algorithme de déviation de flot

L'algorithme de Frank–Wolfe est une méthode de résolution de problèmes d'optimisation non linéaire sur un ensemble convexe. Considérons une fonction $f : \mathbb{R}^n \mapsto \mathbb{R}$ continûment différentiable à minimiser sur un ensemble $\mathcal{X}$ convexe fermé de $\mathbb{R}^n$. Le problème générique (P) s'écrit :

$$(\text{P}) \begin{cases} \min\ f(x) \\ s.c. \\ \qquad x \in \mathcal{X} \end{cases} \tag{4.4}$$

Le principe de cet algorithme est celui d'une méthode de descente dont la direction de descente est déterminée par la résolution d'une linéarisation du problème (P) autour du point courant. Ainsi, à l'itération i, on considère le problème (P) en remplaçant la fonction f par son approximation de Taylor au premier ordre autour de la solution courante x^i. En supprimant les termes constants de la fonction objectif, le sous-problème s'exprime suivant le modèle (FW^i).

$$(\text{FW}^i) \begin{cases} \min\ \nabla f(x^i)^\top y \\ s.c. \\ \qquad y \in \mathcal{X} \end{cases} \tag{4.5}$$

Le sous-problème (FW^i) est un problème convexe. Notons y^i une solution optimale. La direction $d^i = y^i - x^i$ est une direction de descente et tous les points $x^i + \alpha d^i$ pour $\alpha \in [0, 1]$ sont réalisables car $\mathcal{X}$ est convexe. Le point suivant x^{i+1} est alors obtenu par déplacement optimal dans la direction définie par d^i (recherche linéaire).

La méthode de Frank-Wolfe est particulièrement adaptée lorsque l'ensemble $\mathcal{X}$ est un polyèdre. Le sous-problème (FW^i) est alors un programme linéaire dont la solution est un des sommets du polyèdre. Toutefois, la solution courante a tendance à zigzaguer à l'approche de la solution optimale. En effet, l'angle entre deux directions de descente successives peut tendre vers 180°si la solution est à l'intérieur d'une face du polyèdre. La figure 4.1 illustre le comportement typique de la méthode de Frank-Wolfe dans ce cas.

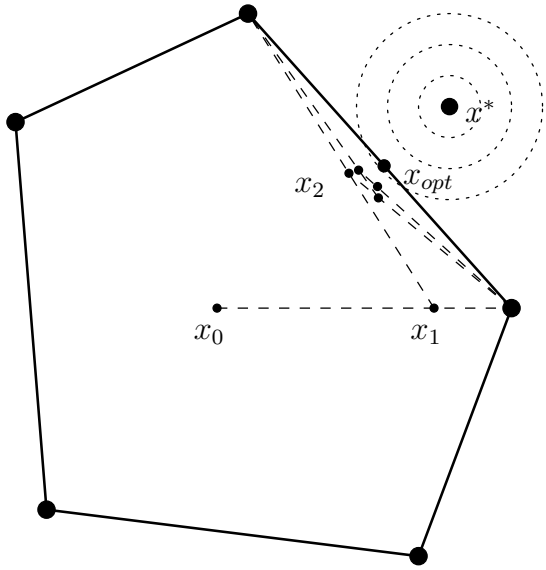


FIG. 4.1 – Illustration de la méthode de Frank-Wolfe.

La méthode de déviation de flot (*Flow Deviation*, [FGK73]) est une application de l'algorithme de Frank-Wolfe aux problèmes de routage. Elle permet de résoudre des problèmes de multiflot dont l'objectif est de minimiser une fonction non linéaire. En particulier, dans [FGK73], les auteurs considèrent un problème de multiflot dont l'objectif à minimiser est le délai moyen Φ modélisé sous la forme suivante :

$$\Phi(x) = \sum_{a \in A} \varphi_a(x_a) \tag{4.6}$$

$$\varphi_a(x_a) = \frac{x_a}{u_a - x_a} \tag{4.7}$$

où $x = \{x_a^k \ \forall a \in A \ \forall k \in \mathcal{K}\}$ est le multiflot et $x_a = \sum_{k \in \mathcal{K}} x_a^k$ l'agrégation sur l'arc a. La fonction Φ joue donc le rôle de fonction barrière rendant les contraintes de capacité inutiles. À l'itération i, le sous-problème défini par la linéarisation de Φ s'écrit :

$$\text{(FD}^i\text{)} \begin{cases} \min \ \nabla\Phi(x^i)^\top y = \displaystyle\sum_{k \in \mathcal{K}} \sum_{a \in A} \frac{\partial \Phi}{\partial x_a^k}(x^i) y_a^k \\ s.c. \\ \displaystyle\sum_{a \in \omega^-(v)} y_a^k - \sum_{a \in \omega^+(v)} y_a^k = \begin{cases} -d^k & \text{si } v = s_k, \\ d^k & \text{si } v = t_k, \\ 0 & \text{sinon.} \end{cases} \quad \forall v \in N \ \forall k \in \mathcal{K} \quad (a) \end{cases} \tag{4.8}$$

Ce programme linéaire ne contient plus que les contraintes de conservation de flot (4.8.a). Il s'agit donc d'un problème de multiflot non contraint de coût minimal. De plus, on constate qu'il est séparable par rapport aux paires origine-destination et que les sous-problèmes se réduisent à des problèmes de plus courts chemins dont les longueurs sont données par :

$$\frac{\partial \Phi}{\partial x_a^k}(x^i) = \varphi'(x_a^i) \qquad \forall a \in A \quad \forall k \in \mathcal{K} \tag{4.9}$$

On « dévie » ainsi à chaque itération, pour toutes les paires origine-destination, une même proportion des flots circulant sur des chemins non optimaux vers les chemins de longueur minimale par rapport aux dérivées premières des fonctions φ_a. Par définition des dérivées premières, ce re-routage d'une partie des flots assure à la fois la conservation des demandes et, au moins localement, la décroissance de la fonction Φ. Cette déviation définit une direction de descente le long de laquelle une méthode de recherche linéaire permet de calculer le déplacement optimal. L'algorithme converge alors vers un minimum global du problème initial.

Notons qu'il est possible de conserver les contraintes de capacités dans le programme non linéaire même si elles sont redondantes avec la fonction barrière Φ. Dans ce cas, les sous-problèmes ne se réduisent plus à de simples problèmes de plus courts chemins mais à des problèmes de flot de coût minimal. La direction de descente n'est plus définie par un seul chemin mais par un flot réalisable vis-à-vis des

capacités. Le flot n'est donc plus dévié vers un unique chemin. L'avantage de cette solution est que la quantité de flot déviée à chaque itération est plus importante. C'est la solution retenue par Bienstock et Raskina [BR02].

4.2.2 Algorithme de Shahrokhi et Matula

Shahrokhi et Matula [SM90] ont proposé un algorithme d'approximation polynomial pour le problème de multiflot concurrent maximal avec capacités uniformes. Ce problème consiste à maximiser un facteur de charge qui multiplie la demande de chaque paire origine-destination jusqu'à saturation des capacités. Même si ce problème se modélise par un programme linéaire, il a curieusement la réputation d'être numériquement très difficile. C'est la raison pour laquelle il a suscité un important courant de recherche vers des algorithmes d'approximation.

Notons $\mathcal{P}_k$ l'ensemble des chemins élémentaires reliant la source s_k à la destination t_k, pour toute paire origine-destination $k \in \mathcal{K}$. La variable x_p^k représente la quantité de flot routée sur le chemin $p \in \mathcal{P}_k$ pour la paire origine-destination $k \in \mathcal{K}$. La contrainte des demandes (4.10.a) assure qu'une même proportion θ de la demande d_k de chaque paire origine-destination $k \in \mathcal{K}$ est routée. Ce facteur de charge θ est donc l'objectif à maximiser. Enfin, la contrainte de capacité (4.10.b) introduit la quantité de flot agrégée x_a sur l'arc $a \in A$. Il est à noter que, les capacités étant uniformes, nous pouvons les fixer à 1 sans perte de généralité. Le problème de multiflot concurrent maximal avec capacités uniforme se modélise alors par la formulation arc-chemin (MAXTHRU).

$$\text{(MAXTHRU)} \begin{cases} \max\ \theta & & & \\ s.c. & & & \\ \quad \displaystyle\sum_{p \in \mathcal{P}_k} x_p^k = \theta d_k & \forall k \in \mathcal{K} & (a) \\ \quad x_a = \displaystyle\sum_{k \in \mathcal{K}} \sum_{p \in \mathcal{P}_k} \delta_a^p x_p^k \leqslant 1 & \forall a \in A & (b) \\ \quad x_p^k \geqslant 0 & \forall k \in \mathcal{K}\ \forall p \in \mathcal{P}_k & (c) \\ \quad \theta \geqslant 0 & & (d) \end{cases} \tag{4.10}$$

Les résultats présentés par Shahrokhi et Matula ont été enrichis par la suite de nombreuses améliorations et extensions à des problèmes proches comme les problèmes dits de « fractional packing » [PSvT95] et les programmes convexes bloc-angulaires [GK94]. Le point commun entre ces approches est la substitution de la fonction objectif de type min-max par une fonction potentiel séparable sur les arcs du réseau. La non-séparabilité sur les arcs de ces fonctions objectif de type min-max est une des explications des difficultés numériques rencontrées par les méthodes de résolution classiques car elle ne favorise pas l'adaptation de techniques de décomposition ou de génération de chemins.

Ainsi le problème de type min-max est approché par la minimisation d'une fonction de pénalité convexe sur un ensemble convexe et le principe de résolution peut être vu comme un algorithme de Frank-Wolfe. L'étape centrale de l'algorithme de Shahrokhi et Matula est le re-routage de certaines paires origine-destination, ne satisfaisant pas les conditions d'optimalité, vers les plus courts chemins calculés avec des longueurs fonctions exponentielles de la charge des arcs.

En effet, le problème de multiflot concurrent maximal avec capacités uniformes est directement lié au problème de minimisation de la congestion maximale qui consiste à minimiser la charge relative maximale sur un arc x_a/u_a. Dans le cas des capacités uniformes, le problème se modélise par le programme linéaire (MINCONG).

$$\text{(MINCONG)} \begin{cases} \min\ \lambda & & & \\ s.c. & & & \\ \quad \displaystyle\sum_{p\in\mathcal{P}_k} x_p^k = d_k & & \forall k \in \mathcal{K} & (a) \\ \quad x_a = \displaystyle\sum_{k\in\mathcal{K}} \sum_{p\in\mathcal{P}_k} \delta_a^p x_p^k \leqslant \lambda & & \forall a \in A & (b) \\ \quad x_p^k \geqslant 0 & & \forall k \in \mathcal{K}\ \forall p \in \mathcal{P}_k & (c) \\ \quad \lambda \geqslant 0 & & & (d) \end{cases} \tag{4.11}$$

On vérifie immédiatement qu'à toute solution réalisable x de (MAXTHRU), dont la congestion maximale vaut 1 (ce qui est forcément le cas de la solution optimale), correspond une solution réalisable $\tilde{x} = \frac{1}{\theta}x$ de (MINCONG) dont la congestion maximale est $\tilde{\lambda} = 1/\theta$. Par conséquent, maximiser θ équivaut à minimiser λ et les chemins supportant les solutions optimales de ces deux problèmes sont les mêmes. Ainsi, Shahrokhi et Matula choisissent la fonction potentiel suivante :

$$\Phi(x) = \sum_{a\in A} e^{\frac{m^2}{\varepsilon}x_a} \tag{4.12}$$

La convergence de l'algorithme est liée à la décroissance suffisante de la fonction Φ à chaque itération. À la suite de ce travail pionnier de Shahrokhi et Matula, de complexité $O(nm^7/\varepsilon^5)$, un certain nombre d'auteurs ont utilisé une fonction potentiel exponentielle comparable pour la résolution approchée de programmes linéaires [Bie02]. Par la suite, Grigoriadis et Khachiyan [GK94] ont légèrement modifié la fonction objectif en prenant le logarithme de la somme des termes exponentiels. Ils ont alors montré la propriété suivante :

Propriété 4.1. *Soit x un multiflot réalisable de congestion maximale λ, et soit la fonction potentiel $\Phi = \ln(\sum_a \exp(\frac{\alpha}{\varepsilon}\frac{x_a}{u_a}))$, alors*

$$\frac{\alpha}{\varepsilon}\lambda \leqslant \Phi(x) \leqslant \frac{\alpha}{\varepsilon}\lambda + \ln m \tag{4.13}$$

où m est le nombre d'arcs.

En particulier, avec $\alpha = \delta + \ln m$, tout multiflot x qui minimise Φ à δ près, minimise la congestion maximale λ à ε près.

Il est intéressant de noter qu'une approche similaire a été effectuée dans [DM07] où il a été observé que la minimisation de la congestion tend à augmenter le nombre de chemins actifs.

