UNIVERSITE de TOULOUSE III - PAUL SABATIER
UFR Mathématiques, Informatique, Gestion

THESE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITE DE TOULOUSE

Délivré par l'Université Toulouse III - Paul Sabatier
Discipline ou spécialité : Systèmes Informatiques

Présentée et soutenue par Cédric FORTUNY
Le 30 juin 2008
Titre :

Estimation du Trafic, Planification et Optimisation des ressources pour l'ingénierie des réseaux IP/MPLS

JURY

Christian BES (professeur)
Bernard FORTZ (chargé de cours)
Philippe MAHEY (professeur)
Jean-Marie GARCIA (directeur de recherche)
Olivier BRUN (chargé de recherche)

Ecole doctorale : EDSYS
Unité de recherche : LAAS-CNRS groupe MRS
Directeur de Thèse : Jean-Marie GARCIA

Remerciements

Ce mémoire est l'aboutissement de trois années d'études effectuées au Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS (LAAS-CNRS) au sein du groupe de Modélisation et contrôle des Réseaux et des Signaux (MRS).

Je tiens à exprimer toute ma reconnaissance et toute ma gratitude à Malik Ghallab, directeur du laboratoire lors de mon arrivée, et Raja Chatila, directeur actuel du laboratoire, pour m'avoir permis de réaliser l'ensemble de ces travaux dans d'excellentes conditions. Il est important de signaler la qualité des moyens mis à disposition dans ce laboratoire.

Je tiens également à adresser toute ma gratitude et mes remerciements à Jean-Marie Garcia, directeur de recherche CNRS, pour avoir accepter de diriger l'ensemble de mes recherches. Je lui serais toujours reconnaissant de m'avoir permis de m'essayer au monde la recherche en acceptant ma candidature de stage. Je le remercie plus particulièrement de m'avoir donné la possibilité de réaliser l'ensemble de ces travaux ainsi que pour ses nombreux conseils durant leurs réalisation. Notre communication a quelques fois été mouvementée, cependant cela nous a permis d'avancer et de nous rapprocher.

Je suis particulièrement reconnaissant envers Bernard Fortz, chargé de cours à l'université libre de Bruxelle, et Philippe Mahey, directeur de l'ISIMA, qui ont accepté d'être les rapporteurs de ma thèse. Mes remerciements portent plus principalement sur la qualité de leurs analyses ainsi que sur la pertinence de leurs remarques. Cependant, je tiens à leur exprimer toute ma gratitude pour leur présence et leur rôle lors de la soutenance de cette thèse.

Je remercie également Prosper Chemouil, directeur de la recherche et du développement en réseau et télécom chez Orange Labs, Christian Bes, professeur à l'université Paul Sabatier, et Olivier Brun, chercheur au LAAS-CNRS, en leurs qualités de membres du jury. Merci pour votre présence, vos questions, vos analyses et vos remarques. Cette soutenance fut un moment mémorable en grande partie grâce à vous.

Tous les travaux réalisés l'ont été en équipe et plus particulièrement l'intégration de ces travaux dans le logiciel NEST de QoS Design. Je tiens donc à dire un grand merci à l'ensemble des collègues avec qui j'ai pu travailler. Les plus anciens qui ne sont plus là :

- François, mon développeur JAVA préféré,
- Cyril et Eric, que de crises de rire ensemble,
- Hassan et Bernard, toujours au rendez-vous de 12h avant d'aller manger,

les anciens fidèles au poste :

- Charly et Anouar, que de discussions ensemble, que ce soit la religion, les complots ou le monde en général, ça me manquera,
- David et Fred, des discussions sans queue ni tête qui me manqueront beaucoup aussi,

et les nouveaux :

- Laurent, s'ils sont tous comme toi a Tahiti, j'y vais direct ! !
- Zied, la relève, je te souhaite tout le courage dont tu auras besoin.

Les 4 ans passés au LAAS n'auraient pas été les mêmes sans votre présence. Je me souviendrais longtemps des moments passés ensemble, les repas en terrasse, les pauses café où l'on refaisait le monde et QoS Design. Bref, que des bons moments qui resterons gravés pour longtemps.

Mes plus grands remerciements vont à Olivier Brun, chercheur au LAAS-CNRS, qui est devenu au fur et à mesure des années un ami cher source de bons conseils. Qu'il est agréable de travailler avec toi Olivier. Ne change rien surtout et si une place se dessine pour travailler à tes côtés, ce serait avec un grand, très grand, plaisir que je l'accepterais. Une profonde pensée à Delphine et à tes deux enfants Hugo et Louise. Je vous souhaite à tous les quatre tout le bonheur que vous méritez.

Je ne saurais oublier l'ensemble des membres du LAAS que j'ai pu rencontrer durant ces quatres années.

Pour terminer, une profonde pensée à tous mes amis toulousains ou d'ailleurs. Cette soirée organisée dans mon dos m'a montré votre amitié et m'a beaucoup touché. Merci a vous tous de m'avoir permis ces évasions régulières le week-end, ces apéros toujours joyeux, ces barbecues même à deux jours de la soutenance. Ces quatres années sont aussi les vôtres.

J'en profite au passage pour souhaiter un bon anniversaire à ma colocatrice, amie et presque soeur qui se reconnaîtra en ce jour ! ! Je ne peux donc pas finir cette page sans dire merci également à Sandrine, ma seconde colocatrice. Vous êtes un peu comme mes petites et grandes soeurs à la fois.

Comme il risque d'y avoir des jaloux, il ne me reste plus qu'à citer l'ensemble des amis : Tedd, Marine, Sandrine, Guillaume, Laure, Helène, Olive, Jess, Ansérine, Flo, Luc, Ophélie, Kévin, Emilie, Adèle, Beb, Manu, Sacha, Thibault, Aurélien, Joy, Florine, Jérome, Vanessa, Jim, Léo, Franck, Llomgui, David, Vanessa . . .et Célinou, . . .J'en oublie certainement, j'espère qu'il me le pardonnerons.

On dit qu'on garde toujours le meilleur pour la fin, alors suivant la tradition, je termine ces remerciements en adressant mes plus sincères remerciements à papa, maman, Mathieu et Laetitia. Même si vous n'êtes pas souvent intervenus dans mes études, une fois a suffit et pour cela je vous serais éternellement reconnaissant. Et dire que j'ai failli ne jamais écrire cette page (et le mémoire d'ailleurs). Et merci, merci beaucoup, pour ce coup d'éclat, cette surprise surprenante qui a suivi ma soutenance en marquant la fin de ces années passées ici. De gros bisous à vous quatre.

Cédric Fortuny. Le 4 juillet 2008.

TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES TABLEAUX

LIST OF ALGORITHMES

Introduction

Avec aujourd'hui plus de 800 millions d'utilisateurs dans le monde, l'Internet a connu un succès fulgurant. Ce succès planétaire, obtenu en moins de dix ans, est le résultat d'une succession de percées, à commencer bien sûr par le développement incroyable du World Wide Web, suivi par l'adoption par le monde professionnel des applications Internet telles que le courrier électronique et, plus récemment, par le déploiement des logiciels de partage de fichiers pair à pair. Le trafic Internet continue à croître de façon très importante, d'une part à cause d'un nombre d'utilisateurs toujours plus élevé, et d'autre part à cause de la part croissante du trafic pair à pair. Parallèlement à cette augmentation soutenue du trafic liée aux services « traditionnels », l'Internet a connu plusieurs évolutions majeures ces dernières années.

Evolution des réseaux IP

La première de ces évolutions concerne sa transformation en une architecture de communication globale supportant également de nouveaux services « temps-réel », tels que la voix sur IP ou la vidéo. Ces services ayant des contraintes de qualité de service beaucoup plus strictes que les services traditionnels évoqués ci-dessus, les nouveaux réseaux IP intègrent un ensemble de mécanismes issus de l'architecture DiffServ pour fournir une qualité de service différenciée aux flux en fonction de leur classe de service. Le développement attendu de ces nouveaux services « temps-réel » est susceptible de bouleverser radicalement les réseaux actuels, en engendrant de nouveaux usages, en attirant beaucoup plus d'utilisateurs et ainsi en accroissant de plusieurs ordres de grandeur les volumes de trafic qu'ils doivent supporter.

Un autre fait marquant de l'évolution des réseaux au cours de ces dernières années est la multiplication des réseaux d'accès, filaires et surtout radios (GPRS, UMTS, WiFi, WiMAX, etc.). Cette évolution conduit vers une réelle convergence des services fournis aux utilisateurs, quels que soient les modes d'accès. Cette convergence va se traduire pour les utilisateurs par la possibilité de toujours accéder à une multiplicité de services Internet, quel que soit le média utilisé et où qu'ils se trouvent. L'Internet de demain sera ainsi une interconnexion de réseaux d'accès hétérogènes permettant d'accéder, via des

cœurs de réseaux IP/MPLS, à un grand nombre de services « universels ».

Toutes ces évolutions conduisent à une multiplication des services offerts par le réseau et à une croissance sans précédent du nombre d'utilisateurs et des volumes de trafics qu'ils génèrent. Dans une société où l'information et la communication ont pris une telle importance, l'interruption des services offerts par le réseau, ou même une dégradation significative de la qualité de service, sont de moins en moins acceptables. La sécurisation des réseaux et le respect des nouvelles exigences de qualité de service sont ainsi des enjeux majeurs.

Nécessité d'une nouvelle ingénierie des réseaux IP

Ces nouvelles exigences de qualité de service, l'augmentation continue du trafic Internet et les impératifs de plus en plus forts de sécurisation des réseaux imposent aux fournisseurs d'accès Internet (FAI) et aux opérateurs une adaptation permanente de leurs infrastructures de communication. Cependant, le contexte concurrentiel qui induit des marges bénéficiaires réduites, ne permet plus d'améliorer les performances d'un réseau par un surdimensionnement excessif des équipements.

Une solution consiste à avoir un suivi plus régulier et plus fin du réseau pour mettre en œuvre des techniques d'ingénierie de trafic. Ces techniques ont pour objectif principal d'éviter les phénomènes de congestion du trafic et les dégradations du service qui en résultent. A notre avis, une telle approche de l'ingénierie des réseaux doit alors reposer sur trois axes essentiels :

- **Supervision du réseau :** il est illusoire de vouloir mettre en œuvre une ingénierie de trafic efficace, sans une connaissance approfondie du réseau et des trafics qu'il achemine. Si l'observation par métrologie du trafic sur les liens est pertinente, elle n'est pas suffisante. Il faut développer une vision globale de la matrice de trafic, et de son évolution sur plusieurs échelles de temps.

- **Planification du réseau :** les objectifs essentiels sont de dimensionner à moindre coût les équipements du réseau et d'optimiser le routage des flux. Pour être utilisé en situation opérationnelle, un outil de planification doit permettre de résoudre ces problèmes en tenant compte de l'ensemble des contraintes matérielles ou techniques auxquelles est confronté le gestionnaire d'infrastructures, mais aussi de la qualité de service cible et du niveau de résilience recherché.
- **Adaptation dynamique du réseau :** Etant donnée la forte variabilité du trafic Internet à court terme, liée notamment à celle de l'activité des utilisateurs, il est impossible d'éviter des dégradations ponctuelles du service offert sans mettre en œuvre des techniques de reconfiguration dynamique des ressources. L'augmentation du degré d'autoadaptativité des réseaux aura pour conséquence l'amélioration de la qualité de service, une meilleure utilisation des ressources globales et une planification beaucoup plus progressive. Les problématiques visées concernent notamment le routage dynamique et la reconfiguration dynamique des LSP dans les réseaux MPLS.

Contributions

L'ensemble de nos contributions porte sur la supervision et la planification des réseaux et a pour caractéristiques communes de tenir compte des contraintes :

- **technologiques** telles que les groupes de liens à risques partagés (SRLG), les cartes de communication (PIC), etc. . .,
- **opérationnelles** par une vision souvent incrémentale des solutions proposées, et
- **financières** par l'intégration de modèles économiques réalistes et utilisés par les opérateurs,

et d'être intégré dans le logiciel NEST dont l'ambition est d'être l'outil de référence de cette nouvelle approche d'ingénierie des réseaux IP.