4.2.3 Problème du flot k-séparable de coût minimal

Au cœur de la résolution de notre problème, l'application de l'algorithme de déviation de flot nécessite la résolution d'un programme linéaire pour déterminer une direction de descente. Dans notre cas, nous verrons que ce programme linéaire correspond à un problème de flot de coût minimal avec des contraintes de k-séparabilité relâchées (relaxation linéaire des variables entières). Ainsi, nous décrivons dans cette partie comment nous pouvons adapter nos modèles et méthodes présentées aux chapitres 2 et 3 à ce problème de flot k-séparable.

Le problème du flot k-séparable de coût minimal consiste à router une demande sur au plus k chemins tout en minimisant le coût d'utilisation du réseau. Ce problème s'inscrit donc dans les problèmes de flot k-séparable que nous avons traités dans les chapitres précédents. Notre approche peut s'adapter facilement à ce nouvel objectif.

Nous utilisons ici les même notations que dans les chapitres précédents. Le nombre maximum de chemins autorisés est noté H. Chacun des chemins, numérotés de $h = 1$ à $h = H$, est modélisé par un flot en variables 0-1. Ce flot est ensuite décomposé sur les chemins. Ainsi, la variable x_p^h indique la quantité de flot routée sur p pour le chemin numéro h. La variable y_p^h est la variable de décision associée. Elle indique si p est le chemin numéro h ou non.

De plus, c_a représente le coût associé à l'arc a. Ainsi, le coût pour un chemin p est $c_p = \sum_a \delta_a^p c_a$. Enfin, d est la demande à router. Le modèle arc-chemin obtenu par décomposition de Dantzig-Wolfe s'établit alors comme suit :

$$
\text{(KMCFP)} \left\{
\begin{array}{lll}
\min \displaystyle\sum_{h=1}^{H} \sum_{p \in \mathcal{P}} c_p x_p^h & & \\
s.c. & & \\
\quad \displaystyle\sum_{h=1}^{H} \sum_{p \in \mathcal{P}} \delta_a^p x_p^h \leq u_a & \forall a \in A & (a) \\
\quad \displaystyle\sum_{h=1}^{H} \sum_{p \in \mathcal{P}} x_p^h = d & & (b) \\
\quad x_p^h - u_p y_p^h \leqslant 0 & \forall h = 1 \dots H,\ \forall p \in \mathcal{P} & (c) \\
\quad \displaystyle\sum_{p \in \mathcal{P}} y_p^h \leqslant 1 & \forall h = 1 \dots H & (d) \\
\quad \displaystyle\sum_{p \in \mathcal{P}} x_p^{h+1} - \sum_{p \in \mathcal{P}} x_p^h \leqslant 0 & \forall h = 1 \dots H-1 & (e) \\
\quad x_p^h \geqslant 0 & \forall h = 1 \dots H,\ \forall p \in \mathcal{P} & (f) \\
\quad y_p^h \in \{0,1\} & \forall h = 1 \dots H,\ \forall p \in \mathcal{P} & (g)
\end{array}
\right. \tag{4.14}
$$

Comme pour le problème du flot k-séparable maximal, les contraintes (4.14.a) assurent le respect des capacités. Les contraintes de couplage (4.14.c) modélisent le lien entre les variables de flot x_p^h et les variables de support de flot y_p^h. Les contraintes (4.14.d) imposent l'unicité dans l'affectation d'un chemin p à un numéro h. Cette modélisation souffre du même problème de symétrie qui peut également être en

partie traité par les contraintes de "variable ordering" (4.14.e) (cf. chapitre 3). Seuls changent la fonction objectif et l'ajout de la contrainte de demande (4.14.b).

Pour la relaxation linéaire, la propriété 3.1 de saturation des contraintes de couplage est également vérifiée. Ceci permet de simplifier la relaxation linéaire qui s'établit alors comme suit :

$$
\text{(RL)} \left\{ \begin{array}{lll}
\min \displaystyle\sum_{h=1}^{H} \sum_{p \in \mathcal{P}} c_p x_p^h & & \\
s.c. & & \\
\quad \displaystyle\sum_{h=1}^{H} \sum_{p \in \mathcal{P}} \delta_a^p x_p^h \leqslant u_a & \forall a \in A & (a) \\
\quad \displaystyle\sum_{h=1}^{H} \sum_{p \in \mathcal{P}} x_p^h = d & & (b) \\
\quad \displaystyle\sum_{p \in \mathcal{P}} \frac{x_p^h}{u_p} \leqslant 1 & \forall h = 1 \dots H & (c) \\
\quad \displaystyle\sum_{p \in \mathcal{P}} x_p^{h+1} - \sum_{p \in \mathcal{P}} x_p^h \leqslant 0 & \forall h = 1 \dots H-1 & (d) \\
\quad x_p^h \geqslant 0 & \forall h = 1 \dots H \quad \forall p \in \mathcal{P} & (e)
\end{array} \right. \tag{4.15}
$$

Notons respectivement $\pi_a \leqslant 0$, σ quelconque, $\lambda^h \leqslant 0$ et $\nu^h \leqslant 0$ les variables duales associées aux contraintes primales $(4.15.a)$, $(4.15.b)$, $(4.15.c)$ et $(4.15.d)$ de la relaxation linéaire (RL). Alors, le sous-problème consiste à trouver une variable minimisant le coût réduit donné par :

$$
\begin{aligned}
\overline{c_p^h} &= c_p - \sum_{a \in A} \delta_a^p \pi_a - \sigma - \frac{\lambda^h}{u_p} - (\nu^{h-1} - \nu^h) \\
&= \sum_{a \in A} \delta_a^p (c_a - \pi_a) - \sigma - \frac{\lambda^h}{u_p} - (\nu^{h-1} - \nu^h)
\end{aligned} \tag{4.16}
$$

Le sous-problème est décomposable en H sous-problèmes indépendants. Chacun se réduit à un problème de chemin optimal où les deux seuls termes dépendant du choix des arcs du chemins sont :

- $\sum_{a \in A} \delta_a^p (c_a - \pi_a)$ qui correspond à une longueur (positive) à minimiser,
- $\frac{-\lambda^h}{u_p}$ qui est également à minimiser puisque $\lambda^h \leqslant 0$, ce qui revient à chercher un chemin de capacité maximale.

Le sous-problème est donc équivalent à celui du flot k-séparable maximal et se résout donc par le même algorithme.

Il reste néanmoins un point délicat à traiter : l'initialisation de la génération de colonnes. Contrairement au problème du flot k-séparable maximal, un sous-ensemble vide de chemins ne permet pas d'obtenir une solution initiale réalisable à cause de la contrainte de demande. De plus, après un branchement, une solution réalisable pour le nœud père dans l'arbre de décision peut être interdite par les

contraintes ajoutées au nœud courant. Il est donc nécessaire de bénéficier d'une procédure rapide et sûre de génération d'une solution initiale.

La méthode que nous avons choisie consiste à générer des chemins susceptibles d'augmenter la quantité de flot routé jusqu'à satisfaire la demande. Le problème correspondant pour cette génération de colonnes est donc ici un problème de flot k-séparable maximum. Cependant, il est à noter que nous avons restreint notre étude aux flots simples par souci de clarté. Dans le cas de multiflots, en utilisant le même raisonnement, le problème à considérer est le problème de multiflot k-séparable concurrent maximal (voir le problème de multiflot concurrent maximal décrit précédemment) pour effectuer une montée en charge jusqu'à router la demande. Ceci explique la modélisation suivante du flot maximum qui facilitera une transition future vers des modèles multiflots. Dans notre cas, c'est la relaxation linéaire du problème qui nous intéresse. Elle s'établit comme suit :

$$
(\mathrm{RL}_2)\begin{cases}
\max\ \theta & & & \\
s.c. & & & \\
\quad \displaystyle\sum_{h=1}^{H}\sum_{p\in\mathcal{P}} \delta_a^p x_p^h \leqslant u_a & \forall a \in A & (a) \\
\quad \displaystyle\sum_{h=1}^{H}\sum_{p\in\mathcal{P}} x_p^h = \theta d & & (b) \\
\quad \displaystyle\sum_{p\in\mathcal{P}} \frac{x_p^h}{u_p} \leqslant 1 & \forall h = 1 \dots H & (c) \\
\quad \displaystyle\sum_{p\in\mathcal{P}} x_p^{h+1} - \sum_{p\in\mathcal{P}} x_p^h \leqslant 0 & \forall h = 1 \dots H-1 & (d) \\
\quad x_p^h \geqslant 0 & \forall h = 1 \dots H \quad \forall p \in \mathcal{P} & (e)
\end{cases}
\tag{4.17}
$$

Ce problème permet de débuter la génération de colonnes sur n'importe quel sous-ensemble de chemins, même vide. Dès que la charge θ est supérieure à 1, la solution est réalisable pour la relaxation linéaire (RL). On peut alors reprendre la génération de colonnes pour le problème du flot k-séparable de coût minimal et l'intégrer dans un algorithme de type Branch and Price.

4.3 Application au k-DCRP

Le modèle (k-DCRP) dispose à la fois de contraintes non linéaires et de variables discrètes. Pour traiter la k-séparabilité, une méthode de Branch and Price est appliquée. Elle est basée sur un reformulation de Dantzig–Wolfe du modèle (k-DCRP). Cette stratégie de branchement repose sur notre capacité à calculer une borne inférieure de la relaxation linéaire du modèle. Pour cela, nous résolvons un problème de flot k-séparable de coût convexe où ce coût est une approximation de l'objectif originale (delai de bout-en-bout maximal). Ce sous-problème est résolu par un algorithme de réduction de potentiel de type Frank-Wolfe qui intègre la génération de colonnes (cf. [Bie02]). Puis, pour chaque solution entière trouvée durant le Branch and Price, une phase de post-optimisation est appliquée afin de calculer la distribution de flot optimale pour l'objectif initial sur un support fixé. L'algorithme s'arrête lorsqu'aucun

branchement ne peut améliorer la meilleur borne supérieure donnée par ces solutions entières. Ainsi, la solution optimale du (k-DCRP) est obtenu. Une description de l'algorithme est rapportée sur la figure 4.2.

Nous appliquons la méthode de séparation et évaluation. Les contraintes d'intégrité sur les variables y_a^h sont donc relâchées. Puis nous utilisons la règle de branchement de Barnhart *et al.* [BJN$^+$98] décrite à la section 3.3.1.

4.3.1 Relaxation linéaire

Dans un premier temps, nous simplifions l'expression du délai de bout-en-bout en remplaçant les variables y_a^h par le ratio x_a^h/u_a dans la contrainte (4.2.g). La contrainte (4.2.c) justifie que le modèle (CR) suivant fournit une borne inférieure du modèle relâché du (k-DCRP). Cela assure la validité de notre méthode de branchement.

$$
\text{(CR)} \begin{cases} \min\ \theta & & & \\ s.c. & & & \\ & (4.2.a), (4.2.b), (4.2.c), (4.2.d), (4.2.e), (4.2.f) & & \\ & \displaystyle\sum_{a \in A} \frac{\lambda x_a^h}{u_a(u_a - x_a)} \leqslant \theta & \forall h = 1 \dots H & (g) \\ & & & \\ & x_a^h \geqslant 0 & \forall a \in A \quad \forall h = 1 \dots H & (h) \\ & y_a^h \geqslant 0 & \forall a \in A \quad \forall h = 1 \dots H & (i) \\ & d_h \geqslant 0 & \forall h = 1 \dots H & (j) \end{cases} \tag{4.18}
$$

Ce problème (CR) de minimisation du délai de bout-en-bout maximal correspond à un problème de min-max "packing". Nous proposons dans la section suivante de résoudre ce problème par un algorithme d'approximation utilisant une fonction potentiel exponentielle [Bie02, Xu01] basé sur l'algorithme de Shahrokhi et Matula [SM90] pour le problème du flot concurrent maximal.

4.3.2 Une méthode de réduction de potentiel pour résoudre (CR)

Notons S l'ensemble $\{x \in \mathbb{R}_+^{A \times \{1,\dots,H\}} / \sum_{h=1}^{H} x_a^h < u_a \ \forall a\}$ (compatibilité avec les capacités). Considérons $\theta^h \geqslant 0$ la fonction délai de bout-en-bout du problème (LR) et $\theta \geqslant 0$ le délai de bout-en-bout maximal définis par :

$$
\theta_a^h(x) = \frac{\lambda x_a^h}{u_a(u_a - x_a)} \qquad \forall x \in S \quad \forall a \in A \quad \forall h \in 1, \dots, H \tag{4.19}
$$

$$
\theta^h(x) = \sum_{a \in A} \theta_a^h(x) \qquad \forall x \in S \quad \forall h \in 1, \dots, H \tag{4.20}
$$

$$
\theta(x) = \max_{h=1 \dots P} \theta^h(x) \qquad \forall x \in S \tag{4.21}
$$

La proposition suivante décrit une propriété importante de ces fonctions qui assure l'exactitude de notre approche.

Proposition 4.1. *La fonction θ_a^h est convexe sur S pour tout $a \in A$ et pour tout $h \in 1, \ldots, H$.*

La démonstration est reportée en annexe A.1.

Alors, $\theta(x)$ peut être évaluée par la fonction potentiel exponentielle $\phi(x, \mu)$ avec $\mu > 0$.

$$\phi(x, \mu) = \mu \ln \sum_{h=1}^{H} \exp(\frac{\theta^h(x)}{\mu}) \tag{4.22}$$

Cette fonction a des propriétés intéressantes. En premier lieu, elle fournit une bonne estimation de la fonction $\theta(x)$ dans le sens où

$$\theta(x) \leqslant \phi(x, \mu) \leqslant \theta(x) + \mu \ln H, \quad \forall x \in S, \quad \forall \mu > 0 \tag{4.23}$$

De plus, cette fonction est convexe comme le montre la proposition suivante.

Proposition 4.2. *La fonction $\phi(x, \mu)$ est convexe sur S pour tout $\mu > 0$.*

La démonstration est reportée en annexe A.2.

La relaxation linéaire (CR) peut donc être résolue de façon approchée en minimisant la fonction convexe $\phi(x, \mu)$ avec les contraintes de flot k-séparable relâchées. Ce problème s'exprime comme un modèle non linéaire (CR2) où μ est un paramètre.

$$\text{(CR2)} \begin{cases} \min \ \phi(x, \mu) \\ s.c. \\ \qquad (4.2.a), (4.2.b), (4.2.c), (4.2.d), (4.2.e), (4.2.f) \\ \\ \qquad x_a^h \geqslant 0 \qquad \forall a \in A \quad \forall h = 1 \ldots H \quad (g) \\ \qquad y_a^h \geqslant 0 \qquad \forall a \in A \quad \forall h = 1 \ldots H \quad (h) \\ \qquad d_h \geqslant 0 \qquad \forall h = 1 \ldots H \quad (i) \end{cases} \tag{4.24}$$

La relaxation de la contrainte non-linéaire de délai de bout-en-bout par cette méthode permet d'aboutir à un schéma d'approximation plus simple à résoudre. En effet, le problème se ramène alors à la minimisation d'une fonction objectif convexe sur un polyèdre. Pour résoudre ce problème, nous avons choisi la méthode de déviation de flot qui a fait l'objet de nombreuses études notamment pour des problèmes de routage dans les réseaux de télécommunication.

À chaque itération i, le calcul de la direction de descente à partir de la solution courante x^i se fait par linéarisation de la fonction objectif. Ce problème s'exprime alors comme le programme linéaire (DFS^i).

$$
(\text{DFS}^i)\begin{cases}
\min\ \nabla\phi(x^i,\mu)^T x & \\
\ s.c. & \\
\quad \displaystyle\sum_{a\in\omega^-(v)} x_a^h - \sum_{a\in\omega^+(v)} x_a^h = \begin{cases} -d_h & \text{if } v=s, \\ d_h & \text{if } v=t, \\ 0 & \text{otherwise.}\end{cases} & \forall h=1\dots H,\ \forall v\in N \quad (a)\\
\quad \displaystyle\sum_{h=1}^{H} x_a^h \leqslant u_a & \forall a\in A \quad (b)\\
\quad \displaystyle\sum_{h=1}^{H} d_h = d & (c)\\
\quad \displaystyle\sum_{a\in\omega^-(v)} \frac{x_a^h}{u_a} \leqslant 1 & \forall v\in V \quad \forall h=1\dots H \quad (d)\\
\quad x_a^h \geqslant 0 & \forall a\in A \quad \forall h=1\dots H \quad (e)\\
\quad d_h \geqslant 0 & \forall h=1\dots H \quad (f)
\end{cases} \tag{4.25}
$$

Une solution optimale de (DFS^i) donne une direction le long de laquelle la recherche linéaire trouve une meilleure solution. Ainsi, l'algorithme suivant est utilisé pour trouver une borne inférieure à la relaxation linéaire (CR2). μ et ϵ sont deux paramètres positifs et U représente la plus grande capacité des arcs du réseau.

Le test d'arrêt est nécessaire pour démontrer le résultat du théorème 4.1. Pour clarifier la lecture de l'algorithme, nous noterons ζ_i la valeur suivante :

$$
\zeta_i = \frac{\epsilon^2\phi(x^i,\mu)^2}{32\lambda\left(\frac{U^2H^3}{\lambda^3}\phi^3(x^i,\mu)+m\right)+\frac{4\lambda^2}{\mu}\left(\frac{UH^2}{\lambda^2}\phi^2(x^i,\mu)+m\right)^2} \tag{4.26}
$$

Algorithme 4.1 Relaxation linéaire du (k-DCRP)

Soit x^0 un flot H-séparable routant la demande.
Répéter
Soit $\bar{x}^i$ une solution optimale de (DFS^i).
$x^{i+1} = (1-\sigma^i)x^i + \sigma^i\bar{x}^i$, où $\bar{\sigma}^i = \arg\min_{0\leqslant\sigma^i\leqslant 1}\{\phi((1-\sigma^i)x^i+\sigma^i\bar{x}^i,\mu)\}$.
$t \leftarrow i+1$
Jusqu'à $\phi(x^i,\mu)-\phi(x^{i+1},\mu) < \zeta_i$
Affecter $(1-\epsilon)\phi(x^i,\mu) - \mu\ln H$ à la borne inférieure.

Le théorème 4.1 nous donne la preuve de l'approximation.

Théorème 4.1. *Soit $\phi^*(\mu)$ une solution optimale de (CR2) et x^i le flot à la fin de l'algorithme 4.1. Alors*

$$
(1-\epsilon)\phi(x^i,\mu) \leqslant \phi^*(\mu) \tag{4.27}
$$

Démonstration. La preuve est adaptée de [BR02].

Considérons un vecteur flot x^i. Pour simplifier les notations, notons

$$g(\sigma) = \phi((1-\sigma)x^i + \sigma\bar{x}^i, \mu) = \phi(x^i + \sigma(\bar{x}^i - x^i)) \tag{4.28}$$

on obtient

$$g(\sigma) - g(0) \leqslant (\phi^*(\mu) - \phi(x^i, \mu))\sigma + \frac{1}{2}g''(\alpha)\sigma^2 \quad \text{où } 0 < \alpha < \sigma \tag{4.29}$$

Pour borner le terme quadratique dans (4.29), notons $x_\alpha = (1-\alpha)x^i + \alpha\bar{x}^i$ et $d = \bar{x}^i - x^i$. Comme $\nu_h(x_\alpha, \mu) < 1 \ \forall h \in \{1, \ldots, H\}$, on obtient de (A.6) et (A.8)

$$g''(\alpha) \leqslant \sum_{h=1}^{H} d^T \nabla^2 \theta^h(x_\alpha) d + \frac{1}{\mu} d^T (ADA^T) d \tag{4.30}$$

La définition des dérivées secondes dans (A.3) implique

$$\begin{aligned}
\sum_{h=1}^{H} d^T \nabla^2 \theta^h(x_\alpha) d &= \sum_{a\in A} \frac{\lambda}{u_a(u_a - x_{\alpha,a})^3} \sum_{h=1}^{H} \Bigg(2 \sum_{i\neq h} \sum_{j\neq h} d_a^i x_{\alpha,a}^h d_a^j \\
&\qquad + 2 \sum_{i\neq h} (u_a - \bar{x}_{\alpha,a}^h + x_{\alpha,a}^h) d_a^i d_a^h + 2(u_a - \bar{x}_{\alpha,a}^h)(d_a^h)^2 \Bigg) \\
\sum_{h=1}^{H} d^T \nabla^2 \theta^h(x_\alpha) d &= \sum_{a\in A} \frac{2\lambda}{u_a(u_a - x_{\alpha,a})^3} \sum_{h=1}^{H} \Big((\bar{d}_a^h)^2 x_{\alpha,a}^h \\
&\qquad + (u_a - \bar{x}_{\alpha,a}^h + x_{\alpha,a}^h) \bar{d}_a^h d_a^h + (u_a - \bar{x}_{\alpha,a}^h)(d_a^h)^2 \Big) \\
&= \sum_{a\in A} \frac{2\lambda}{u_a(u_a - x_{\alpha,a})^3} \sum_{h=1}^{H} \Big((d_a)^2 x_{\alpha,a}^h + (u_a - x_{\alpha,a}) d_a d_a^h \Big) \\
&= \sum_{a\in A} \frac{2\lambda d_a^2}{(u_a - x_{\alpha,a})^3}
\end{aligned} \tag{4.31}$$

Observons que

$$|d_a| = |\bar{x}_a^i - x_a^i| \leqslant u_a \leqslant U \qquad \forall a \in A \tag{4.32}$$

$$\frac{1}{(1-x)^3} \leqslant 8\left(\frac{x}{1-x}\right)^3 + 8 \qquad \forall\, 0 \leqslant x \leqslant 1 \tag{4.33}$$

Sans perte de généralité, on suppose que $u_a \geqslant 1$, pour tout $a \in A$. Alors

$$\begin{aligned}
\sum_{h=1}^{H} d^T \nabla^2 \theta^h(x_\alpha) d &\leqslant 16\lambda \left(U^2 \sum_{a \in A} \left(\frac{\sum_{h=1}^{H} x_{\alpha,a}^h}{u_a(u_a - x_{\alpha,a})} \right)^3 + m \right) \\
&\leqslant 16\lambda \left(U^2 \left(\sum_{h=1}^{H} \sum_{a \in A} \frac{x_{\alpha,a}^h}{u_a(u_a - x_{\alpha,a})} \right)^3 + m \right) \\
&\leqslant 16\lambda \left(U^2 \left(\sum_{h=1}^{H} \frac{\theta^h(x_\alpha)}{\lambda} \right)^3 + m \right) \\
&\leqslant 16\lambda \left(U^2 \left(\frac{H\theta(x_\alpha)}{\lambda} \right)^3 + m \right) \\
&\leqslant 16\lambda \left(\frac{U^2 H^3}{\lambda^3} \phi^3(x_\alpha, \mu) + m \right) \qquad (4.34)
\end{aligned}$$

D'autre part, l'analyse spectrale précédente de la matrice symétrique semi-définie positive D prouve que sa plus grande valeur propre γ est bornée de la façon suivante

$$\gamma = \max_{h \in \{1,\ldots,H\}} 2\nu_h(x,\mu)(1 - \nu_h(x,\mu)) \leqslant \frac{1}{2} \qquad (4.35)$$

Ainsi

$$\begin{aligned}
\frac{1}{\mu} d^T A D A^T d &\leqslant \frac{1}{2\mu} ||A^T d||^2 \\
&\leqslant \frac{1}{2\mu} \sum_{h=1}^{H} \left(\nabla \theta^h(x_\alpha)^T d \right)^2 \\
&\leqslant \frac{1}{2\mu} \left(\sum_{h=1}^{H} \sum_{a \in A} \frac{\lambda}{u_a(u_a - x_{\alpha,a})^2} \left(x_{\alpha,a}^h \sum_{i \neq h} d_a^i + (u_a - \bar{x}_{\alpha,a}^h) d_a^h \right) \right)^2 \\
&\leqslant \frac{1}{2\mu} \left(\sum_{a \in A} \sum_{h=1}^{H} \frac{\lambda}{u_a(u_a - x_{\alpha,a})^2} \left(x_{\alpha,a}^h d_a + (u_a - x_{\alpha,a}) d_a^h \right) \right)^2 \\
&\leqslant \frac{1}{2\mu} \left(\sum_{a \in A} \frac{\lambda d_a}{(u_a - x_{\alpha,a})^2} \right)^2 \qquad (4.36)
\end{aligned}$$

Observons que

$$\frac{1}{(1-x)^2} \leqslant 2\left(\frac{x}{1-x}\right)^2 + 2 \qquad \forall\ 0\ \leqslant x \leqslant 1 \tag{4.37}$$

Alors

$$\begin{aligned} \frac{1}{\mu} d^T A D A^T d &\leqslant \frac{2\lambda^2}{\mu}\left(U \sum_{a\,\in A}\left(\frac{x_{\alpha,a}}{(u_a - x_{\alpha,a})}\right)^2 + m\right)^2 \\ &\leqslant \frac{2\lambda^2}{\mu}\left(\frac{UH^2}{\lambda^2}\phi^2(x_\alpha,\mu) + m\right)^2 \end{aligned} \tag{4.38}$$

Si σ est choisi tel que $\phi(x^i + \sigma(\bar{x}^i - x^i), \mu) \leqslant \phi(x^i, \mu)$

alors (4.34), (4.38) et la convexité de ϕ implique

$$g''(\alpha) \leqslant 16\lambda\left(\frac{U^2H^3}{\lambda^3}\phi^3(x^i,\mu) + m\right) + \frac{2\lambda^2}{\mu}\left(\frac{UH^2}{\lambda^2}\phi^2(x^i,\mu) + m\right)^2 \tag{4.39}$$

En substituant dans (4.29), on obtient

$$\begin{aligned} \phi(x^i + \sigma(\bar{x}^i - x^i), \mu) - \phi(x^i,\mu) &\leqslant (\phi^*(\mu) - \phi(x^i,\mu))\sigma \\ &+\tfrac{1}{2}\left(16\lambda\left(\tfrac{U^2H^3}{\lambda^3}\phi^3(x^i,\mu) + m\right) + \tfrac{\lambda^2}{\mu}\left(\tfrac{UH^2}{2\lambda^2}\phi^2(x^i,\mu) + m\right)^2\right)\sigma^2 \end{aligned} \tag{4.40}$$

Comme expliqué dans [BR02], après un calcul pour minimiser le membre de droite quadratique, on a

$$\phi(x^{t+1},\mu) - \phi(x^i,\mu) \leqslant -\frac{(\phi^*(\mu) - \phi(x^i,\mu))^2}{32\lambda\left(\frac{U^2H^3}{\lambda^3}\phi^3(x^i,\mu) + m\right) + \frac{4\lambda^2}{\mu}\left(\frac{UH^2}{\lambda^2}\phi^2(x^i,\mu) + m\right)^2} \tag{4.41}$$

Si $\phi^*(\mu) < (1-\epsilon)\phi(x^i,\mu)$, on obtient donc

$$\phi(x^{t+1},\mu) - \phi(x^i,\mu) \leqslant -\frac{\epsilon^2\phi^2(x^i,\mu)}{32\lambda\left(\frac{U^2H^3}{\lambda^3}\phi^3(x^i,\mu) + m\right) + \frac{4\lambda^2}{\mu}\left(\frac{UH^2}{\lambda^2}\phi^2(x^i,\mu) + m\right)^2} \tag{4.42}$$

et l'approximation est démontrée. □

On constate que le programme linéaire (DFS^i) correspond à la relaxation linéaire du problème du flot k-séparable de coût minimal dont le vecteur coût est donné par le gradient de $\phi(x,\mu)$. Nous résolvons ce problème par génération de colonnes.