L'organisation du présent mémoire est la suivante :

- Le chapitre 1 approfondit le fonctionnement, les évolutions et les limitations des réseaux IP depuis leur apparition. Une définition plus précise d'une nouvelle approche d'ingénierie est ensuite proposée.

- Le chapitre 2 traitera du problème d'estimation des matrices de trafics. La connaissance des volumes de trafics échangés sur un réseau est un préalable essentiel à son ingénierie. Estimer les trafics à partir de mesures de charges sur les liens (SNMP [37]) est une alternative à une métrologie travaillant au niveau des flots, cette dernière étant trop lourde à supporter pour un réseau. En introduisant une hypothèse sur la répartition des trafics au cours du temps, nous proposons un algorithme d'estimation fournissant un niveau de précision acceptable pour un opérateur, en des temps de calcul raisonnables.

- Nous considérons le problème d'optimisation des métriques de routage IP dans le chapitre 3. Utilisant un voisinage de taille réduite, une nouvelle heuristique aux temps de calcul faibles permet d'obtenir des solutions incrémentales de bonne qualité. Différentes extensions sont également proposées pour permettre une solution de routage robuste et/ou de qualité proche du routage optimal.

- Le protocole MPLS [55] devenant incontournable de part son utilisation massive, nous étudions, dans le chapitre 4, l'intérêt d'optimiser les métriques IP dans un cœur de réseau IP/MPLS. Nous montrerons que l'utilisation des ressources peut être dégradée par une configuration trop simple de MPLS et que l'impact des modifications des métriques IP sur le placement des LSP est le plus souvent positif.

- Résoudre le problème d'affectation des capacités aux liens dans les réseaux permet de faire face aux augmentations de trafic. Ce problème a longtemps été traité sans tenir compte des équipements autres que les liens. Nous proposons, dans le chapitre 5, une nouvelle approche globale du dimensionnement de topologie tenant compte des contraintes matérielles et utilisant des coûts d'équipement réalistes. Une méthode exacte et une heuristique efficace aux temps de calcul réduits sont proposées. Comparées aux solutions affectant aux liens les capacités admissibles

directement supérieures à leurs charges, nos solutions amènent un gain économique significatif.

- Nous détaillerons enfin, dans le chapitre 6, différentes évolutions apportées au logiciel NEST de la start-up QoS Design. Elles portent principalement sur le moteur de routage, l'analyse des pannes, et l'intégration des accès radios 2.5/3G.

CHAPITRE 1

Ingénierie des réseaux IP

Depuis les années 90, Internet a connu un succès fulgurant. D'un côté un nombre d'utilisateurs et des volumes de trafic en perpétuelle augmentation et de l'autre plusieurs évolutions technologiques importantes ont fait d'Internet un outil majeur dans des domaines tels que l'informatique et les télécommunications.

Internet et les réseaux IP en général ont donc subi de nombreuses mutations tant au niveau de leur technologie que de leur ingénierie. Nous proposons dans ce chapitre d'exposer plus précisément les changements intervenus dans le fonctionnement et l'utilisation d'Internet. Nous mettrons en évidence plusieurs problèmes convergeant tous vers la nécessité d'une nouvelle approche pour l'ingénierie des trafics dans les réseaux IP.

1.1 Succès d'Internet et de son service Best Effort

Internet est basé sur le protocole IP (Internet Protocol [16]) permettant la mise en place du routage et la propagation des flots de données. Cette technologie simple associée aux réseaux IP est parfaitement adaptée aux services de type « Best Effort (BE) » initialement fournis par Internet. Ce type de service ne fournit aucune garantie (délai, pertes) aux flots écoulés par le réseau. Si le protocole de transport TCP ([17]) est utilisé, les flots ont tout de même la garantie d'être transmis de manière intègre au bout d'un certain temps.

Le routage consiste en une mise en place des routes permettant la communication entre les différents éléments composant les réseaux IP : les routeurs. Ces derniers sont les points d'aiguillages formant l'architecture du réseau IP. Ils possèdent tous une table de routage résumant l'information de routage sous forme de prochain saut (« next hop »), c'est à dire qu'une entrée de cette table associe une destination au prochain routeur vers lequel le flot doit se diriger. De manière plus précise, il existe sur chaque routeur un ensemble d'interfaces permettant la communication avec les routeurs connectés à ce routeur. C'est

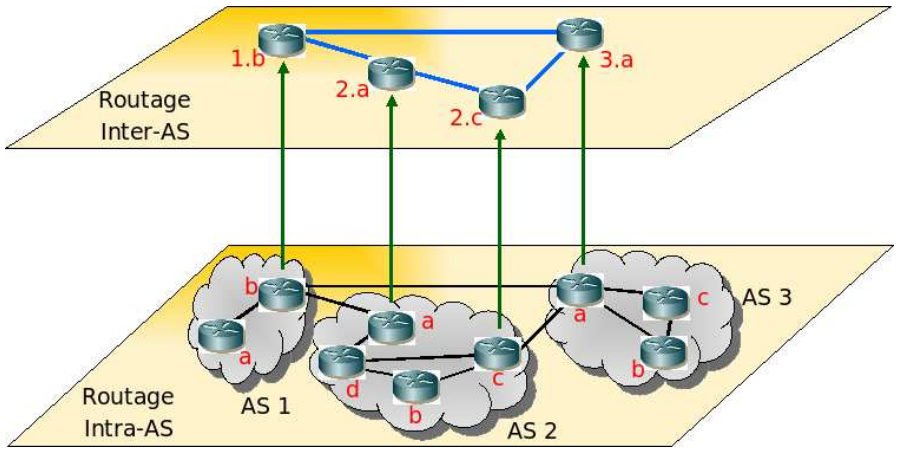


FIG. 1.1 – Routage IP hiérarchique

en fait l'interface à emprunter qui est renseignée dans les tables de routage.

Les réseaux IP sont hiérarchiques car composés de systèmes autonomes (AS) indépendamment gérés les uns des autres comme illustré sur la figure 1.1. Chaque système autonome implémente un protocole de routage intra domaine (« Interior Gateway Protocol (IGP) ») tel que OSPF ([45]), ISIS ([23]), RIP ([21]), ou EIGRP ([20]). Ce dernier met en place les routes entre les routeurs d'un même AS à l'aide d'un algorithme calculant les plus courts chemins (Dijkstra [3], Bellman Ford [2]) au sens des métriques associées à chaque interface. Les pseudo codes de ces deux algorithmes sont donnés dans les tableaux respectifs 1.1(a) et 1.1(b). Ils permettent pour une source S et pour toute destination v de connaître la distance $D(v)$, le nœud précédent sur le(s) plus court(s) chemin(s) $pred(v)$ à partir des métriques $c(i,j)$ sur les interfaces allant de i vers j.

(a) **Algorithme de Dijsktra**

$N = \{S\}$ où S est le nœud source
Pour chaque nœud v
| Si v est adjacent à S
| | Alors $D(v) = c(S,v)$
| | Sinon $D(v) = \infty$
Tant que tous les nœuds ne sont pas dans N
| Trouver $w \notin N$ tel que $D(w)$ est minimum
| Ajouter w à N
| **Pour** tous les nœuds $v \notin N$ adjacents à w
| | $D(v) = min(D(v), D(w) + c(w,v))$
| | $pred(v) = argmin(D(v), D(w) + c(w,v))$

(b) **Algorithme de Bellman-Ford**

Pour chaque nœud v
| Si $v = S$
| | Alors $D(v) = 0$
| | Sinon $D(v) = \infty$
Pour $i = 1$ jusqu'à (Nombre de sommets - 1)
| **Pour** chaque arc (u,v) du graphe
| | si $D(u) + c(u,v) < D(v)$ alors
| | | $D(v) = D(u) + c(u,v)$
| | | $pred(v) = u$

TAB. 1.1 – Algorithmes des plus courts chemins utilisés par les protocoles de routage IP

Les routes choisies sont donc complètement définies par les métriques choisies par l'opérateur. Deux heuristiques classiques existent pour le choix des métriques :

- toutes les métriques égales à 1 pour minimiser le nombre de sauts (nombre de routeurs traversés) associé à chaque route, ou
- la métrique inversement proportionnelle à la capacité de l'interface pour favoriser les chemins ayant plus de bande passante.

Le routage entre systèmes autonomes (ou routage inter domaine) est assuré par un « Exterior Gateway Protocol (EGP) » dont le plus communément utilisé est BGP ([26] Border Gateway Protocol). Il est également programmé pour calculer les plus courts chemins à partir de métriques. Cependant, ce routage est totalement lié aux différentes politiques entre opérateurs.

La mise en place du routage est donc totalement distribuée et automatisée. Ceci simplifie d'une part la gestion du réseau pour les opérateurs et permet d'autre part son adaptation dynamique aux pannes. Lors de la transmission d'un flux de données (voir figure 1.2), celui-ci est tout d'abord découpé en plusieurs paquets puis chaque paquet est transmis sur le réseau après que son information de routage ai été ajoutée sous la forme d'une entête IP. Chaque routeur recevant un paquet le place dans une file d'attente, puis le route (décide sur quelle interface il doit l'envoyer) en le plaçant dans une file d'attente de l'interface de sortie choisie.

La capacité des files d'attente (le nombre maximal de paquets en attente qu'elle supporte) est un critère de performance important. En effet, lorsqu'un paquet est placé dans une file d'attente, la charge (en nombre de paquets) de la file d'attente peut être

- très faible comparée à la capacité, dès lors le temps de traitement sera peu élevé,
- proche de la capacité, dans ce cas le délai de traitement devient plus long, ou
- égale à la capacité, ce qui entraîne la perte du paquet.

Ainsi, les délais de transmission des paquets dépendent des délais de propagation des liens ainsi que du nombre et de la charge des interfaces réseaux traversées.

Le service ainsi proposé par Internet est du type « Best Effort » ne garantissant au mieux que le paquet finira par arriver un jour (pour TCP). Ce service a été pendant longtemps adapté aux classiques transmissions de données effectuées sur Internet comme l'échange de fichiers ou de mails ce qui a permis la croissance rapide d'Internet. Cependant, comme nous allons le montrer juste après, cela n'est plus suffisant.

1.2 Evolutions majeures d'Internet depuis ses débuts

Les évolutions autour d'Internet peuvent être vues sous deux angles : celui de son utilisation et celui de son évolution technologique. Le premier porte principalement sur l'évolution du nombre d'utilisateurs ainsi que sur l'apparition régulière de nouveaux services. Le second, plus technique, porte sur

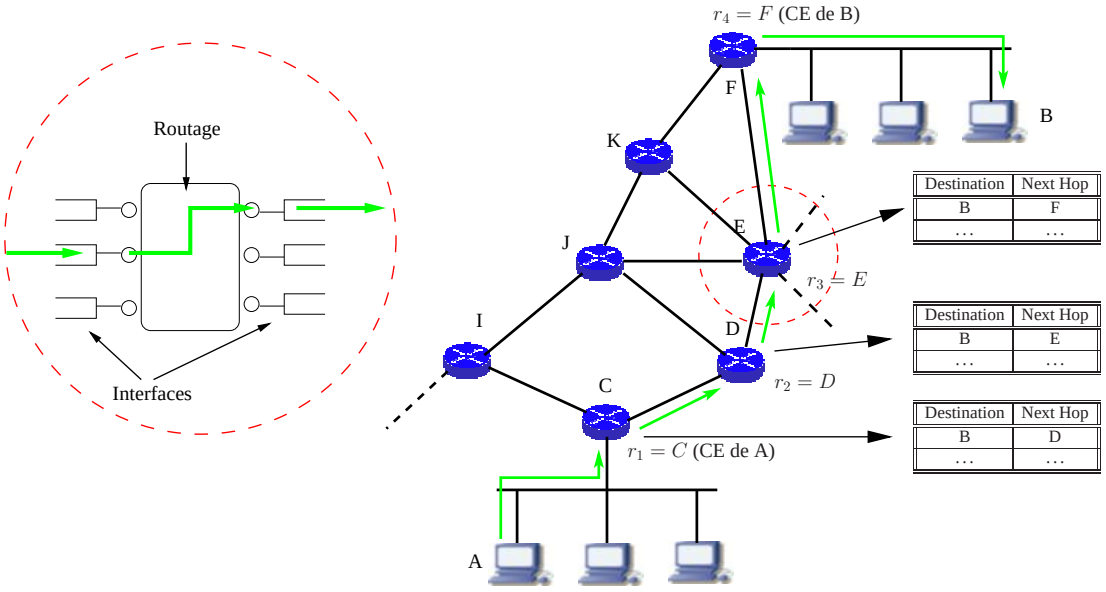


FIG. 1.2 – Propagation dans réseau IP

l'intégration d'un ensemble de nouvelles technologies qui est venue enrichir Internet et qui a conduit à une réelle convergence des services proposés.