4.3.3 Génération de colonnes

Soit (DFS$_2^i$) le modèle obtenu par décomposition de Dantzig–Wolfe sur le modèle (DFS^i). Pour chaque chemin $p \in \mathcal{P}$, x_p^h représente la contribution de flot sur le chemin p pour le numéro de chemin h. δ_a^p est le vecteur indicateur qui identifie les arcs a constituant le chemin p.

Les variables x_a^h de (DFS^i) sont alors remplacées par

$$x_a^h = \sum_{p\in\mathcal{P}} \delta_a^p x_p^h \qquad \forall a \in A \quad \forall h = 1,\dots,H \tag{4.43}$$

$$\text{(DFS}_2^i\text{)} \begin{cases} \min \displaystyle\sum_{h=1}^{H}\sum_{p\in\mathcal{P}} c_p^h x_p^h & \\ s.c. & \\ \quad \displaystyle\sum_{h=1}^{H}\sum_{p\in\mathcal{P}} \delta_a^p x_p^h \leqslant u_a \quad \forall a \in A & (a) \\ \quad \displaystyle\sum_{h=1}^{H}\sum_{p\in\mathcal{P}} x_p^h = d & (b) \\ \quad \displaystyle\sum_{p\in\mathcal{P}} \frac{x_p^h}{u_p} \leqslant 1 \qquad \forall h = 1\dots H & (c) \\ \quad x_p^h \geqslant 0 \qquad \forall p \in \mathcal{P} \quad \forall h = 1\dots H & (d) \end{cases} \tag{4.44}$$

Notons que le calcul du gradient de $\phi(x,\mu)$ est relativement simple.

$$\begin{aligned} \frac{\partial\phi}{\partial x_a^h}(x,\mu) &= \frac{\lambda}{u_a(u_a-x_a)^2}\left(\sum_{i\neq h}\nu_i(x,\mu)x_a^h + \nu_h(x,\mu)(u_a-\bar{x}_a^h)\right) \\ &= \frac{\lambda(x_a^h + \nu_h(x,\mu)(u_a-x_a))}{u_a(u_a-x_a)^2} \\ &= \frac{\theta_a^h(x) + \frac{\lambda}{u_a}\nu_h(x,\mu)}{u_a - x_a} \end{aligned} \tag{4.45}$$

Pour chaque chemin p et pour chaque numéro h, le coût c_p^h est donné par

$$c_p^h = \sum_{a \in p} \frac{\partial \phi}{\partial x_a^h}(x^i, \mu) \tag{4.46}$$

Les contraintes $(4.25.d)$ donnent les contraintes de capacité $(4.44.a)$. Les contraintes $(4.25.a)$ implique que $\sum_{p \in \mathcal{P}} x_p^h = d_h$ pour tout $h = 1 \dots H$. Alors les contraintes $(4.25.e)$ mènent à la contrainte de demande $(4.44.b)$.

Les contraintes $(4.44.c)$ correspondent au choix d'au plus un chemin p pour chaque numéro de chemin h (contraintes $(4.25.f)$ comme justifié ci-dessous).

Propriété 4.2. *Les variables y_a^h et les contraintes $(4.25.b, c)$ peuvent être supprimées du modèle (DFS$_2^i$).*

Démonstration. Soit x une solution réalisable de (DFS$_2^i$). Soit $\overline{y}$ défini par

$$\overline{y_a^h} = \frac{\sum_{p \in \mathcal{P}} \delta_a^p \frac{x_p^h}{u_p}}{\sum_{p \in \mathcal{P}} \frac{x_p^h}{u_p}} \qquad \forall a \in A \quad \forall h = 1 \dots H \tag{4.47}$$

La définition de $\overline{y}$ assure le respect des contraintes $(4.25.b)$. De la même manière, les contraintes $(4.44.c)$ et la définition de la capacité u_p implique que (contraintes $(4.25.c)$)

$$u_a \overline{y_a^h} \geqslant \sum_{p \in \mathcal{P}} \delta_a^p \frac{u_a}{u_p} x_p^h \geqslant \sum_{p \in \mathcal{P}} \delta_a^p x_p^h = x_a^h \qquad \forall a \in A \quad \forall h = 1 \dots H$$

Il découle de la définition de l'ensemble $\mathcal{P}$ que les chemins traversant un même nœud sont distincts, et les contraintes $(4.25.f)$ sont satisfaites :

$$\sum_{a \in \delta^-(v)} y_a^h = \frac{\sum_{a \in \delta^-(v)} \sum_{p \in \mathcal{P}} \delta_a^p \frac{x_p^h}{u_p}}{\sum_{p \in \mathcal{P}} \delta_a^p \frac{x_p^h}{u_p}} \leqslant 1 \qquad \forall v \in V \quad \forall h = 1 \dots H$$

Donc les variables y_a^h et les contraintes $(4.25.b, c)$ ne contraignent pas les variables x_p^h. □

Le sous-problème de génération de colonnes se décompose en H problèmes de chemin optimal car il s'agit de minimiser le coût réduit $\bar{c}_p^h$ défini par

$$\begin{aligned} \bar{c}_p^h &= c_p^h - \sum_{a \in p} \pi_a - \frac{\lambda_h}{u_p} - \nu \\ &= \sum_{a \in p} \left(\frac{\partial \phi}{\partial x_a^h}(x^i, \mu) - \pi_a\right) - \frac{\lambda_h}{u_p} - \nu \end{aligned} \tag{4.48}$$

où $\pi_a \leqslant 0$ (resp. $\lambda_h \leqslant 0$ et $\nu \in \mathbb{R}$) sont les variables duales associées aux contraintes (4.44.a) (resp. (4.44.c) et (4.44.b)). Le calcul du chemin élémentaire de coût minimal s'effectue de la même manière que pour le problème du flot k-séparable (cf. algorithme 3.1). Il s'agit de trouver le meilleur compromis entre le plus court chemin et le chemin de capacité maximale. La génération de colonnes s'arrête lorsque le coût réduit associé au chemin optimal est positif ou nul

4.3.4 Post-optimisation

Un autre point important est que la solution optimale est calculée par rapport à la fonction relâchée $\phi(x, \mu)$ définie dans (4.22). Comme l'algorithme 4.1 donne une borne inférieure de la fonction $\theta(x)$ définie dans (4.21), un support optimal est trouvé pour les solutions entière et relâchée (supports qui peuvent être différent pour chaque solution). Comme la distribution de flot sur les chemins actifs peut être sous-optimale pour $\theta(x)$, nous appliquons une procédure de post-optimisation sur chaque support entier trouvé au cours de l'exploration de l'arbre de décision.

Cette procédure de post-optimisation calcule une distribution optimale du flot sur un support fixe pour la fonction $\theta(x)$. On a alors au plus H chemins p_h et on calcule la distribution d_h pour tout h qui minimise $\tilde{\theta}(d)$ défini par

$$\tilde{\theta}^h(d) = \sum_{a \in A} \frac{\lambda \delta_a^h}{u_a - x_a} = \sum_{a \in p_h} \frac{\lambda}{u_a - x_a} \tag{4.49}$$

$$\tilde{\theta}(d) = \max_{h \in \{1,\ldots,H\}} \tilde{\theta}^h(d) \tag{4.50}$$

où $\delta_a^h = 1$ si $a \in p_h$ et 0 sinon, et $x_a = \sum_{h=1}^{H} \delta_a^h d_h$.

La solution proposée par le Branch and Price est un flot réalisable. La fonction $\tilde{\theta}$ assure de toujours satisfaire les contraintes de capacité. Il reste uniquement la contrainte de demande à vérifier. C'est pourquoi le gradient de la fonction $\tilde{\phi}$ est projeté pour trouver une direction de descente réalisable.

$$\tilde{\phi}(d) = \mu \ln \sum_{h=1}^{H} e^{\tilde{\theta}^h(d)/\mu} \tag{4.51}$$

La définition du gradient de $\tilde{\phi}$ nécessite la définition des dérivées partielles de $\tilde{\theta}^h$ et de l'expression de $\tilde{\nu}^h$

$$\frac{\partial \tilde{\theta}^h}{\partial d_i}(d) = \sum_{a \in p_h \cap p_i} \frac{\lambda}{(u_a - x_a)^2} \tag{4.52}$$

$$\tilde{\nu}^h(d) = \frac{e^{\tilde{\theta}^h(d)/\mu}}{\sum_{i=1}^{H} e^{\tilde{\theta}^i(d)/\mu}} \tag{4.53}$$

Alors

$$\frac{\partial \tilde{\phi}}{\partial d_i}(d) = \sum_{h=1}^{H} \tilde{\nu}^h(d) \sum_{a \in p_h \cap p_i} \frac{\lambda}{(u_a - x_a)^2} \tag{4.54}$$

La direction de descente $\Delta(d)$ est donnée par

$$\Delta_h(d) = \frac{1}{H} \sum_{i=1}^{H} \frac{\partial \tilde{\phi}}{\partial d_i}(d) - \frac{\partial \tilde{\phi}}{\partial d_h}(d) \tag{4.55}$$

Dans cette direction, $\tilde{\theta}$ est minimisé en utilisant sa dérivée directionnelle

$$\tilde{\theta}'(d, \Delta) = \max_{h \in I(d)} [\nabla \tilde{\theta}^h(d)]^T \Delta \tag{4.56}$$

où $I(d) = \{h \mid \tilde{\theta}^h(d) = \tilde{\theta}(d)\}$.

Enfin, notons que pour réduire l'exploration, une solution initiale est calculée. On peut remarquer que n'importe quel flot k-séparable routant la demande est réalisable pour le (k-DCRP). Ainsi, on peut utiliser une solution optimale du problème du flot k-séparable de coût minimal. Les capacités sont néanmoins réduites au $4/5$ des capacités initiales pour empêcher la solution d'être trop proche de la saturation. Le coût de chaque arc est initialisé à λ/u_a^2 qui correspond au gradient de $\lambda/(u_a - x_a)$ (cf. (4.1)) à l'origine. Les différentes étapes de l'algorithme de résolution du (k-DCRP) sont résumées sur la figure 4.2.

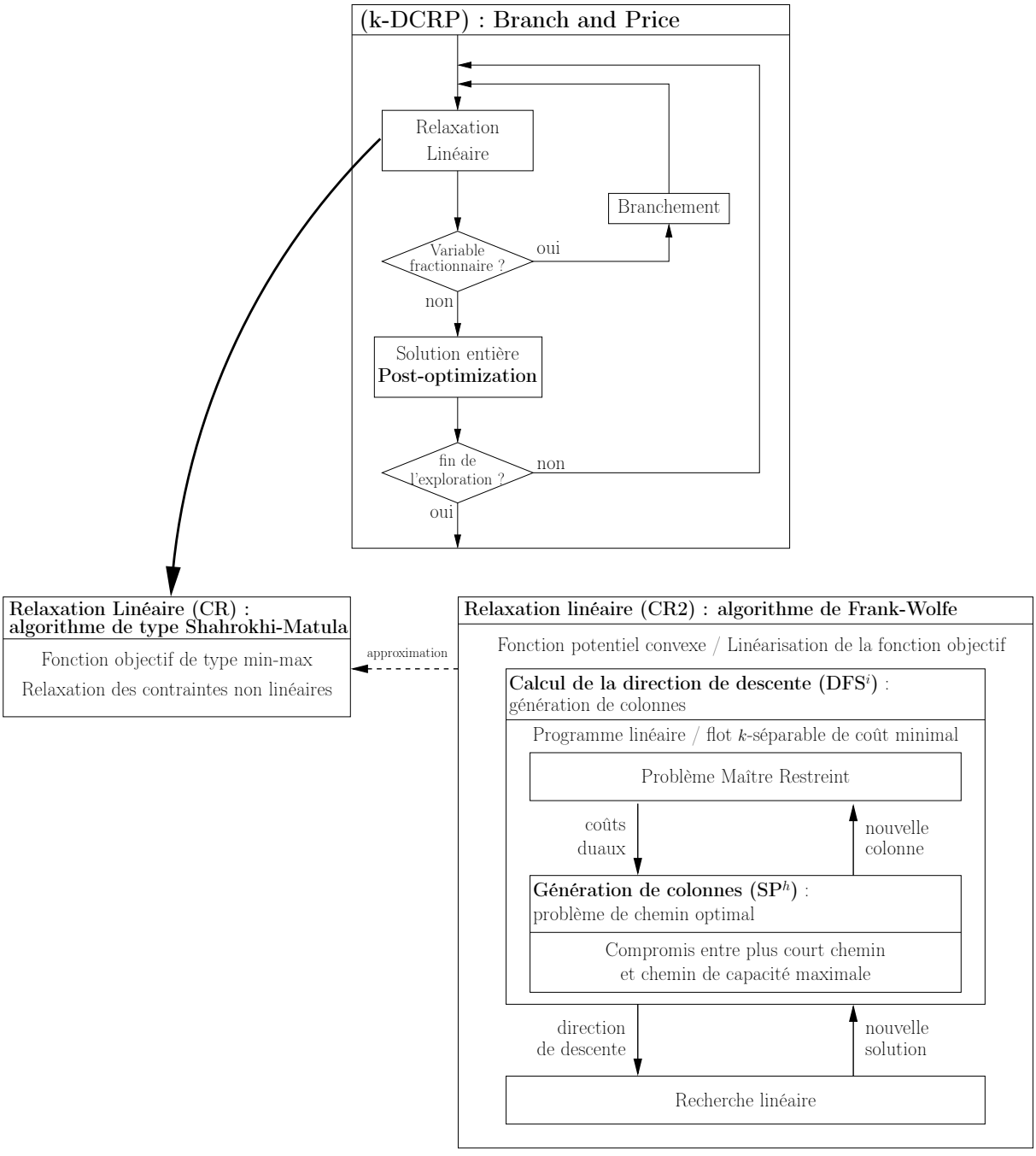


FIG. 4.2 – Résolution du (k-DCRP).

4.4 Résultats numériques

Deux ensembles d'instances ont été utilisés pour mesurer l'efficacité de notre approche. Le premier est un ensemble de réseaux de télécommunication optique de 10 à 14 nœuds et avec une faible connectivité.

Un second ensemble d'instances a été généré en utilisant le générateur de « transit grid » développé par G. Waissi[1] (cf. section 3.5). La topologie de ces instances (figure 4.3) se rapproche des réseaux de transport. Avec une plus forte connectivité, ils nous permettront de tester les limites de notre approche. Pour nos expériences, nous utiliserons des grilles de 10 nœuds. Les capacités ont été choisies aléatoirement dans l'intervalle $[1; 1000]$. La demande a été calculée de la façon suivante : soit H le nombre maximum de chemins actifs autorisés. Un flot k-séparable maximal est calculé (par la méthode du Branch and Price présentée au chapitre 3). Soit F^* la valeur de cette solution optimale. La demande est alors de $3F^*/4$ et dépend ainsi de H.

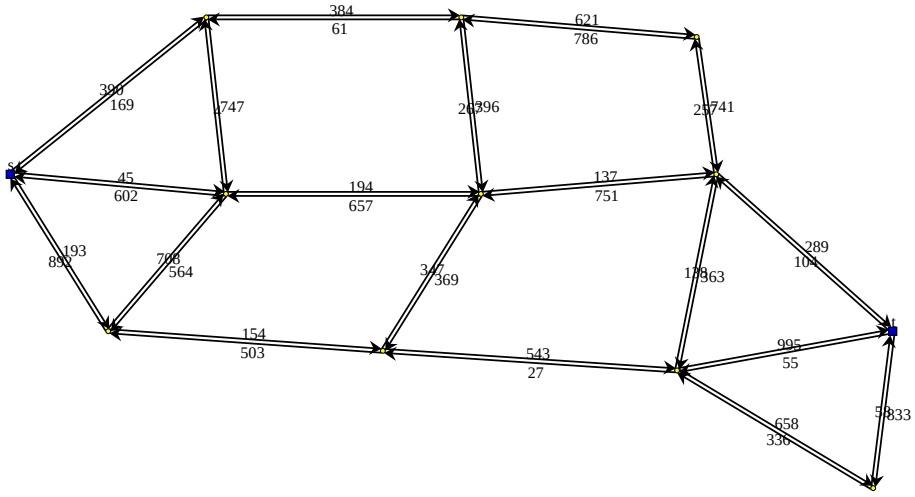


FIG. 4.3 – « transit grid » avec 12 nœuds et 38 arcs.

Les tests ont été effectués sur un ordinateur muni d'un processeur Pentium 4 et de 4 Go de mémoire vive. La durée limite d'une exécution a été fixée à trois heures. Le programme a été implémenté dans le langage C et compilé avec gcc (version 3.3.2) avec le flag d'optimisation -O2 activé.

Pour comparer une solution du modèle (k-MADP) avec une solution du modèle (k-DCRP), nous rapportons la valeur correspondante du plus grand délai de bout-en-bout. Cependant, la distribution du flot sur les chemins d'une solution optimal de (k-MADP) est optimale pour la fonction ψ. Pour avoir une comparaison juste de la performance de chaque modèle, la procédure de post-optimisation décrite précédemment est appliquée à cette solution. Nous rapportons alors la nouvelle valeur du plus grand délai de bout-en-bout.

Le tableau 4.1 présente les résultats pour les réseaux de télécommunication tandis que le tableau 4.2 montre ceux pour les « transit grids » à 10 nœuds. Pour chaque instance, les tests ont été effectués pour un

[1] `http://www.informatik.uni-trier.de/~naeher/Professur/research`

valeur croissante de H (et donc une demande croissante comme expliqué précédemment). Les résultats sont présentés à la fois pour (k-DCRP) (section « Délai de bout-en-bout ») and (k-MADP) (section « Delai moyen »). Pour chaque méthode, la valeur optimale (colonne « z^* ») et le temps d'exécution en secondes (colonne « CPU ») sont rapportés. Pour le modèle (k-MADP), la colonne « délai » correspond à l'évaluation de la solution optimale par rapport à la fonction objectif du modèle (k-DCRP). La colonne « PO » indique la même valeur après la procédure de post-optimisation.

instance		Délai moyen				Délai de bout-en-bout	
graphe	H	z^*	CPU (s)	delai	PO	z^*	CPU (s)
abilene	1	15.00	0.04	2.00	**2.00**	**2.00**	0.06
11-14	2	32.91	0.05	2.30	**2.20**	**2.20**	0.17
	3	16.44	0.34	1.43	**1.41**	**1.38**	2.57
	4	16.44	1.50	1.43	**1.41**	**1.38**	10.22
	5	16.44	1.96	1.43	**1.41**	**1.38**	31.26
	6	16.44	2.84	1.43	**1.41**	**1.38**	249.09
cost	1	5.10	0.02	0.11	**0.11**	**0.11**	0.01
11-25	2	9.02	0.39	0.22	**0.17**	**0.17**	5.19
	3	12.28	4.73	0.30	**0.21**	**0.21**	42.53
	4	16.14	18.77	0.41	**0.27**	**0.27**	290.12
	5	7.97	474.28	0.26	0.21	**0.16**	4926.12
nsf	1	5.10	0.01	0.17	**0.17**	**0.17**	0.15
14-21	2	7.56	0.33	0.17	**0.17**	**0.17**	1.88
	3	12.59	0.80	0.28	**0.23**	**0.23**	8.98
	4	15.25	9.05	0.30	0.28	**0.26**	66.81
	5	8.76	32.79	0.22	0.19	**0.18**	229.90
	6	8.74	134.20	0.22	**0.18**	**0.18**	895.62
zang	1	7.63	0.04	0.35	**0.35**	**0.35**	0.16
10-15	2	11.76	0.68	0.33	**0.32**	**0.32**	6.84
	3	21.66	5.51	0.46	**0.43**	**0.43**	44.81
	4	10.49	49.99	0.30	0.28	**0.27**	1060.92
	5	10.46	538.17	0.30	0.28	$\leqslant$**0.27**	-
	6	$\leqslant$10.46	-	0.30	**0.28**	$\leqslant$0.29	-

TAB. 4.1 – Réseaux de télécommunication avec une faible connectivité.

Sur le tableau 4.1, on peut voir pour le (k-DCRP) que le temps d'exécution croît rapidement quand H augmente. Pour de petites valeurs de H, le (k-MADP) produit de bonnes approximations de la solution optimale du (k-DCRP) après application de la procédure de post-optimisation (PO). Cependant, quand H augmente, l'écart entre les solutions proposées par (k-MADP) et (k-DCRP) se creuse. Une autre qualité en faveur de (k-MADP) est le faible temps d'exécution nécessaire en comparaison avec (k-DCRP). Par exemple, sur l'instance « abilene », pour $H = 6$, deux ordres de grandeur séparent les temps d'exécution respectifs des deux méthodes. Pour l'instance, « zang » une solution approchée a pu être calculée dans la limite de temps pour la plus grande valeur de H ($H = 5$).

instance		Délai moyen				Délai de bout-en-bout	
graphe	H	z^*	CPU (s)	delai	PO	z^*	CPU (s)
tg10-3	1	2.92	0.06	2.06	**2.06**	**2.06**	0.40
12-40	2	5.84	4.42	2.23	**2.22**	**2.22**	24.84
	3	8.09	76.88	2.45	**2.33**	**2.33**	1173.87
	4	10.26	1871.23	2.57	2.47	$\leqslant$**2.46**	-
	5	$\leqslant$11.15	-	2.72	2.59	$\leqslant$**2.58**	-
tg10-4	1	9.14	0.18	2.99	**2.99**	**2.99**	3.35
12-40	2	14.82	9.81	2.85	**2.77**	**2.77**	47.22
	3	18.72	220.06	3.71	3.10	**3.06**	2195.04
	4	$\leqslant$19.94	-	4.06	3.44	$\leqslant$**3.22**	-
tg10-5	1	12.74	0.05	2.18	**2.18**	**2.18**	0.59
12-38	2	12.42	9.71	1.89	**1.81**	**1.81**	85.07
	3	15.94	667.70	3.35	2.31	**2.21**	9592.66
	4	$\leqslant$17.55	-	3.20	3.06	$\leqslant$**2.59**	-
tg10-6	1	7.14	0.04	1.54	**1.54**	**1.54**	0.72
12-40	2	14.16	2.20	2.22	**1.93**	**1.93**	15.03
	3	17.94	72.28	2.36	2.31	**2.10**	526.02
	4	21.86	3953.02	3.22	2.41	$\leqslant$**2.16**	-
	5	$\leqslant$12.19	-	2.24	1.87	$\leqslant$**1.64**	-
tg10-7	1	6.79	0.03	1.86	**1.86**	**1.86**	1.36
12-40	2	13.49	8.50	3.05	2.56	**2.55**	58.27
	3	18.56	419.56	3.31	2.87	**2.86**	6233.33
	4	$\leqslant$20.05	-	3.61	3.32	$\leqslant$**3.08**	-
tg10-10	1	10.68	0.06	2.87	**2.87**	**2.87**	1.54
12-40	2	18.38	4.74	4.60	3.71	**3.70**	47.60
	3	19.78	403.24	4.08	3.81	**3.50**	4592.37
	4	$\leqslant$22.65	-	5.63	**3.86**	$\leqslant$4.12	-

TAB. 4.2 – « Transit grids » à 10 nœuds.

Le tableau 4.2 présente les résultats sur les instances de type « transit grid ». Ces instances sont pus difficile à résoudre pour plusieurs raisons : premièrement, leur topologie est plus dense que pour les réseaux de télécommunication. Deuxièmement, les capacités sont très variables. Pour ces instances, on peut voir que (k-MADP) fournit de bonnes approximations pour un temps d'exécution raisonnable. (k-DCRP) est capable de résoudre ces instances pour H jusqu'à 3. Comme le temps d'exécution croît rapidement en fonction de H, $H = 4$ est ici la limite pour le (k-DCRP).

Les travaux présentés dans ce chapitre 4 ont fait l'objet d'un rapport de recherche [TDM06].

CONCLUSION ET PERSPECTIVES

Ces travaux de thèse s'inscrivent dans une étude plus globale des problèmes de conception de réseaux sécurisés (tolérants aux pannes) posés par les évolutions des technologies, des protocoles et des demandes en matière de qualité de services. En collaboration avec nos partenaires du projet PRESTO[2], nous avons été amenés à étudier les différentes problématiques liées aux nouvelles architectures réseaux et aux besoins croissants de fiabilité des services offerts par les opérateurs.