1.2.1 Evolution vers une architecture de communication globale

Depuis ses premiers pas où de simples fichiers étaient transmis d'un ordinateur vers un autre, l'utilisation d'Internet a fait un long chemin. Internet est devenu une architecture de communication globale supportant maintenant des applications temps réel tels que la voix sur IP ou la vidéo à la demande.

Ces nouvelles applications n'ont pas les mêmes besoins en termes de Qualité de Service (QoS) que les premiers services proposés par Internet. La qualité de service (QoS) représente des limitations fortes sur

- la latence ou le délai de bout en bout : le temps entre le début de l'émission d'un flot de communication et la fin de sa réception,
- les probabilités de perte des paquets, et
- la gigue : moyenne des écarts des latences des différents paquets appartenant à un même flot de communication.

La voie sur IP par exemple exige des latences et gigues faibles pour pouvoir fonctionner, ainsi le « Best Effort » proposé au début ne suffit plus. Les nouveaux réseaux IP intègrent donc de nouvelles technologies permettant d'offrir des qualités de service différenciées aux flux en fonction de leur classe de service. Parmi celles-ci, on peut citer les deux principales approches utilisées.

L'approche IntServ basée sur les flots [31]

L'idée principale de cette approche est de contrôler l'admission des flots en s'assurant qu'un trafic aura la QoS qu'il requiert avant de l'envoyer. Dans le cas contraire, le trafic n'est pas émis.

Pour cela, l'ensemble des routeurs doit permettre une évaluation de la QoS de bout en bout et cela impose la connaissance de la totalité du chemin. Ainsi, un mécanisme de négociation (géré par RSVP [39]) permet d'autoriser ou non la communication en négociant les ressources nécessaires le long de la route. Cependant, les routeurs doivent garder en mémoire toutes les connexions autorisées pour pouvoir répondre correctement aux requêtes de RSVP.

Cette forte consommation de ressources sur les routeurs est problèmatique pour des réseaux de grandes tailles. On parle alors de problème de passage à l'échelle. L'approche IntServ n'a donc pas connu de grand succès pour les cœurs de réseau de taille conséquente.

L'approche DiffServ basée sur une différenciation par classes de service [44]

Cette approche permet de distinguer chaque paquet par son type de service (ToS) ou sa classe de service (CoS) dépendant du service (transfert de fichiers, mail, VoIP, ToIP, etc…). Dans le domaine DiffServ défini par un ensemble de nœuds appliquant la même politique de QoS, les champs ToS des paquets IP sont remplacés par les champs DS qui contiennent la classe DiffServ du paquet et le statut du paquet en terme de profilage (« in/out profile »). Cette transformation de l'entête IP est effectuée par les routeurs à la frontière du domaine DiffServ, les routeurs à l'intérieur du même domaine se servent de ces nouvelles informations pour le routage.

Les routeurs frontières effectuent un ensemble de traitements illustrés par la figure 1.3 lorsqu'un paquet se présente à l'entrée du domaine DiffServ :

- Le « Meter » est un organe qui permet une mesure en temps réel des trafics. Si ces mesures dépassent les valeurs fixées avec l'opérateur, il en informe alors certains autres organes qui se chargeront soit de **jeter** (supprimer) le paquet, soit de le **déclasser** c'est à dire que sa priorité est diminuée (on parle alors de paquet « out profile »).
- Le « Classifier » permet une première classification du paquet entrant en fonction des « Service Level Agreement » (SLA) représentant le contrat passé avec l'opérateur. Chaque classe de paquet possède des garanties propres données par l'opérateur.
- Le « Marker » détermine la classification du paquet ou « Per Hop Behaviour » suivant les informations transmises par les organes antérieurs. Il marque ensuite le paquet en modifiant son entête.
- Le « Shaper » régule ensuite l'émission des paquets et
- le « Dropper »permet d'effectuer cette régulation en supprimant ou en déclassant certains paquets marqués par les organes précédents.

Lors du transit du paquet dans le domaine DiffServ, une politique d'ordonnancement et de traitement dépendant du marquage des paquets est mise en place. Ainsi, l'ensemble des priorités est défini à

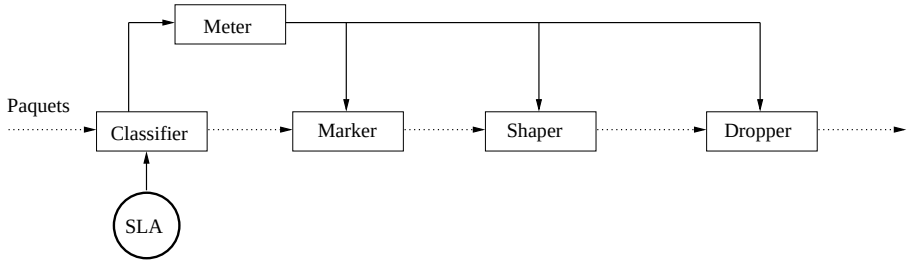


FIG. 1.3 – Traitement d'un paquet à l'entrée du domaine DiffServ

l'entrée du domaine et appliqué lors de la traversée de ce domaine.

Pour cela, chaque interface du cœur de réseau DiffServ possède plusieurs files d'attente, chacune étant associée à un ensemble de classes. L'ensemble des files d'attente d'une même interface est géré par un système de priorité définissant un ordre de traitement : certaines files peuvent être vidées en premier avant de traiter les paquets se trouvant dans les autres files par exemple.

Le mécanisme DiffServ permet donc d'assurer une meilleure QoS pour certaines classes tout en négligeant celle des autres. De plus, le routage par ToS (proposé par OSPF par exemple) permet des routages différents selon de type de service du paquet favorisant des routes plus courtes pour les paquets plus prioritaires. L'absence de contrôle d'admission, basé sur une négociation de QoS de bout en bout, ne permet cependant pas de garantie de QoS de bout en bout.

MPLS

Une autre technologie importante associée aux réseaux IP est MPLS (Multi Protocol Label Switching [68]) qui permet un routage par commutation de labels à opposer au routage par prochain saut effectué par IP.

La commutation de label étant effectuée au niveau 2 de la couche protocolaire, elle est plus rapide que le routage IP (niveau 3). Le succès de MPLS est donc majoritairement dû à la diminution des temps de routage, cependant, cette différence importante il y a quelques années n'est plus d'actualité depuis l'apparition de nouveaux routeurs aux capacités de traitement améliorées. Un autre aspect de MPLS, le « Traffic Engineering (TE) » , permet une gestion fine de la QoS distribuée aux différents flots transitant dans le cœur de réseau, nous reviendrons plus longuement sur ce point dans le chapitre 4.

Les réseaux actuels pourraient bien être bouleversés par le développement attendu de ces « nouveaux » services tant au niveau du nombre d'utilisateurs que des nouveaux usages qui pourraient leur être associés.

1.2.2 Convergence technologique

On assiste depuis une dizaine d'années à la convergence des services proposés par Internet. Etant défini comme une interconnexion de réseaux hétérogènes, il n'est pas surprenant de voir de plus en plus de réseaux d'accès de type différents permettre aux utilisateurs un accès à un nombre toujours croissant de services sur Internet. Les nouveaux accès filaires mais surtout radio comme le GRPS, l'UMTS, le WiFi, ou le WiMAX fournissent maintenant à l'ensemble des utilisateurs la possibilité d'accéder en permanence à un ensemble de services au travers de cœurs de réseaux IP/MPLS. La figure 1.4 illustre bien la convergence actuelle et future.

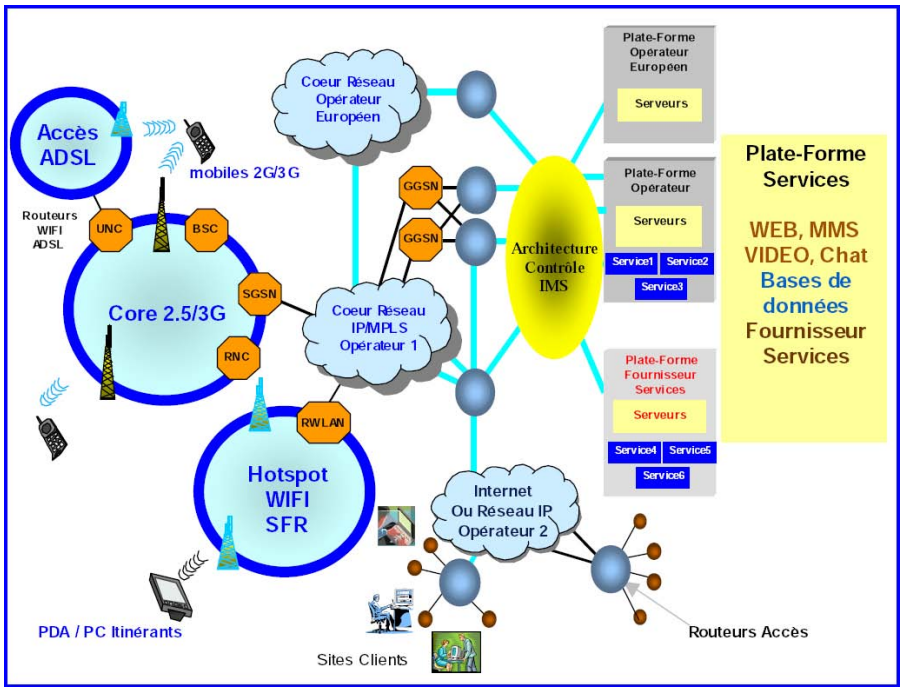


FIG. 1.4 – Convergence technologique au travers des réseaux IP/MPLS

Quel que soit le mode d'accès utilisé, un utilisateur peut donc accéder à un ensemble de services « universels » . Cette connexion quasi permanente à ces services pousse les utilisateurs de plus en plus nombreux à une utilisation de plus en plus fréquente des réseaux IP.

Ces deux évolutions majeures conduisent donc à une augmentation sans précédent du nombre d'utilisateurs et des volumes de trafic qu'ils génèrent. Internet n'est donc plus utilisé uniquement pour les échanges de fichiers ou pour le mail. On se sert maintenant d'Internet pour connaître la météo, ou pour voir la dernière vidéo du moment par exemple et ce depuis son téléphone portable si on le souhaite.

1.3 Rôle crucial d'Internet dans de nombreux domaines

L'engouement pour les réseaux IP a été tel qu'ils ont maintenant un rôle central dans les activités quotidiennes d'un grand nombre de gens. Ce rôle crucial ne peut que croître étant donné les tendances que nous venons de décrire. Cependant, la part croissante de l'Internet dans nos activités quotidiennes soulève de nouveaux problèmes.

Ainsi, il existe des réseaux IP dédiés à la défense nationale ou à la navigation aérienne par exemple dans lesquels aucune interruption de service ne peut être tolérée. Dans de tels réseaux, les services vitaux doivent être maintenus à tout prix aux dépens des autres services mineurs. La notion de résilience, ou la capacité à réagir aux pannes, est donc un enjeu majeur que l'on associe maintenant aux réseaux IP, cette tendance se généralisant de plus en plus même pour des réseaux moins sensibles.

De même, de nombreuses entreprises utilisent les réseaux IP pour interconnecter leur sites distants et ce de manière sécurisée grâce aux VPN (« Virtual Private Network »). Elles aussi utilisent de plus en plus les nouveaux services tels que la téléphonie sur IP ou la vidéo conférence. Payant un prix souvent plus élevé que les utilisateurs classiques, elles attendent des opérateurs des garanties de QoS d'un niveau élevé.

Chacun des plus de 800 millions d'utilisateurs de réseaux IP diffère des autres par la QoS recherchée ainsi que par les volumes de trafics générés. Cependant, la tendance majeure est une croissance exponentielle des deux et les Fournisseurs d'Accès à Internet (FAI) ou les opérateurs se doivent donc de réagir pour réussir à garantir une qualité de service de plus en plus exigeante tout en écoulant des volumes de trafics en perpétuelle augmentation.

1.4 Surdimensionnement des équipements

Initialement, pour faire face à ces nouvelles exigences de QoS et à l'augmentation soutenue du trafic, les opérateurs ou FAI se sont tournés vers un surdimensionnement des équipements. Les principales sources de dégradation de la QoS étant les zones de congestion du réseau, limiter les risques de congestion par une augmentation conséquente des ressources permettait d'écouler les volumes de trafics tout en garantissant la QoS requise.

Cependant cette démarche n'est plus viable économiquement. Le contexte concurrentiel qui induit des marges bénéficiaires réduites ne permet plus d'améliorer les performances d'un réseau IP par un surdimensionnement excessif des équipements.

De plus, d'un point de vue technique, cette démarche doit être modifiée si l'on veut vraiment maîtriser l'évolution des réseaux. La garantie de performances en toutes situations, y compris en cas de panne ou de variations fortes du trafic par exemple, ne peut s'obtenir sans une nouvelle approche de l'ingénierie des réseaux.

1.5 Nécessité d'une nouvelle approche

Une nouvelle approche basée sur un suivi plus régulier et plus fin du réseau pour pouvoir mettre en œuvre des techniques d'ingénierie de trafics doit être étudiée. Le dimensionnement de topologie, l'optimisation du routage, la configuration des mécanismes DiffServ ou encore le placement des LSP dans MPLS sont des exemples de ces techniques qui existent déjà depuis plusieurs années. Cependant, il n'existe pas encore d'approche cohérente autour de ces outils qui se doit de reposer sur trois points essentiels.

- **La Supervision**

  Aucune ingénierie ne peut être efficace sans la connaissance approfondie du réseau et des trafics qu'il achemine. La connaissance du réseau est obtenue à partir d'informations statiques concernant sa topologie telles que la capacité des différents équipements, la configuration du routage, celle des VPN etc.... La connaissance des trafics est, elle, dynamique. Cependant, même si l'information instantanée ou passée des trafics est pertinente, elle n'est pas suffisante. Il faut être capable de prédire les évolutions des volumes des trafics par une analyse de leurs tendances (« traffic trends »).
  Estimer les matrices de trafics du réseau en évitant de s'appuyer sur une métrologie trop coûteuse est un exemple de problème associé à la supervision qu'il est difficile mais important de résoudre.

- **La Planification**

  La planification des réseaux porte autant sur le dimensionnement des équipements que sur l'optimisation du routage. Elle consiste donc à adapter le réseau existant aux volumes de trafics et exigences de QoS qu'il doit supporter. Elle doit bien sûr intégrer les notions de résilience pour garantir non seulement une utilisation adéquate des ressources dans le réseau nominal mais aussi des performances « acceptables » si le réseau est dans un état de panne. Planifier son réseau revient donc également à anticiper par la mise en place de scénarios de secours pour chaque panne possible du réseau.

- **Le Contrôle dynamique**

  Enfin, le dernier point essentiel de l'approche globale d'ingénierie des trafics porte sur la réactivité du réseau. En effet, en raison de l'ensemble des évolutions majeures associées aux réseaux IP que nous avons développées plus tôt, aucun réseau ne peut éviter des dégradations ponctuelles de la QoS du fait de la forte variabilité des trafics.

  Seules des techniques de reconfiguration dynamique des ressources permettent au réseau une adaptabilité permanente aux phénomènes variables qu'il doit supporter. Parmi les problématiques classiques associées au contrôle dynamique, on peut trouver le routage dynamique et la reconfiguration dynamique des LSP.

Avec une telle approche, les opérateurs et les FAI peuvent faire face aux nouvelles exigences de leurs clients dont le nombre et l'activité ne cesse d'augmenter.

1.6 Contributions

C'est dans cette optique que l'ensemble de nos travaux ont été menés. Au sein de la start-up QoS Design qui développe une suite logicielle d'outils pour la supervision, la planification, et la simulation des réseaux IP/MPLS, nous avons contribué à l'amélioration des deux premiers points évoqués juste avant

- la supervision : par l'amélioration des techniques d'estimation des matrices de trafics (chapitre 2),
- et la planification des réseaux : par une contribution principale sur l'optimisation du routage IP (chapitre 3 et 4) et une nouvelle approche pour le dimensionnement des topologies (chapitre 5).

L'ensemble de nos travaux ont un ensemble de caractéristiques communes portant sur des aspects technologiques, opérationnels, et financiers. Ils sont tous intégrés dans le logiciel NEST.

CHAPITRE 2

Estimation des Matrices de Trafic

Le chapitre précédent a permis de mettre en avant la nécessité de techniques d'ingénierie de trafic de plus en plus efficaces. L'ensemble de ces techniques supposent la connaissance a priori de la demande en trafic que le réseau devra supporter pour pouvoir proposer des solutions visant à utiliser au mieux les ressources disponibles.

Connaître l'état de la demande en trafic à un instant donné est donc un problème crucial dont une solution évidente est la **métrologie**. Cela consiste à mesurer les trafics circulant au travers du réseau à l'aide de protocoles logiciels particuliers s'executant sur les routeurs ou encore à l'aide de sondes. La mesure des trafics possède l'avantage d'être précise mais son inconvénient majeur est qu'elle surcharge le réseau. Consommation de CPU sur les routeurs, ralentissement des trafics au travers des sondes et surtout trafics de « reporting » générés sur le réseau sont des exemples de ces surcharges.

S'il n'est pas possible de mesurer de manière exacte les trafics en temps réel, il devient alors inévitable de les estimer. **L'estimation** consiste à utiliser des informations indirectes pour évaluer la demande en trafic que le réseau supporte à un instant donné.

Au delà de la connaissance instantanée des trafics, les techniques d'ingénierie visant à la planification des réseaux ont également besoin de prédictions des trafics. Basées sur des analyses de tendances (« traffic trends ») faites à partir des trafics connus, ces prédictions permettent d'agir en avance sur le réseau. Les travaux effectués portent peu sur ce dernier point qui sera discuté plus loin dans ce chapitre.

Depuis maintenant plus d'une dizaine d'années, les techniques d'estimation ont été prises en considération par de nombreux groupes de recherche. En effet, la mesure était alors beaucoup trop lourde à supporter pour les réseaux de l'époque. Beaucoup de pistes ont été étudiées mais il est vrai qu'aucune n'a réellement débouché sur des résultats indiscutables. En parallèle de ces recherches, les techniques de mesures et de collectes d'informations sur les réseaux se sont améliorées. Cependant, même si la mesure s'est grandement allégée en termes de surcoût sur les réseaux, la compétition avec les techniques d'estimation est encore d'actualité. Les dernières techniques proposées sur ce sujet sont d'ailleurs com-

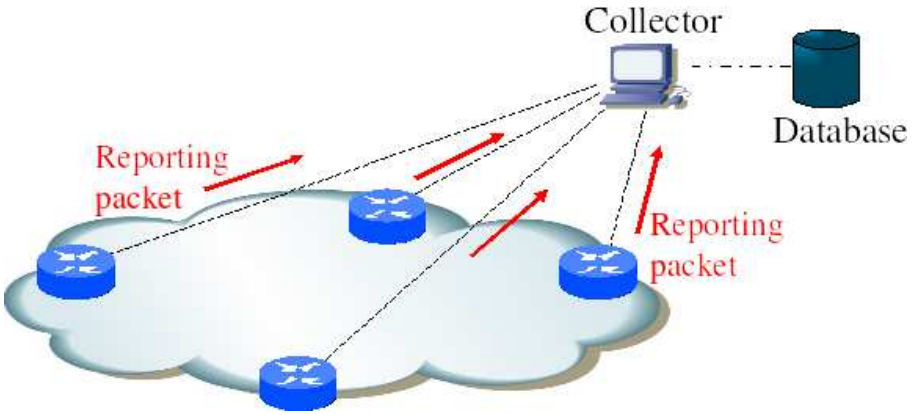


FIG. 2.1 – Un outil de métrologie : NetFlow (Cisco/IPFix)

posées des deux aspects.

L'outil de référence en termes de métrologie est NetFlow (Cisco/IPFix [93]). Comme illustré sur la figure 2.1, chaque routeur sur lequel NetFlow est executé fournit de manière régulière des informations sur les flots qu'il a traités. Parmi celles-ci, on trouve par exemple la source, la destination, l'application (le port), ou la charge du flot.

Le trafic généré vers le collecteur d'information est donc conséquent et entraîne des surcharges sur le réseau. La consommation CPU sur le routeur ralentit considérablement leur fonction principale de routage des paquets. Cette solution n'est donc pas acceptable pour un réseau opérationnel. L'amélioration des techniques de métrologie nous semble cependant être un thème de recherche important possédant des applications conséquentes dans le monde de l'ingénierie des réseaux. Néanmoins, nos travaux ne portant pas sur ce point, nous ne reviendrons pas sur ce sujet.

L'estimation des matrices de trafic consiste donc à trouver les demandes des trafics supportés par le réseau à partir d'autre données mesurées sur le réseau. Le protocole SNMP ([37]), executé sur la majorité des routeurs utilisés, permet justement des mesures plus légères. Il fournit une valeur moyenne des charges des interfaces toutes les 5 minutes. La quantité d'information remontée vers le collecteur est donc très faible comparée à NetFlow ce qui permet son utilisation de manière continue. Néanmoins, les mesures fournies par SNMP ne permettent pas de distinguer les différents flots (source, destination, application), les trafics doivent donc être estimés.

La figure 2.2 expose la configuration de base d'un cœur de réseau sur lequel nous cherchons à évaluer les trafics. Nous considérerons donc un cœur de réseau (intérieur du cercle pointillé sur la figure) connu dans lequel le routage est donné et la plupart du temps considéré comme fixe au cours du temps. Dans ce cœur de réseau, nous pourrons obtenir des mesures sur les charges des liens (notées Y). Les points d'accès à ce cœur de réseau sont les routeurs frontières (routeurs rouges sur la figure) et les trafics cherchés sont ceux circulant d'un routeur frontière vers un autre. On supposera également que les routeurs du cœur de réseau ne sont ni source, ni puit de trafic. La totalité du trafic « entrant »dans le

réseau par les routeurs edges (ou frontières) devra donc « sortir » par ces mêmes routeurs frontières.

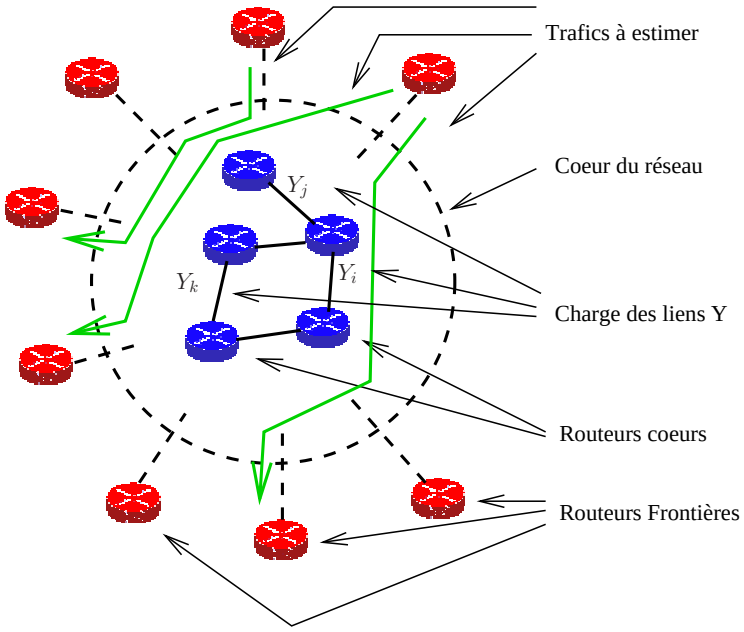


FIG. 2.2 – Modèle de réseau pour l'estimation des matrices de trafic

Etant donné l'aspect extrêmement attractif de ce sujet dans le monde scientifique depuis les premiers travaux de Vardi en 1996 ([38]), nous consacrerons la prochaine partie de ce chapitre à un état de l'art sur le sujet.