Dans un premier temps, nous nous sommes intéressés plus particulièrement aux flots k-séparables. En effet, contrairement aux réseaux IP, les décisions de routage dans les réseaux MPLS se prennent principalement au niveau du routeur d'entrée. Celui-ci affecte à chaque paquet un LSP (*Label Switch Path*) qui détermine son parcours de bout-en-bout du réseau. Ainsi, il est naturel de chercher à limiter le nombre de ces LSP afin, entre autres, de désengorger les tables de routage. Cette contrainte se modélise alors par la notion de flot k-séparable [BKS05] qui généralise le flot entier (flot routé sur un unique chemin).

À notre connaissance, peu d'études traitent des flots k-séparables et celles-ci portent généralement sur des résultats de complexité et des algorithmes d'approximations [BKS05, KS06, Kol05, MS06]. Cependant, à la suite de [Kle96], de nombreux travaux abordent des problèmes de multiflot entier. En particulier, nous nous sommes appuyés sur les travaux de Barnhart *et al.* [BHV00] pour proposer des modèles, pour les problèmes de flots k-séparables, adaptés à une résolution exacte par la méthode du Branch and Price. Par souci de clarté, nous avons décrit notre approche dans le cas particulier du flot k-séparable maximal (une seule paire origine-destination). Toutefois, nous avons gardé une démarche suffisamment générale pour qu'elle puisse être adaptée à d'autres problèmes de flot ou multiflot k-séparable.

Les résultats obtenus par notre approche sont très encourageants. Elle permet de résoudre dans des temps acceptables des instances de grandes tailles (jusqu'à 100 nœuds et 400 arcs). En particulier, nous avons mis en évidence l'intérêt de la méthode de génération de colonnes (modèle arc-chemin) par rapport au modèle arc-sommet. Dans notre cas, le modèle arc-chemin permet d'éviter un grand nombre de solutions fractionnaires inhérentes à la formulation arc-sommet. Cela se concrétise dans l'analyse du gap (écart relatif entre la solution optimale fractionnaire et la solution optimale entière) qui est beaucoup plus faible pour le Branch and Price.

Enfin, en s'inspirant d'études telles que [SS01], nous avons considéré des contraintes visant à « casser » cette symétrie et à limiter l'exploration de l'arbre de décision en imposant une hiérarchie entre les variables, tout en conservant l'optimum. L'ajout de telles contraintes s'est rapidement avéré très efficace

[2]Protection et sécurisation des nouvelles architectures de réseaux. Projet de l'ACI CNRS Sécurité Informatique. `http://www-sop.inria.fr/mascotte/PRESTO/`

dans la majorité des cas. Cependant, nous avons pu observer une dégradation des temps d'exécution dans quelques cas pathologiques. La saturation de ces nouvelles contraintes provoquent un fractionnement plus important de la solution alors même qu'une solution entière de même valeur peut exister. Cette situation limite actuellement les performances de notre approche. De plus, cette situation semble se produire également pour d'autres problèmes [SS01]. Il pourrait alors s'avérer judicieux de porter nos prochains efforts de recherche sur cette problématique.

Dans un deuxième temps, nous avons appliqué notre approche des flots k-séparables au problème du délai de bout-en-bout. Ce problème consiste à router une certaine demande en minimisant le délai de bout-en-bout maximal. En s'appuyant sur des travaux existants [BKP03, BaO06], nous avons modélisé le délai de bout-en-bout d'un paquet comme l'agrégation des délais des arcs parcourus par ce paquet, et les délais sur les arcs sont modélisés par la classique fonction de Kleinrock [Kle64]. Ainsi, pour un flot donné, le délai de bout-en-bout maximal dépend du parcours des paquets dont ce flot est l'agrégation. Ce problème revet alors non seulement une problématique d'optimisation non linéaire au travers de la modélisation du délai mais également un caractère combinatoire relatif au choix des chemins parcourus par les paquets.

Cependant, nous avons remarqué que nos modèles pour les flots k-séparables pouvaient s'adapter à ce problème en offrant une gestion particulièrement intéressante du délai de bout-en-bout. En effet, en modélisant chaque chemin actif séparément, chaque délai de bout-en-bout correspondant s'exprime par une fonction convexe après relaxation des contraintes d'intégrité des variables entières. Ainsi nous pouvons utiliser des méthodes d'optimisation convexe pour résoudre le problème.

Devant la complexité de ce problème, nous avons su combiner plusieurs méthodes performantes pour proposer une résolution exacte. Après relaxation linéaire des variables entières, le problème devient un problème de minimax, continu, avec des contraintes non linéaires. Ce problème est résolu par l'algorithme de Shahrokhi et Matula [SM90], initialement utilisé sur une problème de multiflot concurrent maximal et qui a fait l'objet de nombreuses études, généralisations et améliorations [Bie02]. Le cœur de cet algorithme nécessite la minimisation d'une fonction convexe sur un polyèdre. Pour cela, l'algorithme de déviation de flot semble adapté même si sa convergence, dans notre cas, est une des limitations de notre approche. Enfin, cet algorithme nécessite à chaque itération le calcul d'une direction de descente définie comme solution fractionnaire d'un problème de flot k-séparable de coût minimal, lui-même résolu par génération de colonnes.

Les résultats numériques obtenus reflètent toute la complexité du problème et les limites en matière stabilité numérique de notre approche. Toutefois, sur des instances de tailles modestes (10 à 15 nœuds), notre méthode a le mérite de proposer des solutions exactes pour le problème du délai de bout-en-bout. D'autre part, nous avons proposé de comparer ces résultats avec des résultats approchés obtenus par une méthode similaire appliquée à la minimisation du délai moyen sur un flot k-séparable. Il est alors intéressant de remarquer que ces résultats approchés sont proches de l'optimum pour des temps d'exécution beaucoup plus faibles. Cela illustre également une limite particulière de notre méthode liée au choix d'une fonction potentielle exponentielle pour relâcher les contraintes non linéaires.

Ces travaux de thèse marquent une étape intéressante dans le traitement des problèmes de flots k-séparables. L'engouement des opérateurs de réseaux de télécommunication pour le protocole MPLS laisse augurer de nouveaux problèmes d'optimisation caractérisés par la nécessité de contrôler plus pré-

cisément le parcours des paquets dans le réseau. Ainsi il est probable qu'à l'avenir les flots ne puissent plus être considérés dans un problème de routage indépendamment de leur support. Une meilleur compréhension de la structure des flots k-séparables permettrait d'améliorer notre approche ou d'en définir de plus performantes.

Dans un premier temps, il serait intéressant de concrétiser notre étude des relations entre les flots k-séparables et les coupes en déterminant des coupes améliorantes. De même, l'analyse polyédrale peut s'avérer utile si elle nous permet de définir une génération de coupes conciliante avec la génération de colonnes au sein d'une approche unifiée par Branch and Cut and Price. Elle peut également nous amener à définir des règles de branchements plus performantes.

D'autre part, les limites que nous avons soulevées au niveau des contraintes visant à réduire la symétrie de nos modèles forment un axe de recherche intéressant car il trouve des applications dans de nombreux problèmes discrets. Cette problématique semble à la fois complexe et peu étudiée dans la littérature. Pourtant une solution performante pourrait être à l'origine d'améliorations conséquentes dans de nombreux domaines.

Enfin, notre approche du problème de délai de bout-en-bout témoigne de sa complexité. Cependant, il ne s'agit ici que des prémisses dans le traitement de contraintes de qualité de service avancées. De nombreux travaux viendront sans aucun doute développer des solutions adaptées aux problèmes posés par les opérateurs de réseaux de télécommunication soucieux de proposer de nouveaux services toujours plus performants.

ANNEXE A

PREUVES DE CONVEXITÉ DES FONCTIONS UTILISÉES DANS LA RÉSOLUTION DU k-DCRP

A.1 Preuve de la proposition 4.1

Nous démontrons ici la convexité sur S de la fonction $\theta_a^h : S \to \mathbb{R},\ \theta_a^h(x) = \frac{\lambda x_a^h}{u_a(u_a - x_a)}$ pour tout $a \in A$ et pour tout $h \in 1, \ldots, H$.

Démonstration. Cette fonction θ_a^h dépend uniquement des variables associées à l'arc a. Ainsi, les dérivées partielles par rapport aux variables associées aux autres arcs sont nulles. Nous étudions donc seulement les dérivées partielles par rapport aux variables x_a^i pour $i = 1, \ldots, H$. Pour simplifier les notations, $\bar{x}_a^h$ représente le complément de x_a^h dans x_a :

$$\bar{x}_a^h = \sum_{i \neq h} x_a^i = x_a - x_a^h \tag{A.1}$$

Le calcul des dérivées partielles donne

$$\frac{\partial \theta_a^h}{\partial x_a^i}(x) = \begin{cases} \dfrac{\lambda(u_a - \bar{x}_a^h)}{u_a(u_a - x_a)^2} & \text{si } i = h, \\ \dfrac{\lambda x_a^h}{u_a(u_a - x_a)^2} & \text{sinon.} \end{cases} \tag{A.2}$$

Puis les dérivées partielles secondes sont

$$\frac{\partial^2 \theta_a^h}{\partial x_a^i \partial x_a^j}(x) = \begin{cases} \dfrac{2\lambda x_a^h}{u_a(u_a - x_a)^3} & \text{si } i \neq h \text{ et } j \neq h, \\ \dfrac{2\lambda(u_a - \bar{x}_a^h)}{u_a(u_a - x_a)^3} & \text{si } i = h \text{ et } j = h, \\ \dfrac{\lambda(u_a - \bar{x}_a^h + x_a^h)}{u_a(u_a - x_a)^3} & \text{sinon.} \end{cases} \tag{A.3}$$

Notons $H_a^h(x)$ la matrice hessienne de la fonction θ_a^h. Alors, $\forall x \in S$, $\forall a \in A$, $\forall h = 1, \ldots, H$

$$\begin{aligned} x^T H_a^h(x) x &= \sum_{i=1}^{H}\sum_{j=1}^{H} x_a^i \frac{\partial^2 \theta_a^h}{\partial x_a^i \partial x_a^j}(x) x_a^j \\ &= \frac{\lambda}{u_a(u_a - x_a)^3} \left(2\sum_{i\neq h}\sum_{j\neq h} x_a^h x_a^i x_a^j + 2\sum_{i\neq h}(u_a - \bar{x}_a^h + x_a^h) x_a^h x_a^i \right. \\ &\quad \left. + 2(u_a - \bar{x}_a^h)(x_a^h)^2 \right) \end{aligned}$$

$$\begin{aligned} x^T H_a^h(x) x &= \frac{2\lambda x_a^h}{u_a(u_a - x_a)^3} \left((\bar{x}_a^h)^2 + (u_a - \bar{x}_a^h + x_a^h)\bar{x}_a^h + (u_a - \bar{x}_a^h) x_a^h \right) \\ &= \frac{2\lambda x_a^h}{u_a(u_a - x_a)^3} \left(u_a(\bar{x}_a^h + x_a) \right) \\ &= \frac{2\lambda x_a^h x_a}{(u_a - x_a)^3} \end{aligned} \tag{A.4}$$

Donc $x^T H_a^h x \geqslant 0$, $\forall x \in S$, $\forall a \in A$, $\forall h = 1 \ldots H$ □

A.2 Preuve de la proposition 4.2

Nous démontrons ici la convexité sur S de la fonction $\phi : S \times \mathbb{R}^{*+} \to \mathbb{R},\ \phi(x, \mu) = \mu \ln \sum_{h=1}^{H} e^{\frac{\theta^h(x)}{\mu}}$ pour tout $\mu > 0$.