2.1 Etat de l'art

Les charges sur les liens Y peuvent être facilement connues à l'aide du protocole SNMP. Il est actuellement présent sur la quasi totalité des routeurs et permet de mesurer des moyennes de charge sur 5 minutes sur l'ensemble des liens du réseau.

Trouver la matrice de trafic X ayant généré les charges Y au travers d'un réseau dont le routage est connu (matrice de routage A) revient à inverser l'équation de propagation :

$$\mathbf{Y} = A\mathbf{X} \tag{2.1}$$

Cependant, le nombre de liens présents dans le réseau (noté L) est la plupart du temps bien inférieur au nombre de couples Origine/Destination (OD) (noté c). Nous somme en présence d'un système sous-contraint : il y a plus d'inconnues que d'équations. Il n'y a donc pas une seule solution à ce système mais une infinité de solutions et la décomposition en valeurs singulières de la matrice A permet de définir cet espace de solutions.

Si nous notons r le rang de la matrice A, il est possible de trouver une base orthonormée $(\mathbf{u_i})_{1\leq i\leq L}$ de R^L et une base orthonormée $(\mathbf{v_i})_{1\leq i\leq c}$ de R^c tel que :

$$\begin{aligned} A\mathbf{X} &= \sum_{k=1}^{r} \mathbf{u_k}\sigma_k(\mathbf{v_k}^T\mathbf{X}) && (2.2)\\ \sigma_k &= \sqrt{\lambda_k} = \sqrt{\nu_k} \ \forall k = 1..r \\ image(A) &= vect(\mathbf{u_k},\ k = 1..r) && (2.3)\\ null(A) &= vect(\mathbf{v_k},\ k = r+1..c) && (2.4) \end{aligned}$$

Où le vecteur λ de taille c (respectivement ν de taille L) contient les valeurs propres de la matrice A^TA (respectivement AA^T) ordonnées de manière décroissante.

Par conséquent , on peut dire que
- l'image de l'application A est l'espace vectoriel généré par les vecteurs $(\mathbf{u_k},\ k = 1..r)$ (équation 2.3)
- et que le noyau de l'application A est l'espace généré par les vecteurs $(\mathbf{v_i})_{1\leq i\leq c}$ (équation 2.4). De manière plus précise, pour tout $\mathbf{Z}$ s'écrivant sous la forme $\mathbf{Z} = \sum_{k=r+1}^{c} z_k\mathbf{v_k}$ alors $A\mathbf{Z} = 0$.

Notons S l'espace des solutions du système (2.1), et $\mathbf{X_1} = \sum_{k=1}^{c} x_k^1\mathbf{v_k}$, et $\mathbf{X_2} = \sum_{k=1}^{c} x_k^2\mathbf{v_k}$ deux vecteurs de S tel que $\mathbf{X_1} \neq \mathbf{X_2}$. Dès lors :

$$x_k^1 = x_k^2,\ \forall k = 1..r$$

On peut donc trouver un ensemble de valeurs (z_k) pour $k = r+1..c$ tel que $\mathbf{X}_1 = \mathbf{X}_2 + \sum_{k=r+1}^{c} z_k$. L'ensemble des solutions S peut alors être écrit sous une autre forme :

$$S = \{\mathbf{X}^*\} \bigoplus vect(\mathbf{v_k}, k = r+1..c)$$

Où l'opérateur $\bigoplus$ effectue la somme de deux espaces et $\mathbf{X}^*$ est une solution du système (2.1).

Soit une matrice orthogonale U de taille $L\times L$ construite à partir des vecteurs colonnes $\mathbf{u_k}$ placés les uns à coté des autres, ainsi qu'une matrice orthogonale V^T de taille $c\times c$ construite à partir des vecteurs lignes $\mathbf{u_k}^T$ placés les uns au dessus des autres. Soit Σ la matrice de taille $L \times c$ dont les seuls éléments non nuls sont les r premières valeurs sur la diagonale : $\Sigma_{k,k} = \sigma_k$. Alors on retrouve la formulation classique de la décomposition en valeurs singulières :

$$A = U\Sigma V^T \quad (2.5)$$

Nous pouvons donc trouver un ensemble de solutions exactes au système (2.1) mais nous n'avons aucun moyen de savoir quelle solution est la matrice cherchée. La figure 2.3 illustre ce problème avec un réseau de petite taille (seulement 3 nœuds). Nous ne donnons pas la matrice de routage dans cet exemple car elle est évidente. Les deux vecteurs proposés $\mathbf{X}^1$ et $\mathbf{X}^2$ génèrent les mêmes charges sur ce réseau. Cependant, nous n'avons aucun moyen de savoir lequel des deux est la solution.

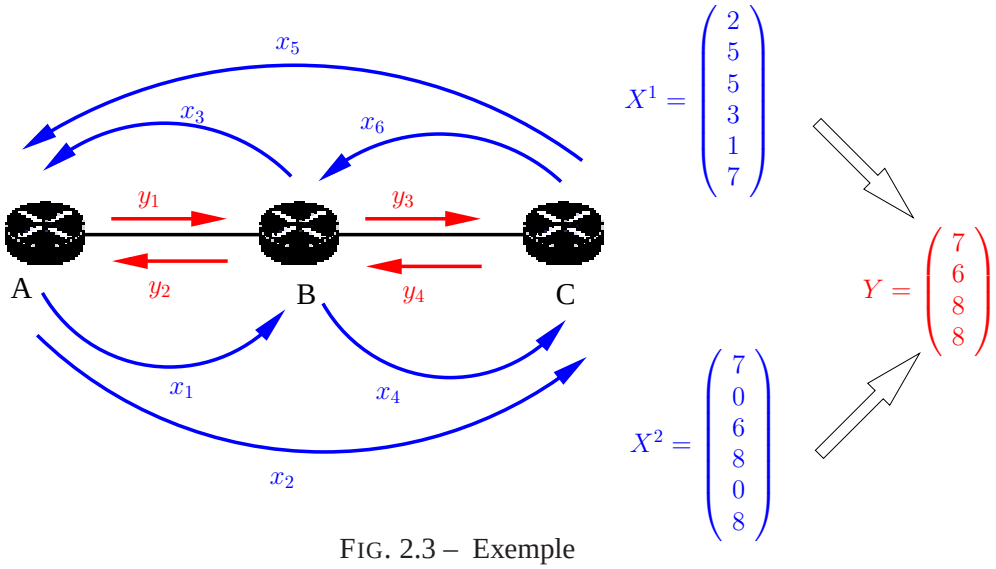


FIG. 2.3 – Exemple

Etant donné que les charges sur les liens ne sont pas suffisantes pour trouver de manière exacte la matrice de trafic recherchée, de l'information supplémentaire doit être ajoutée. Les travaux déjà effectués autour de l'estimation de trafic ont tous cherché à ajouter de l'information. Les techniques proposées diffèrent donc principalement sur la manière dont cette information est ajoutée.

Nous ne donnerons pas trop de détail sur chacune des méthodes présentées ensuite, mais nous conseillons la lecture de [65] où un état de l'art complet des plus anciennes méthodes est proposé ainsi que celle de [84] pour des précisions sur les méthodes les plus récentes.

2.1.1 Utilisation de la covariance de la charge des liens et d'information temporelle a priori

L'information ajoutée dans une première génération de méthodes fut la covariance de la charge des liens. Une série de mesures SNMP au cours du temps est utilisée. C'est à dire que nous cherchons à identifier les trafics au cours du temps (principalement les moyennes) à partir de plusieurs mesures faites dans le temps. Des relations entre la covariance des charges des liens et l'évolution des trafics ont permis de combler ce manque d'information dont nous parlions plus haut. Le premier auteur à publier une méthode de ce type fut Vardi en 1996. De nombreux travaux ont ensuite été entrepris dans cette direction dans les années qui suivirent. Récemment, de nouveaux travaux utilisant cette approche, mais avec de nouvelles hypothèses, ont été entrepris et ont donné de bons résultats.

L'idée principale est d'estimer le second moment de la distribution de charge sur les liens au cours du temps et de le relier aux moyennes des trafics de manière à augmenter le rang du système que l'on cherche à résoudre.

Tout d'abord, en prenant la moyenne (premier moment) de $\mathbf{Y}$ (notée $\overline{\mathbf{Y}}$) et de $\mathbf{X}$ (notée λ) à partir de l'équation (2.1), on peut écrire l'équation du premier moment (2.6). Ensuite, en notant $\Sigma^{(y)}$ la ma-

trice de covariance des charges des liens et $S^{(y)}$ le vecteur de dimension $\frac{L(L-1)}{2}$ contenant les éléments diagonaux et du triangle supérieur de $\Sigma^{(y)}$, nous pouvons écrire l'équation du second moment (2.7). Où $B_{ij,k} = A_{ik}A_{kj}$, et $S^{(x)}$ représente le vecteur de dimension c contenant les éléments diagonaux de la matrice de covariance des trafics $\Sigma^{(x)}$ (supposée être diagonale vu que les trafics sont supposés être indépendants les uns des autres).

$$\overline{\mathbf{Y}} = A\lambda \tag{2.6}$$
$$S^{(y)} = BS^{(x)} \tag{2.7}$$

Des hypothèses supplémentaires doivent être ajoutées à cette nouvelle formulation pour qu'elle puisse être résolue. Suivant les hypothèses choisies, nous pouvons distinguer plusieurs approches.

Modèle Poissonnien

Vardi, dans [38], suppose que les trafics suivent des distributions Poissonniennes indépendantes $\mathbf{X} \sim \text{Poisson}(\lambda)$. Son objectif premier est de prouver l'identifiabilité du système :

$$\forall y \quad p(y,\lambda) = p(y,\lambda^{'}) \Rightarrow \lambda = \lambda^{'} \tag{2.8}$$

Il prouve alors dans cet article que les valeurs moyennes des flots sont identifiables sous cette hypothèse. Cela provient en fait de l'égalité entre la moyenne et la variance. Son objectif était donc d'estimer le vecteur des trafics moyens λ à l'aide de méthodes statistiques. Pour cela, Vardi propose d'utiliser une méthode de maximisation de vraisemblance avec un algorithme EM.

Plus tard, en 1998, Tebaldi et West ([46]) supposent le même modèle de distribution a priori pour les trafics mais ils proposent une approche de type Bayesienne pour l'analyse et l'estimation.

Cependant, comme l'hypothèse de trafic Poissonnien n'est pas vérifiée dans les réseaux IP (voir [74], [84] et [65] pour plus de détails), toutes ces méthodes ne donnent pas de résultats suffisamment précis pour être utilisées par les fournisseurs d'accès. L'idée de tenir compte des covariances des charges des liens a été utilisée avec d'autres hypothèses sur la distribution des trafics.

Modèle Gaussien et loi de puissance

Cao et Al. considèrent un modèle Gaussien pour les trafics dans [52] (2000) : la distribution de probabilité des trafics est supposée être gaussienne et les trafics sont supposés être indépendants les uns des autres : $\mathbf{X}_t \sim \text{normal}(\lambda, \Sigma)$.

Cependant, comme il n'existe plus d'égalité entre la moyenne et la variance des trafics, une autre hypothèse doit être faite de manière à rendre le système observable : Cao et Al. supposent que la loi de puissance s'applique sur les trafics étudiés. Les auteurs peuvent alors écrire la matrice de covariance en fonction des moyennes :

$$\Sigma = \Phi \ \text{diag}(\sigma^2(\lambda_{\mathbf{1}}), \ldots, \sigma^2(\lambda_{\mathbf{c}})) \tag{2.9}$$

avec Φ qui est un facteur d'échelle et σ^2 qui représente la loi de puissance entre moyenne et variance dépendant d'une constante K_p à définir :

$$\sigma^2(\lambda) = \lambda^{K_p} \tag{2.10}$$

On remarque alors qu'en choisissant $\Phi = 1$ et $K_p = 1$, on retombe sur l'approximation proposée par Vardi dans [38].