Démonstration. La preuve est une adaptation de [Xu01]. Le calcul des dérivées partiels du premier et second ordre donne

$$\nabla_x \phi(x,\mu) = \sum_{h=1}^{H} \nu_h(x,\mu) \nabla \theta^h(x) \tag{A.5}$$

$$\begin{aligned} \nabla_x^2 \phi(x,\mu) &= \sum_{h=1}^{H} \left(\nu_h(x,\mu) \nabla^2 \theta^h(x) + \frac{1}{\mu} \nu_h(x,\mu) \nabla \theta^h(x) \nabla \theta^h(x)^T \right) \\ &\quad - \frac{1}{\mu} (\sum_{h=1}^{H} \nu_h(x,\mu) \nabla \theta^h(x)) (\sum_{h=1}^{H} \nu_h(x,\mu) \nabla \theta^h(x))^T \end{aligned} \tag{A.6}$$

$$\text{où} \quad \nu_h(x,\mu) = \frac{\exp(\theta^h(x)/\mu)}{\sum_{h=1}^{H} \exp(\theta^h(x)/\mu)} \in]0,1[, \quad \sum_{h=1}^{H} \nu_h(x,\mu) = 1. \tag{A.7}$$

Notons

$$\begin{aligned} A &= [\nabla \phi^1(x), \ldots, \nabla \phi^H(x)], \\ \Lambda &= \text{diag}(\nu_h(x,\mu)), \\ \Upsilon &= \begin{bmatrix} \nu_1(x,\mu) \\ \vdots \\ \nu_P(x,\mu) \end{bmatrix} [\nu_1(x,\mu), \ldots, \nu_P(x,\mu)], \\ D &= \Lambda - \Upsilon \end{aligned}$$

On a alors

$$\begin{aligned} &\tfrac{1}{\mu} \left(\sum_{h=1}^{H} \nu_h(x,\mu) \nabla \theta^h(x) \nabla \theta^h(x)^T - \left(\sum_{h=1}^{H} \nu_h(x,\mu) \nabla \theta^h(x) \right) \left(\sum_{h=1}^{H} \nu_h(x,\mu) \nabla \theta^h(x) \right)^T \right) \\ &= \tfrac{1}{\mu} (A \Lambda A^T - A \Upsilon A^T) = \tfrac{1}{\mu} A D A^T \end{aligned} \tag{A.8}$$

Par le théorème de Gershgorin, les valeurs propres de D appartiennent à l'union U_H de H cercles.

$$U_H = \bigcup_{h=1}^{H} \{ z \in \mathbb{C} \ : \ |z - d_{hh}| \leqslant R_h(D) \} \tag{A.9}$$

où

$$\begin{aligned} d_{hh} &= \nu_h(x,\mu) - \nu_h^2(x,\mu) \qquad \forall h = 1, \ldots, H \\ R_h(D) &= \sum_{i \neq h} \nu_h(x,\mu) \nu_i(x,\mu) \qquad \forall h = 1, \ldots, H \end{aligned}$$

Alors

$$\begin{aligned} R_h(D) &= \sum_{i=1}^{H} \nu_h(x,\mu)\nu_i(x,\mu) - \nu_h^2(x,\mu) \\ &= \nu_h(x,\mu)\sum_{i=1}^{H} \nu_i(x,\mu) - \nu_h^2(x,\mu) \\ &= \nu_h(x,\mu) - \nu_h^2(x,\mu) \\ &= d_{hh} \end{aligned} \tag{A.10}$$

Ainsi, chaque valeur propre de $\Lambda - \Upsilon$ est positive et la matrice $\frac{1}{\mu} ADA^T$ est semi-définie positive.

D'autre part, on a

$$\begin{aligned} x^T \left(\sum_{h=1}^{H} \nu_h(x,\mu) \nabla^2 \theta^h(x) \right) x &= \sum_{h=1}^{H} \nu_h(x,\mu) \sum_{a \in A} x^T \nabla^2 \theta_a^h(x) x \\ &= \sum_{h=1}^{H} \nu_h(x,\mu) \sum_{a \in A} \frac{2\lambda x_a^h x_a}{(u_a - x_a)^3} \\ &= 2\lambda \sum_{a \in A} \left(\frac{x_a}{(u_a - x_a)^3} \sum_{h=1}^{H} \nu_h(x,\mu) x_a^h \right) \end{aligned} \tag{A.11}$$

Considérons un vecteur $x \in S$ tel que $x^T \left(\sum_{h=1}^{H} \nu_h(x,\mu) \nabla^2 \theta^h(x) \right) x = 0$.

Alors

$$\frac{x_a}{(u_a - x_a)^3} \sum_{h=1}^{H} \nu_h(x,\mu) x_a^h = 0 \quad \forall a \in A \tag{A.12}$$

Comme $\nu_h(x,\mu) > 0$ pour tout h, on en déduit que $x = 0$.

Ainsi $\sum_{h=1}^{H} \nu_h(x,\mu) \nabla^2 \theta^h(x)$ est défini positif et donc $\nabla_x^2 \phi(x,\mu)$ l'est aussi. □

RÉFÉRENCES BIBLIOGRAPHIQUES

[AdC03] Filipe ALVELOS et José Manuel Vasconcelos Valério de CARVALHO : Comparing Branch-and-price Algorithms for the Unsplittable Multicommodity Flow Problem. *In Proceedings of the International Network Optimization Conference*, pages 7–12, 2003.

[AMO93] Ravindra K. AHUJA, Thomas L. MAGNANTI et James B. ORLIN : *Network Flows - Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[AR02] Alper ATAMTÜRK et Deepak RAJAN : On splittable and unsplittable capacitated network design arc-set polyhedra. *Mathematical Programming*, 92:315–333, 2002.

[AW00] Jeffrey A. APPLEGET et R. Kevin WOOD : Explicit-constraint branching for solving mixed-integer programs. *In* Manuel LAGUNA et José Luis GONZÁLEZ VELARDE, éditeurs : *Computing tools for modeling, optimization, and simulation : interfaces in computer science and operations research*, volume 12 de *Operations research / computer science interfaces series*, pages 245–261. Kluwer Academic, Boston, 2000.

[BAG03] Walid BEN-AMEUR et Eric GOURDIN : Internet routing and related topology issues. *SIAM J. Discret. Math.*, 17(1):18–49, 2003.

[BAMLG01] W. BEN-AMEUR, N. MICHEL, B. LIAU et E. GOURDIN : Routing strategies for IP networks. *Telektronikk*, pages 145–158, March 2001.

[BaO06] Walid BEN-AMEUR et Adam OUOROU : Mathematical models of the delay constrained routing problem. *Algorithmic Operations Research*, 1(2), 2006.

[Bar96] Francisco BARAHONA : Network design using cut inequalities. *SIAM J. on Optimization*, 6(3):823–837, 1996.

[BDL+01a] Ayan BANERJEE, John DRAKE, Jonathan P. LANG, Brad TURNER, Daniel O. AWDUCHE, Lou BERGER, Kireeti KOMPELLA et Yakov REKHTER : Generalized Multiprotocol Label Switching : an Overview of Signaling Enhancements and Recovery Techniques. *In IEEE Communications Magazine*, volume 39-7, pages 144–151, 2001.

[BDL+01b] Ayan BANERJEE, John DRAKE, Jonathan P. LANG, Brad TURNER, Kireeti KOMPELLA et Yakov REKHTER : Generalized Multiprotocol Label Switching : an Overview of Routing and Management Enhancements. *In IEEE Communications Magazine*, volume 39-1, pages 144–150, 2001.

[Ben62] J. F. BENDERS : Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252, 1962.

[BG92] Dimitri BERTSEKAS et Robert GALLAGER : *Data networks (2nd ed.)*. Prentice-Hall, Inc., 1992.

[BHV00] Cynthia BARNHART, Christopher A. HANE et Pamela H. VANCE : Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. *Operations Research*, 48(2):318–326, 2000.

[Bie02] Daniel BIENSTOCK : *Potential function methods for approximately solving linear programming problems, Theory and Practice*. Kluwer Academics Publishers, Boston, 2002.

[BJN+98] Cynthia BARNHART, Ellis L. JOHNSON, George L. NEMHAUSER, Martin W.P. SAVELSBERGH et Pamela H. VANCE : Branch-and-price : column generation for solving huge integer programs. *Operations Research*, 46:316–329, 1998.

[BKP03] Sergio BEKER, Daniel KOFMAN et Nicolas PUECH : Off-Line MPLS Layout Design and Reconfiguration : Reducing Complexity under Dynamic Traffic Conditions. *In INOC Conference*, pages 61–66, 2003.

[BKS05] Georg BAIER, Ekkehard KÖHLER et Martin SKUTELLA : The k-splittable flow problem. *Algorithmica*, 42(3-4):231–248, 2005.

[BM00] D. BIENSTOCK et G. MURATORE : Strong inequalities for capacitated survivable network design problems. *Mathematical Programming*, 89(1):127–147, 2000.

[BMW89] A. BALAKRISHNAN, T.L. MAGNANTI et R.T. WONG : A dual-ascent procedure for large-scale uncapacitated network design. *Operations Research*, 37(5):716–740, 1989.

[BOdKK01] James E. BURNS, Teunis J. OTT, Johan M. de KOCK et Anthony E. KRZESINSKI : Path Selection and Bandwidth Allocation in MPLS Networks : a Non-Linear Programming Approach. *In ITCom Conference*, 2001.

[BR02] Daniel BIENSTOCK et Olga RASKINA : Asymptotic analysis of the flow deviation method for the maximum concurrent flow problem. *Mathematical Programming*, 91(3):479–492, February 2002.

[BT97] Dimitris BERTSIMAS et John TSITSIKLIS : *Introduction to Linear Optimization*. Athena Scientific, Belmont, MA, 1997.

[CB00] Anna CHARNY et Jean-Yves Le BOUDEC : Delay bounds in a network with aggregate scheduling. *In QofIS '00 : Proceedings of the First COST 263 International Workshop on Quality of Future Internet Services*, pages 1–13, London, UK, 2000. Springer-Verlag.

[CdMD+02] Didier COLLE, Sophie de MAESSCHALCK, Chris DEVELDER, Pim Van HEUVEN, Adelbert GROEBBENS, Jan CHEYNS, Ilse LIEVENS, Mario PICKAVET, Paul LAGASSE et Piet DEMEESTER : Data-Centric Optical Networks and Their Survivability. *In IEEE Journal on Selected Areas in Communications*, volume 20-1, pages 6–20, 2002.

[Dak65] Robert J. DAKIN : A tree-search algorithm for mixed integer programming problems. *The Computer Journal*, 8(3):250–254, 1965.

[DDSS95] J. DESROSIERS, Y. DUMAS, M.M. SOLOMON et F. SOUMIS : Time constrained routing and scheduling. *In* M.O. BALL, T.L. MAGNANTI, C.L. MONMA et G.L. NEMHAUSER, éditeurs : *Network Routing*, volume 8 de *Handbooks in Operations Research and Management Science*, pages 35–139. Elsevier Science Publishers B.V., Amsterdam, The Netherlands, 1995.

[DGA+99] Piet DEMEESTER, Michael GRYSEELS, Achim AUTENRIETH, Carlo BRIANZA, Laura CASTAGNA, Giulio SIGNORELLI, Roberto CLEMENTE, Mauro RAVERA, Andrej JAJSZCZYK, Dariusz JANUKOWICZ, Kristof Van DOORSELAERE et Yohnosuke HARADA : Resilience in Multilayer Networks. *In IEEE Communications Magazine*, volume 37-8, pages 70–76, 1999.

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

[DM07] Christophe DUHAMEL et Philippe MAHEY : Multicommodity flow problems with a bounded number of paths : A flow deviation approach. *Networks*, 49(1):80–89, 2007.

[dMCG+02] Sophie de MAESSCHALK, Didier COLLE, Adelbert GROEBBENS, Chris DEVELDER, Ilse LIEVENS, Paul LAGASSE, Mario PICKAVET, Piet DEMEESTER, Fausto SALUTA et Marco QUAGLIOTTI : Intelligent Optical Networking for Multilayer Survivability. *In IEEE Communications Magazine*, volume 40-1, pages 42–49, 2002.

[dMVDH99] Olivier du MERLE, Daniel VILLENEUVE, Jacques DESROSIERS et Pierre HANSEN : Stabilized column generation. *Discrete Math.*, 194(1-3):229–237, 1999.

[DSD84] Jacques DESROSIERS, FranÃ§ois SOUMIS et Martin DESROCHERS : Routing with time windows by column generation. *Networks*, 14(4):545–565, 1984.

[dSVG01] Amaro F. de SOUSA, Rui VALADAS et Luis GOUVEIA : Dimensioning ATM Networks Using 2-Layer Hierarchical Virtual Path Layouts. *In 9th International Conference on Telecommunication Systems*, 2001.

[DW60] George B. DANTZIG et Philip WOLFE : Decomposition principle for linear programs. *Operations Research*, 8:101–111, 1960.

[EJLW01] Anwar ELWALID, Cheng JIN, Steven LOW et Indra WIDJAJA : MATE : MPLS Adaptive Traffic Engineering. *In IEEE Infocom*, pages 1300–1309, 2001.

[FF58] Lester R. FORD et Delbert R. FULKERSON : A suggested computation for maximal multicommodity network flow. *Management Science*, 5:97–101, 1958.

[FGK73] Luigi FRATTA, Mario GERLA et Leonard KLEINROCK : The flow deviation method : An approach to store-and-forward communication network design. *Networks*, 3:97–133, 1973.

[FLL+06] Ricardo FUKASAWA, Humberto LONGO, Jens LYSGAARD, Marcus Poggi de ARAGÃO, Marcelo REIS, Eduardo UCHOA et Renato F. WERNECK : Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Math. Program.*, 106(3):491–511, 2006.

[FT00] Bernard FORTZ et Mikkel THORUP : Internet traffic engineering by optimizing OSPF weights. *In INFOCOM (2)*, pages 519–528, 2000.

[FW56] M. FRANK et P. WOLFE : An algorithm for quadratic programming. *Naval Research Logistics Quarterly 3*, pages 95–110, 1956.

[GG61] P. GILMORE et R. GOMORY : A linear programming approach to the cutting stock problem. *Operations Research*, 9:849–859, 1961.

[GG63] P. GILMORE et R. GOMORY : A linear programming approach to the cutting stock problem - part II. *Operations Research*, 11:863–888, 1963.

[GK87] Monique GUIGNARD et Siwhan KIM : Lagrangean decomposition : A model yielding stronger lagrangean bounds. *Math. Program.*, 39(2):215–228, 1987.

[GK94] Michael D. GRIGORIADIS et Leonid G. KHACHIYAN : Fast approximation schemes for convex programs with many blocks and coupling constraints. *SIAM Journal on Optimization*, 4(1):86–107, février 1994.

[GLON04] Eric GOURDIN, Bernard LIAU, Adam OUOROU et Dritan NACE : Optimisation des réseaux de télécommunication. *Bulletin de la Société Française de Recherche Opérationnelle et d'Aide à la Décision*, 12:8–12, 2004.

[GLS81] Martin GRÖTSCHEL, László LOVÁSZ et Alexander SCHRIJVER : The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

[GM95] M. GONDRAN et M. MINOUX : *Graphes et algorithmes*. Eyrolles, Paris, 3e édition, 1995.

[Gou01] E. GOURDIN : Optimizing Internet networks. *OR/MS Today*, pages 46–49, April 2001.

[Gro04] W. D. GROVER : *Mesh Based Survivable Transport Networks : Options and Strategies for optical, MPLS, SONET and ATM networking*. Prentice Hall PTR, 2004.

[GV02] J.-L. GOFFIN et J.-P. VIAL : Convex nondifferentiable optimization : a survey focussed on the analytic center cutting plane method. *Optimization Methods and Software*, 17(5):805–867, 2002.

[IMG98] Rainer R. IRASCHKO, M. H. MACGREGOR et Wayne D. GROVER : Optimal capacity placement for path restoration in stm or atm mesh-survivable networks. *IEEE/ACM Trans. Netw.*, 6(3):325–336, 1998.

[ISS03] Paola IOVANNA, Roberto SABELLA et Marina SETTEMBRE : A Traffic Engineering System for Multilayer Networks Based on the GMPLS Paradigm. *In IEEE Network*, volume 17-2, pages 28–37, 2003.

[Kar84] Narendra KARMARKAR : A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, December 1984.

[Ken78] J.L. KENNINGTON : A survey of linear cost multicommodity network flows. *Operations Research*, 26(2):209–236, 1978.

[Kha79] Leonid G. KHACHIYAN : A polynomial algorithm in linear programming. *Soviet Mathematics Doklady*, 20:191–194, 1979.

[Kle64] L. KLEINROCK : *Communication Nets : Stochastic Message Flow and Delay*. McGraw-Hill, New-York, 1964. Out of Print. Reprinted by Dover Publications, 1972.

[Kle96] Jon M. KLEINBERG : *Approximation algorithms for disjoint paths problems*. Thèse de doctorat, Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science, 1996.

[Kol05] Stavros KOLLIOPOULOS : Minimum-cost single-source 2-splittable flow. *Information Processing Letters*, 94:15–18, 2005.

[KR05] K. KOMPELLA et Y. REKHTER : Label Switched Paths (LSP) Hierarchy with Generalized Multi-Protocol Label Switching (GMPLS) Traffic Engineering (TE). RFC 4206 (Proposed Standard), octobre 2005.

[KS06] Ronald KOCH et Ines SPENKE : Complexity and approximability of k-splittable flows. *Theor. Comput. Sci.*, 369(1-3):338–347, 2006.

[Lap00] Philip A. LAPLANTE, éditeur. *Dictionary of Computer Science, Engineering, and Technology*. CRC Press, 2000.

[Las70] Leon S. LASDON : *Optimization Theory for Large Systems*. MacMillan, 1970.

[LC00] A. LISSER et P. CHARDAIRE : Minimum cost multicommodity flow. *In Handbook of applied optimisation*, pages 97–100. Resende & Parados éditeurs, 2000.

[LD60] A. LAND et A. DOIG : An automatic method of solving discrete programming problems. *Econometrika*, 28(3):497–520, 1960.

[LD05] Marco E. LÜBBECKE et Jacques DESROSIERS : Selected topics in column generation. *Operations Research*, 53(6):1007–1023, 2005.

[Liu02] Kevin H. LIU : *IP over WDM*. J. Wiley & sons, 2002.

[LNN95] Claude LEMARÉCHAL, Arkadii NEMIROVSKII et Yurii NESTEROV : New variants of bundle methods. *Mathematical Programming*, 69(1):111–147, 1995.

[Mah94] Ali Ridha MAHJOUB : Two-edge connected spanning subgraphs and polyhedra. *Math. Program.*, 64(2):199–208, 1994.

[Mah02] Philippe MAHEY : Decomposition methods for mathematical programming. *In Handbook of Applied Optimization*, chapitre 7. P. Pardalos and M.G.C. Resende éditeurs, oxford university press édition, 2002.

[Min83] Michel MINOUX : *Programmation Mathématique : Théorie et algorithmes*, volume 1 de *Collection Technique et Scientifique des Télécommunications*. Dunod, Paris, France, 1983.

[Min89] M. MINOUX : Network synthesis and optimum network design problems : Models, solution methods and applications. *Networks*, 19:313–360, 1989.

[MM00] Gurusamy MOHAN et C. Siva Ram MURTHY : Lightpath Restoration in WDM Optical Networks. *In IEEE Network*, volume 14-6, pages 24–32, 2000.

[MPRD99] E.Q.V. MARTINS, M.M.B. PASCOAL, D.M.L.D. RASTEIRO et J.L.E. DOS SANTOS : The optimal path problem. *Investigação Operacional*, 19:43–60, 1999.

[MS06] Maren MARTENS et Martin SKUTELLA : Flows on few paths : Algorithms and lower bounds. *Networks*, 48(2):68–76, 2006.

[MTUP04] Vahab S. MIRROKNI, Marina THOTTAN, Huseyin UZUNALIOGLU et Sanjoy PAUL : A Simple Polynomial Time Framework To Reduced Path Decomposition in Multi-Path Routing. *In proceedings of INFOCOM 2004*, pages 739–749, 2004.

[MV04] Jean François MAURRAS et Sonia VANIER : Network synthesis under survivability constraints. *4OR*, 2(1):53–67, 2004.

[OMV00] A. OUOROU, P. MAHEY et J.-Ph. VIAL : A survey of algorithms for convex multicommodity flow problems. *Management Science*, 46(1):126–147, 2000.

[PKGK03] Nicolas PUECH, Mohamed KOUBAA, Maurice GAGNAIRE et Josué KURI : Models for Path Protection in WDM Optical Mesh Networks. *In INOC Conference*, pages 472–477, 2003.

[PSvT95] Serge A. PLOTKIN, David B. SHMOYS et &#201 ;va TARDOS : Fast approximation algorithms for fractional packing and covering problems. *Math. Oper. Res.*, 20(2):257–301, 1995.

[Puj00] Guy PUJOLLE : *Les Réseaux, 3ème édition*. Editions Eyrolles, 2000.

[QMJ03] Yang QIN, Lorne MASON et Ke JIA : Study on a Joint Multiple Layer Restoration Scheme for IP over WDM Networks. *In IEEE Network*, volume 17-2, pages 43–48, 2003.

[RF81] David M. RYAN et Brian A. FOSTER : An integer programming approach to scheduling. *In* Anthony WREN, éditeur : *Computer Scheduling of Public Transport : Urban Passenger Vehicle and Crew Scheduling*, pages 269–280. North-Holland Publishing Company, 1981.

[RLA04] B. RAJAGOPALAN, J. LUCIANI et D. AWDUCHE : IP over Optical Networks : A Framework. RFC 3717 (Informational), mars 2004.

[RM99] S. RAMAMURTHY et Biswanath MUKHERJEE : Survivable wdm mesh networks, part 1 - protection. *In INFOCOM*, pages 744–751, 1999.

[RVC01] E. ROSEN, A. VISWANATHAN et R. CALLON : Multiprotocol Label Switching Architecture. RFC 3031 (Proposed Standard), janvier 2001.

[SM90] Farhad SHAHROKHI et D. W. MATULA : The maximum concurrent flow problem. *J. ACM*, 37(2):318–334, 1990.

[SRM02] Laxman SAHASRABUDDHE, S. RAMAMURTHY et Biswanath MUKHERJEE : Fault Management in IP-Over-WDM Networks : WDM Protection Versus IP Restoration. *In IEEE Journal on Selected Areas in Communications*, volume 20-1, pages 21–33, 2002.

[SS01] Hanif D. SHERALI et J. Cole SMITH : Improving discrete model representations via symmetry considerations. *Management Science*, 47(10):1396–1407, 2001.

[SS03] Galen H. SASAKI et Ching-Fong SU : The Interface between IP and WDM and its Effect on the Cost of Survivability. *In IEEE Communications Magazine*, volume 41-1, pages 74–79, 2003.

[TDM06] Jérôme TRUFFOT, Christophe DUHAMEL et Philippe MAHEY : k-Splittable Delay Constrained Routing Problem : A Branch and Price Approach. Rapport technique RR-06-08, LIMOS, Aubière, France, 2006. en révision dans Networks.

[Van05] François VANDERBECK : Implementing mixed integer column generation. *In* G. DESAULNIERS, J. DESROSIERS et M. M. SOLOMON, éditeurs : *Column Generation*, pages 331–358. Springer-Verlag, Boston, MA, 2005.

[Vat04] Bénédicte VATINLEN : *Optimisation du routage dans les réseaux de télécommunications avec prise en compte de la qualité de service*. Thèse de doctorat, LIP6 - université Paris 6 et Bouygues Telecom, 9 2004.

[VW96] François VANDERBECK et Laurence A. WOLSEY : An exact algorithm for ip column generation. *Operations Research Letters*, 19(4):151–159, 1996.

[Xu01] Song XU : Smoothing method for minimax problems. *Comput. Optim. Appl.*, 20(3):267–279, 2001.

Résumé :
L'augmentation constante des débits de transmission des données a fait évoluer les réseaux IP vers de nouveaux services. La tendance actuelle est à la convergence des réseaux où chaque opérateur veut offrir une multitude de services à ces clients. Toutefois, la maîtrise de la qualité de service reste encore aujourd'hui un défi majeur pour la recherche. Dans ce contexte, ces travaux de thèse étudient l'influence des nouveaux protocoles dans la conception de réseaux haut débit tolérants aux pannes.

Dans un premier temps, nous nous sommes intéressés aux spécificités du routage dans les réseaux MPLS (Multi-Protocol Label Switching). Dans l'approche classique du routage IP, chaque paquet est indépendant et les décisions de routage se prennent à chaque noeud intermédiaire du réseau. Cependant, l'engorgement des tables de routage et la gestion de la qualité de service font partie des multiples raisons qui ont motivé l'émergence de protocoles favorisant les mécanismes de commutation. C'est le cas du protocole MPLS qui prévoit que les décisions de routage soient prises au niveau du noeuds d'entrée du réseau. Ainsi, toute une partie du flux est associée à un chemin dans le réseau. Pour limiter la taille des tables de commutation, il est alors nécessaire de limiter le nombre de ces chemins. Cette contrainte forte sur le support des flux change considérablement la complexité des problèmes de routage. Notre contribution porte principalement sur la modélisation des problèmes de flots k-séparables par des programmes linéaires en nombres entiers adaptés à une résolution exacte. La nature du problème nous a amenés à considérer une généralisation de l'application de la méthode du Branch and Price au problème du multiflot monorouté.

D'autre part, nous nous sommes intéressés à la contrainte de délai de bout-en-bout. Cette contrainte est peu étudiée dans la littérature car elle revêt un caractère non linéaire et entier très complexe. Cependant, nos modèles de flots k-séparables offrent des possibilités intéressantes pour gérer le délai de bout-en-bout sur les chemins actifs. Le problème de minimisation du délai de bout-en-bout maximal est alors résolu par combinaison de plusieurs méthodes.

Abstract :
The rise of modern broadband communication networks made IP networks evolve toward new services. The current trend is the convergence of networks where each operator wants to offer a multitude of services to his customers. However, the control over the quality of service is still a major challenge for research. In this context, these thesis work studies the influence of the new protocols in the design of high-speed fault-tolerant networks.

As a first step, we were interested in the specificities of routing in MPLS networks (MultiProtocol Label Switching). In the traditional approach of IP routing, each packet is independent and routing decisions are made at each intermediate node of the network. However, the bottleneck of traffic routing tables and the management of the quality of service are among the reasons that led to the emergence of protocols supporting mechanisms for commutation. This is the case of the MPLS protocol which provides that routing decisions are taken at the input nodes of the network. Thus, any part of the stream is associated with a path in the network. To limit the size of the commutation tables, it is necessary to limit the number of those paths. This strong constraint on the design of the flow considerably changes the complexity of routing problems. Our contribution focuses on the modeling of k-splittable flow problems by mixed integer linear programs adapted to an exact resolution. The nature of the problem led us to consider a broader use of the Branch and Price method with the unsplittable multicommodity flow problem.

On the other hand, we were interested in the end-to-end delay constraint. This constraint is not studied much in the literature because it takes on very complex non-linear and integer characteristics. However, our models of k-splittable flow offer interesting possibilities to manage the end-to-end delay on the active paths. The problem of minimizing the maximum end-to-end delay is then solved by a combination of several methods.