Les auteurs présentent un certain nombre de résultats expérimentaux validant leurs hypothèses et affirment alors que le modèle associé à un facteur de puissance K_p fixé est identifiable. L'algorithme développé par Cao et Al est basé sur une estimation du maximum de vraisemblance.

Cependant, Medina et Al ont prouvé dans leur article [65] que l'hypothèse Gaussienne n'est pas tout le temps vérifiée dans les réseaux réels. Les valeurs du modèle peuvent changer pour différentes heures de mesures et s'avérer complètement fausses certaines fois. Dans le même article, Medina et Al. affirment par contre que la loi de puissance est vérifiée dans les réseaux réels.

Loi de puissance seule

Nous avons vu jusque là que le système (2.6)-(2.7) était identifiable sous différentes hypothèses :
- distribution Poissonnienne des trafics, ou
- distribution Gaussienne des trafics et loi de puissance : la relation entre la moyenne et la variance conduit à l'identifiabilité du système.

Allant plus loin, Soule et Al. dans [81] ont prouvé que les statistiques du second ordre pour les trafics (leurs covariances) sont identifiables à partir des seules équations du second moment sur les liens si les hypothèses d'indépendance des trafics et de routage stable sont supposées. Dès lors, en supposant une relation entre variance et moyenne comme la loi de puissance, les moyennes des trafics peuvent être trouvées.

Plus récemment en 2006, Vaton et al. ont décidé d'explorer cette approche dans [87]. Ils proposent de résoudre directement les équations du second moment (2.7) en utilisant la méthode du pseudo inverse. Cela donne en fait une estimation au sens des moindres carrés : $S^{(x)} = (BB^T)^{-1}B^T S^{(y)}$. Ensuite, Vaton et Al. n'ont plus qu'à estimer les moyennes (i.e. le vecteur λ de dimension c) et pour se faire, ils proposent deux méthodes : une méthode de projection et une méthode de minimisation sous contrainte. Les résultats de ces algorithmes sont relativement bons comparés à ceux de la technique d'estimation du maximum de vraisemblance (Maximum Likelihood Estimation MLE) mais surtout, le temps de calcul est grandement réduit.

2.1.2 Utilisation d'informations spatiales ou temporelles a priori

Nous avons présenté juste avant une première approche qui consiste à supposer une loi marginale pour les trafics de manière à ajouter de l'information au problème de base (2.1). Cependant vu les résultats limités de ces approches qui sont principalement dus à la difficulté de bien représenter

l'évolution des trafics dans le temps, d'autres auteurs ont cherché à introduire de l'information a priori mais portant sur des relations soit spatiales soit temporelles.

Modèle de gravité

Le modèle de gravité proposé par Zhang et Al. dans [70] suppose que la matrice de trafic suit des règles de gravité c'est à dire que les trafics sont liés spatialement. Ainsi, le trafic entre deux nœuds i et j est proportionnel au trafic entrant dans le réseau par le nœud i et au trafic sortant du réseau par le nœud j :

$$x_{i,j} \propto x_{i,*} \times x_{*,j} \qquad x_{i,*} = \sum_{j \neq i} x_{i,j} \tag{2.11}$$

Changement de routage

Une autre manière pour ajouter de l'information est de faire plusieurs mesures SNMP tout en faisant évoluer le routage au cours de ces mesures. Ces évolutions de routage doivent bien sur être connues. Cela a été proposé dans [82] comme une des techniques possibles ajoutant de l'information temporelle. C'est à dire qu'entre chaque mesure, on fait évoluer la topologie de manière à ajouter de l'information.

Dans [78], les auteurs proposent une heuristique pour calculer le nombre minimum de changements de métrique de manière à obtenir un système de rang plein. Mais même si un nombre minimum de changements peut être trouvé, changer le routage un certain nombre de fois de manière à effectuer des mesures semble inacceptable pour les opérateurs. De plus, les trafics doivent être considérés comme constants entre chaque mesure sinon, ce n'est qu'une multiplication du système de base.

Nous venons de présenter rapidement deux techniques qui essaient d'obtenir des informations supplémentaires à partir d'hypothèses spatiales ou temporelles sur les trafics. Cependant, même si ces techniques peuvent se révéler être efficaces dans certaines conditions d'application, qu'en est-il de leurs résultats si les hypothèses formulées sont fausses ?

2.1.3 Utilisation combinée d'informations spatiales et temporelles

Pour faire face au problème des méthodes précédentes, une troisième génération de techniques tente de représenter le comportement réel des trafics. [82] dresse un état de l'art de ces techniques et elles ont toutes le même principe de fonctionnement : un modèle est construit à partir des mesures Netflow (calibration du modèle) puis ce modèle est utilisé pour prédire les trafics.

Filtre de Kalman

Le problème d'estimation de la matrice de trafic X à partir des mesures Y à l'aide de la relation $Y = AX$ peut se ramener à celui d'un filtre où l'on cherche à estimer les entrées à partir des sorties. Soule et Al. proposent donc dans leur article [83] d'utiliser une telle approche basée sur un filtre de Kalman. L'idée développée est de proposer un modèle de filtre adapté à l'estimation des matrices de trafic, de le calibrer à l'aide de mesures Netflow puis de l'utiliser ensuite pour estimer les futurs trafics.

Les auteurs ne donnent pas d'informations dans cet article ([83]) sur l'observabilité du système. Cependant, si la matrice C représentant la dynamique de $X_{k+1} = CX_k$ est diagonale, le système n'est pas observable. Or, si les trafics sont considérés comme indépendants les uns des autres, cette matrice devrait être diagonale d'où une non-observabilité du système. Il semblerait donc que les mesures Netflow permettent d'identifier quelques corrélations entre les flots ce qui donne au système son observabilité.

Approche des fanouts

Papagiannaki et Al. ont montré dans ([79]) que même si les trafics ne sont pas constants au cours du temps, il existe certaines propriétés temporelles au niveau des proportions émises à partir d'un nœud vers tous les autres. Plus précisément, en notant E_i la totalité des trafics entrant dans le cœur de réseau par le nœud i et $p_{i,j}$ la proportion de trafic circulant entre i et j (c'est à dire que $X_{i,j} = E_i p_{i,j}$), Papagiannaki et Al. démontrent que les coefficients de proportion (les $(p_{i,j})$), qu'ils nomment les fanouts, ont un comportement relativement cyclique. L'idée proposée est alors de mesurer les fanouts durant une journée à l'aide de Netflow puis d'utiliser ces fanouts pour donner une estimation des futurs trafics. Une explication plus approfondie de cette hypothèse est proposée dans la partie suivante.

A notre avis, toutes les techniques de troisième génération permettant une estimation en temps réel des trafics sont à la fois très performantes et inutilisables en pratique. Tout simplement parce que l'utilisation d'un outil de métrologie comme NetFlow de manière régulière ne peut être adaptée aux réseaux opérationnels.

Cependant, ajouter de l'information temporelle et spatiale semble une approche prometteuse permettant de s'abstraire des techniques de métrologie fine telle que NetFlow. Nous proposons en effet une contribution dans ce domaine permettant une estimation efficace des trafics à partir des seules mesures SNMP.

2.2 Approche développée

A notre avis, la notion de Fanouts et les hypothèses de faible variabilité mises en avant par Papagiannaki dans [79] nous semblent être des propriétés importantes et utilisées bien en dessous de leurs capacités. En effet, les auteurs montrent également dans cet article que les fanouts ont une faible variabilité sur une durée d'une heure. Dans la plupart des réseaux d'entreprise ou d'opérateurs, les trafics ont des répartitions peu changeantes. La présence de serveurs, l'utilisation de point d'accès à internet sont

des exemples fréquents qui valident ces hypothèses d'après nous.

Nous proposons donc de partir de l'hypothèse de faibles variations des fanouts (les $p_{i,j}$) sur des périodes courtes de l'ordre d'une heure. L'hypothèse de cyclo-stationnarité [79] sera utilisée de manière différente pour la prédiction des trafics une fois ceux-ci estimés.

Soule et Al. ont proposé dans [82] d'utiliser la propriété de faible variabilité des fanouts pour construire une méthode de troisième génération : mesurer les Fanouts sur un jour à l'aide de Netflow puis les utiliser dans des étapes de prédiction.

Nous pensons que la cyclo-stationnarité des fanouts est plus hypothétique. Il est étonnant de supposer des proportions identiques entre une utilisation du réseau en semaine ou en week-end par exemple. Nous proposons donc d'estimer les fanouts (donc les trafics) pour chaque période de quasi stationnarité puis d'étudier les tendances éventuelles pour permettre une prédiction acceptable des trafics futurs. Cette notion de prédiction sera discutée vers la fin de ce chapitre.

Pour estimer les fanouts (les $p_{i,j}$), nous utiliserons plusieurs mesures SNMP au cours de la période considérée durant laquelle les fanouts sont supposés subir de faibles variations. Ces mesures seront liées les unes aux autres par la propriété des fanouts. Les trafics étant liés par la définition de ces mêmes fanouts. Ainsi, partant du problème d'estimation des matrices de trafic tel qu'il a été présenté depuis le début de ce chapitre, nous proposons d'introduire

- une relation temporelle entre les mesures via la quasi stationnarité des coefficients,
- et une relation spatiale entre les trafics via la notion de Fanout

L'objectif que nous cherchons à atteindre est donc l'estimation des trafics ayant généré la série de mesure SNMP, c'est à dire l'estimation, à chaque instant de mesure, des trafics engendrant ces mesures. Ces trafics seront obtenus à partir d'un fanout unique dont le calcul est expliqué plus loin.

La figure 2.4 illustre la notion de fanout ainsi que quelques notations utilisées : le trafic total entrant dans le cœur de réseau par le nœud i (E_i) peut être récupéré à l'aide de mesures SNMP en sommant les charges des liens partant de i, les fanouts (les variables $p_{i,j}$) doivent être estimés et les trafics $X_{i,j}$ seront directement calculés par la formule $X_{i,j} = p_{i,j}E_i$.

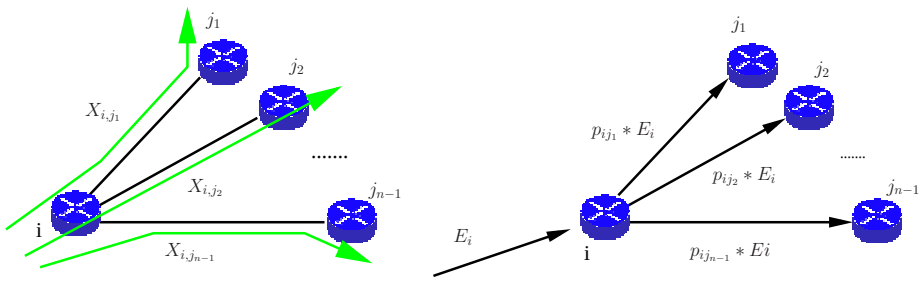


FIG. 2.4 – Description des fanouts

Les inconnues seront donc les coefficients contenus dans le vecteur $\mathbf{P}$ de dimension c. Deux notations pourront être utilisées suivant le cas d'utilisation. En effet, par un jeu d'indice, nous noterons indifféremment ces coefficients p_i où i est un couple Origine-Destination (donc $1 \leq i \leq c$) ou alors $p_{i,j}$ où i sera le nœud source et j le nœud destination (donc $(i,j) \in [|1,N|]^2$ avec $i \neq j$). Si nous résumons ces premières notations, nous pouvons écrire :

$$\mathbf{P} = \begin{pmatrix} p_{1,2} \\ \vdots \\ p_{1,N} \\ p_{2,1} \\ p_{2,3} \\ \vdots \\ p_{N,N-1} \end{pmatrix} = \begin{pmatrix} p_1 \\ \vdots \\ p_{N-1} \\ p_N \\ p_{N+1} \\ \vdots \\ p_c \end{pmatrix}$$

Nous gardons les mêmes notations que précédemment pour la matrice de routage A. De plus, nous allons faire τ mesures SNMP et pour chaque mesure $t, 1 \leq t \leq \tau$, nous aurons le vecteur $\mathbf{Y}^t$ de dimension L qui contiendra les mesures de charge sur les liens et le vecteur $\mathbf{E}^t$ de dimension c qui sera construit de la manière suivante : $N-1$ fois la valeur du trafic entrant au nœud 1 à l'instant t (E_1^t) puis $N-1$ fois la valeur du trafic entrant au nœud 2 à l'instant t (E_2^t) etc ...Nous pouvons donc écrire ceci sous la forme :

$$\widehat{\mathbf{E}}_i^t = \underbrace{(E_i^t, E_i^t, \ldots, E_i^t)}_{N-1} \qquad \mathbf{E^t} = \begin{pmatrix} (\widehat{\mathbf{E}}_1^t)^T \\ (\widehat{\mathbf{E}}_2^t)^T \\ \vdots \\ (\widehat{\mathbf{E}}_N^t)^T \end{pmatrix}$$

Si nous reprenons l'équation de propagation (2.1) et que nous l' écrivons pour l'instant t avec les notations définies ci-dessus, on obtient :

$$\mathbf{Y}^t = A\mathbf{P}\mathbf{E}^t \tag{2.12}$$

Ce système est valable pour chaque instant de mesure t donc le problème entier revient à estimer le vecteur $\mathbf{P}$ solution du système suivant :

$$\begin{pmatrix} \mathbf{Y}^1 \\ \mathbf{Y}^2 \\ \vdots \\ \mathbf{Y}^\tau \end{pmatrix} = \begin{pmatrix} A\mathbf{P}\mathbf{E}^1 \\ A\mathbf{P}\mathbf{E}^2 \\ \vdots \\ A\mathbf{P}\mathbf{E}^\tau \end{pmatrix} \tag{2.13}$$

Si les fanouts sont constants, il existe une solution P unique pour τ suffisamment grand (rang plein). En pratique, on peut supposer que les fanouts varient peu mais qu'ils ne sont pas constants. Nous allons donc chercher la valeur des fanouts permettant de minimiser l'erreur quadratique sur les mesures de charge.

$$\underset{\mathbf{P}}{\text{Minimiser}} \quad \Gamma(P)$$

Avec :

$$\Gamma(P) = \sum_{t=1}^{\tau} ||\mathbf{Y}^t - A\mathbf{P}\mathbf{E}^t||^2 = \sum_{t=1}^{\tau}\sum_{l=1}^{L} \left(\frac{y_l^t - \sum_{od=1}^{c} a_{l,od}\, p_{od}\, E^t_{s(od)}}{y_l^t} \right)^2 \tag{2.14}$$

Où $s(od)$ est la source du couple OD od

Sous les contraintes :

$$\begin{cases} \sum_{j \neq i} p_{i,j} = 1 & \forall i = 1..N \\ 0 \leq p_i & \forall i = 1..c \end{cases} \tag{2.15}$$

L'espace des solutions admissibles du problème est convexe. En effet, supposons que $\mathbf{P}^1$ et $\mathbf{P}^2$ soient deux solutions respectant les contraintes (2.15). Alors, pour tous réel $\lambda \in [0,1]$, la solution $\mathbf{P}(\lambda) = \lambda\mathbf{P}^1 + (1-\lambda)\mathbf{P}^2$ est également admissible au sens des contraintes (2.15).

Le problème mathématique posé est la minimisation d'une fonction quadratique sur un espace convexe de solutions. Nous pouvons donc le résoudre à l'aide de méthodes itératives du type direction de descente admissible. Cependant, ces méthodes sont parfois lentes à converger aussi nous proposerons également une heuristique cherchant un compromis entre la qualité des résultats proposés et les temps de calcul.

2.3 Méthodes de direction de descente admissible

L'ensemble de ces méthodes est adapté à la résolution de problème de minimisation de fonction coût convexe sur des ensembles convexes. Les méthodes de direction de descente admissible permettent de former une suite de solutions admissibles convergeant vers ce minimum quelle que soit la solution initiale admissible. Nous expliquons dans la suite de cette partie comment construire une telle méthode ainsi que celle que nous proposons pour résoudre le problème (2.14).

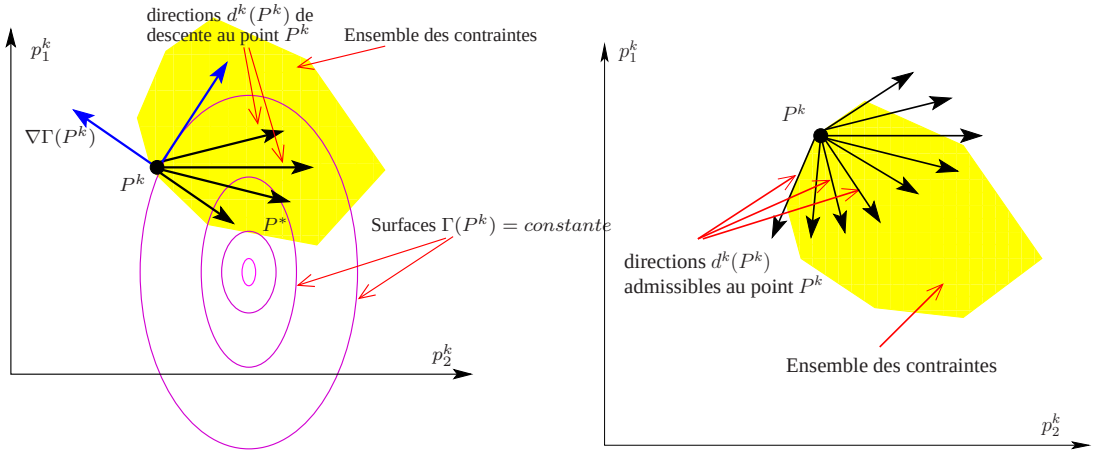
Notons $\mathbf{P}^k$ notre vecteur de solution à l'itération k, alors nous construisons la solution suivante par :

$$\mathbf{P}^{k+1} = \mathbf{P}^k - \beta^k \mathbf{d}^k(\mathbf{P}^k) \tag{2.16}$$

où $d^k(\mathbf{P}^k)$ est un vecteur de dimension identique à celle de $\mathbf{P}^k$ qui représente la direction de descente admissible et β^k un réel représentant le pas de variation dans cette direction.

Le gradient $\nabla\Gamma(\mathbf{P})$ d'une fonction continue $\Gamma(\mathbf{P})$ est connu pour être orthogonal aux surfaces isocoûts (ayant la même valeur de la fonction) et pour pointer à l'opposé des isocoûts de valeurs inférieures comme cela peut être illustré dans la figure 2.5(a). Ainsi, $\mathbf{d}^k(\mathbf{P}^k)$ est une direction de descente (permettant une diminution du coût) si son produit vectoriel avec le gradient est négatif. Il est

donc fréquent de construire la direction de descente à partir des valeurs du gradient évalué sur la solution courante $\mathbf{P}^k$.



(a) Directions de descente (b) Directions admissibles

FIG. 2.5 – Directions de descente admissibles

Pour chaque itération k, $\mathbf{P}^k$ doit être admissible au sens des contraintes (2.15). Supposons donc que $\mathbf{P}^k$ le soit pour une itération k donnée. Alors nous pouvons écrire :

$$\begin{aligned} \forall i = 1..N\ ,\ \sum_{j\neq i} p_{i,j}^{k+1} &= \sum_{j\neq i} p_{i,j}^{k} - \beta^k \sum_{j\neq i} d_{i,j}^k(P^k) \\ \text{Donc}\quad \forall i = 1..N\ ,\ \sum_{j\neq i} p_{i,j}^{k+1} = 1 &\Leftrightarrow \sum_{j\neq i} d_{i,j}^k(\mathbf{P}^k) = 0 \end{aligned} \tag{2.17}$$

La direction $\mathbf{d}^k(\mathbf{P}^k)$ doit donc être judicieusement choisie pour tenir compte des contraintes (voir figure 2.5(b). Le choix du pas β^k sera contraint pour assurer que les valeurs $p_{i,j}^{k+1}$ soient comprises entre 0 et 1. En effet, en supposant que $\forall(i,j)\ \ 0 \leq p_{i,j}^k \leq 1$, alors :

$$\begin{aligned} \forall(i,j)\ \ 0 \leq p_{i,j}^{k+1} \leq 1 \quad &\Leftrightarrow \quad -p_{i,j}^k \leq -\beta^k d_{i,j}^k(\mathbf{P}^k) \leq 1 - p_{i,j}^k \\ &\Leftrightarrow \quad \begin{cases} \beta^k \leq \dfrac{p_{i,j}^k - 1}{d_{i,j}^k(\mathbf{P}^k)} & \text{si}\ \ d_{i,j}^k(\mathbf{P}^k) < 0 \\ \beta^k \leq \dfrac{p_{i,j}^k}{d_{i,j}^k(\mathbf{P}^k)} & \text{si}\ \ d_{i,j}^k(\mathbf{P}^k) > 0 \end{cases} \end{aligned} \tag{2.18}$$

Dès lors, il existe de nombreuses variantes permettant de trouver une direction de descente admissible compte tenu des paramètres du problème traité. Le gradient projeté, le gradient conjugué, la méthode de « Frank and Wolfe » [1] sont des exemples parmi tant d'autres qui s'appuient sur le gradient de la fonction coût pour calculer la direction de descente.

Nous proposons dans la suite de cette partie de développer plus en profondeur le gradient projeté et la méthode de « Franck and Wolfe ». Cela nous permettra d'expliquer leurs problèmes de convergence lors de la résolution du problème traité.

2.3.1 Calcul du gradient pour notre problème

Nous pouvons calculer les gradients à partir de la formule du coût donnée dans (2.14) pour un couple (i,j) :

$$\frac{\partial \Gamma}{\partial p_{i,j}} = -2 \sum_t \sum_l \frac{a_{l,ij} E_i^t}{(y_l^t)^2} \left(y_l^t - \sum_{od} a_{l,od} E_{s(od)}^t p_{od} \right) \tag{2.19}$$

Notons alors $\overline{y}_l^t$ la charge évaluée sur l'interface l pour l'instant t à partir de la solution courante $\mathbf{P}^k$, nous pouvons écrire :

$$\begin{aligned} \overline{\mathbf{Y}^t} &= A\mathbf{P}^k\mathbf{E}^t \\ \overline{y}_l^t &= \sum_{ij=1}^{c} a_{l,ij} p_{ij}^k E_i^t \end{aligned} \tag{2.20}$$

Dès lors :

$$\frac{\partial \Gamma}{\partial p_{i,j}} = -2 \sum_t \sum_l \frac{a_{l,ij} E_i^t (y_l^t - \overline{y}_l^t)}{(y_l^t)^2} \tag{2.21}$$

On remarque qu'une interface utilisée par l'OD (i,j) dont l'estimation de charge est trop grande va apporter une contribution positive au gradient ce qui guidera l'algorithme à diminuer la valeur du trafic pour cet OD. Par contre, si l'estimation de charge est trop petite, alors l'algorithme aura tendance à faire décroître les trafics des couples OD passant par cette interface.

2.3.2 Gradient projeté

Le gradient projeté effectue une projection du gradient $\nabla\Gamma(\mathbf{P})$ sur les contraintes, dans le cadre de notre problème, cela revient à former la direction $\mathbf{d}^k$ de la manière suivante :

$$\forall i = 1..N \;\; \forall j = 1..N \;\; i \neq j \quad d_{i,j}^k(\mathbf{P}^k) = \frac{\partial \Gamma}{\partial p_{i,j}}(\mathbf{P}^k) - \frac{1}{N-2} \sum_{\substack{u=1 \\ u \neq j,\; u \neq i}}^{N} \frac{\partial \Gamma}{\partial p_{i,u}}(\mathbf{P}^k) \tag{2.22}$$

Calculée de telle manière à partir du gradient de la fonction, la direction $\mathbf{d}^k$ respecte bien la contrainte (2.17). Sans détailler les calculs, cela nous donne pour un couple (i,j) :

$$d_{i,j}^k = 2 \sum_t \sum_l \frac{\overline{y}_l^t - y_l^t}{(y_l^t)^2} \left(a_{l,ij} - \frac{1}{N-2} \sum_{k \neq j} a_{l,ik} \right) \tag{2.23}$$

2.3.3 « Frank and Wolfe »

Si nous utilisons la méthode de « Frank and Wolfe », alors le calcul de $\mathbf{d}^k(\mathbf{P}^k)$ est différent. Après avoir évalué le gradient de la fonction $\Gamma(\mathbf{P}^k)$ par rapport à l'ensemble des variables, on construit un point extrémal $\mathbf{P}^{k*}$ où :

$$\forall i = 1..N \quad p_{i,j}^{k*} = \begin{cases} 1 & \text{si} \quad j = \underset{l \neq i}{\text{Argmin}} \left(\dfrac{\partial \Gamma}{\partial p_{i,l}}(\mathbf{P}^k) \right) \\ 0 & \text{sinon} \end{cases} \tag{2.24}$$

La direction de descente admissible est ensuite formée par la différence vectorielle entre la solution courante et la solution extrémale : $\mathbf{d}^k(\mathbf{P}^k) = (\mathbf{P}^k - \mathbf{P}^{k*})$. Il est aussi aisé de montrer qu'ainsi construit $\mathbf{d}^k(\mathbf{P}^k)$ vérifie bien la contrainte (2.17).

2.3.4 Choix du pas

Le choix de β^k à chaque itération a une importance conséquente sur les performances de tels algorithmes. L'heuristique communément utilisée et qui donne de bons résultats est la suivante :

- Initialement : β^0 est égal au pas maximum (β_{max}) divisé par 2.
- A l'itération k : si $\beta^k > \beta_{max}$ alors $\beta^k > \beta_{max}/2$.
- A la fin de l'itération k :
  - Si le coût a augmenté : $\beta^{k+1} = \beta^k * 0.8$
  - Sinon : $\beta^{k+1} = \beta^k * 1.2$

Cependant, la convergence théorique vers la solution optimale quelle que soit la solution initiale admissible est obtenue par un choix optimal de la valeur de β à utiliser à chaque itération. Il est donc difficile d'implémenter un algorithme basé sur l'une des techniques proposées convergeant réellement de manière certaine vers la solution optimale. En pratique, nous avons implémenté les deux méthodes proposées et sur les nombreux exemples traités, malgré le calcul du pas β permettant à chaque itération une diminution maximale du coût, les algorithmes ont tendance à se bloquer sur les contraintes (un $p_{i,j}$ nul). Nous proposons donc une heuristique efficace permettant de contourner ce problème.

Remarque 2.1 *Une étude approfondie des problèmes de convergence des méthodes présentées permet de pointer du doigt le problème rencontré. Un lien utilisé par un petit nombre de couples OD issus de la même source (de l'ordre de 2 ou 3) et dont la charge est très surestimée aura pour conséquence de faire diminuer les variables de ces couples dans des quantités identiques. De ce fait, si l'une de ces variables est plus petite que les autres, elle limitera la variation possible et bloquera l'algorithme qui cherche à faire décroître la charge de ce lien. De manière plus générale, le couple (u, v) ayant le gradient le plus grand peut être associé à une variable p_{uv} très petite et dans ce cas, l'algorithme se bloque. Pour éviter cela, il faudrait choisir une direction de descente guidée par les gradients extrêmes des couples (u, v) associés à des variables p_{uv} ni trop petites, ni trop grandes. C'est justement ce que nous proposons dans la partie suivante.*

2.4 Algorithme de recherche locale pour résoudre le problème de minimisation

Pour palier aux problèmes de convergence et pour permettre une résolution plus rapide, nous proposons d'utiliser une heuristique basée sur une technique de recherche locale. En fait, le problème rencontré plus tôt sur la convergence des méthodes de descente classiques peut être contourné par un calcul plus judicieux de la direction de descente. La méthode que nous proposons doit être vue comme une méthode de direction de descente admissible, cependant le calcul de la direction de descente ainsi que celui du pas peuvent être modélisés par une méthode de recherche locale.

L'heuristique proposée forme une suite de solutions admissibles telles que pour chaque itération k, la solution $\mathbf{P}^{k+1}$ est choisie dans le voisinage V^k de la solution $\mathbf{P}^k$ et vérifie $\Gamma(\mathbf{P}^{k+1}) < \Gamma(\mathbf{P}^k)$.

Dans les parties suivantes, nous allons donc tout d'abord décrire la structure de voisinage que nous utilisons et ses différentes propriétés, puis nous détaillerons l'exploration de ce voisinage. Ensuite, nous expliquerons rapidement l'étape d'initialisation de l'algorithme avant d'expliquer de manière plus précise l'algorithme que nous proposons.

2.4.1 Définition du voisinage

Le voisinage que nous proposons permet d'éviter les problèmes de convergence expliqués en partie par la remarque 2.1. Nous notons V^k le voisinage de la solution $\mathbf{P}^k$, dès lors une solution $\mathbf{P}$ appartient au voisinage V^k si et seulement s'il existe deux couples OD (u, v) et (u, w) différents tels que les conditions suivantes sont vérifiées :

$$\begin{cases} p^k_{uv} & > & \delta^k & \\ p^k_{uw} & < & 1-\delta^k & \\ p_{uw} & = & p^k_{uw}+\delta^k & \text{Si } ij = uw \\ p_{uv} & = & p^k_{uv}-\delta^k & \text{Si } ij = uv \\ p_{ij} & = & p^k_{ij} & \forall ij, \quad ij \neq uv, \quad ij \neq uw \end{cases} \tag{2.25}$$

La figure 2.6 illustre la construction d'un voisin (noté $\mathbf{P}(u, v, w)$) où le coefficient $p_{u,v}$ est diminué de δ^k et le coefficient $p_{u,w}$ est augmenté de δ^k. Le voisin considéré est donc construit en déviant du trafic du couple OD (u, v) vers le couple OD (u, w). Les contraintes (2.15) sont donc vérifiées pour chaque solution de V^k à partir du moment où $\mathbf{P}^k$ les vérifie également. Il faut remarquer ici que la quantité déviée vaut δ^k, l'utilité et l'évolution de ce paramètre seront données plus loin.

Si nous revenons à la notion de gradient vue plus tôt, le voisin $\mathbf{P}(u, v, w) \in V^k$ permettant la meilleure diminution de coût est tel que le triplet (u, v, w) maximise la différence de gradient donnée par l'équation (2.26) sous les contraintes $p_{uv} \geq \delta^k$ et $p_{uw} \leq 1 - \delta^k$. Le choix du voisin est donc totalement lié aux gradients évalués pour la solution courante comme les méthodes de direction de descente admissible.

$$\frac{\partial \Gamma}{\partial p_{uw}}(\mathbf{P}^k) - \frac{\partial \Gamma}{\partial p_{uv}}(\mathbf{P}^k) \tag{2.26}$$

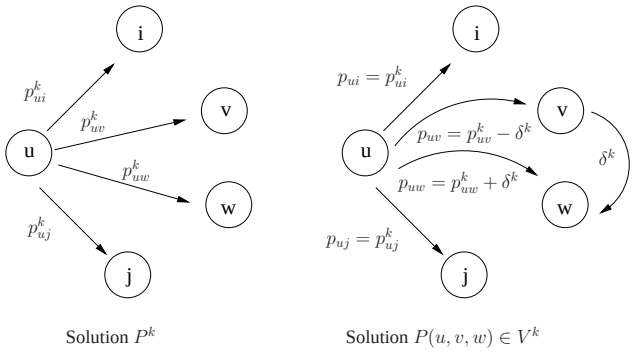


FIG. 2.6 – Construction du voisin $P(u, v, w)$

L'heuristique proposée explore la totalité du voisinage ainsi construit et dont la taille est de l'ordre de N^3. Il est cependant possible de diminuer les temps de calculs nécessaires à l'exploration du voisinage en évitant un calcul complet des caractéristiques d'une solution (propagation des trafics, évaluation du coût). Nous proposons d'utiliser une relation simple entre le voisin étudié et la variation de coût $(\Gamma(\mathbf{P}) - \Gamma(\mathbf{P}^k))$ à la solution courante $\mathbf{P}^k$.

2.4.2 Evaluation des voisins

Soit $\mathbf{P}$ une solution admissible, et soit $\overline{\mathbf{Y}}^t$ les charges associées à cette solution pour tout instant t $(1 \leq t \leq \tau)$ (cf. équation (2.20)). Soit un couple OD uv, et soit $\mathbf{P}^\delta$ la solution $\mathbf{P}$ modifiée où le paramètre p_{uv} est incrémenté d'une quantité δ_{uv}. En notant $\Delta\Gamma_{uv}$ la variation de coût entre $\mathbf{P}$ et $\mathbf{P}^\delta$, nous obtenons :

$$
\begin{aligned}
\Delta\Gamma_{u,v} &= \Gamma(\mathbf{P}^\delta) - \Gamma(\mathbf{P}) = \sum_{t=1}^{\tau} [\|\mathbf{Y}^t - A\mathbf{P}^\delta\mathbf{E}^t\|^2 - \|\mathbf{Y}^t - A\mathbf{P}\mathbf{E}^t\|^2] \\
&= \sum_{t=1}^{\tau}\sum_{l=1}^{L} \frac{1}{(y_l^t)^2} \left((y_l^t - \sum_{od=1}^{c} a_{l,od} p^\delta_{od} E^t_{s(od)})^2 - (y_l^t - \sum_{od=1}^{c} a_{l,od} p_{od} E^t_{s(od)})^2 \right) \\
&= \sum_{t=1}^{\tau}\sum_{l=1}^{L} \left(\frac{(y_l^t - \overline{y}_l^t + a_{l,uv}\delta_{uv}E^t_u)^2 - (y_l^t - \overline{y}_l^t)^2}{(y_l^t)^2} \right) \\
&= \sum_{t=1}^{\tau}\sum_{l \in L_{uv}} \left(\frac{\Delta y^t_{uv,l}(2(\overline{y_l^t} - y_l^t) + \Delta y^t_{uv,l})}{(y_l^t)^2} \right) \qquad (2.27) \\
\text{Avec ,}\ \Delta y^t_{uv,l} &= \delta_{uv}\, a_{l,uv}\, E^t_u \quad \forall l \in L_{uv}
\end{aligned}
$$

Où L_{uv} contient les liens utilisés pour router du trafic entre u et v c'est à dire les liens l tels que $a_{l,uv} \neq 0$. En fait, cette formule permet de calculer la variation de coût en identifiant les rares liens dont la charge a changé suite à la modification de $\mathbf{P}$.

La variation de coût associée à un voisin $\mathbf{P}(u, v, w) \in V^k$ est donc égale à $\Delta\Gamma_{u,w} - \Delta\Gamma_{u,v}$ avec des variations respectives $\delta_{uv} = -\delta^k$ et $\delta_{uw} = \delta^k$. En évitant la majorité des liens où rien ne change,

on accélère grandement les temps de calcul ce qui est nécessaire vu la taille du voisinage.

Si nous supposons que la variation δ^k est suffisamment petite, nous pouvons proposer une approximation au premier ordre de la variation de coût associée à chaque voisin $P(u,v,w) \in V^k$ par :

$$\begin{aligned}
\Delta\Gamma_{u,w} - \Delta\Gamma_{u,v} &= \sum_{t=1}^{\tau}\sum_{l\in L_{uw}} \frac{2\Delta y^t_{uw,l}(\overline{y^t_l} - y^t_l)}{(y^t_l)^2} - \sum_{t=1}^{\tau}\sum_{l\in L_{uv}} \frac{2\Delta y^t_{uv,l}(\overline{y^t_l} - y^t_l)}{(y^t_l)^2} \\
&= \sum_{t=1}^{\tau}\sum_{l=1}^{L} 2\delta^k E^t_u \left(a_{l,uw} - a_{l,uv}\right) \frac{(\overline{y^t_l} - y^t_l)}{(y^t_l)^2} \\
&= \delta^k \left(\frac{\partial\Gamma}{\partial p_{uw}}(\mathbf{P}^k) - \frac{\partial\Gamma}{\partial p_{uv}}(\mathbf{P}^k) \right) \qquad (2.28)
\end{aligned}$$

Nous sommes donc sûr que l'algorithme a convergé à l'itération k dès lors qu'il n'existe aucun triplet (u,v,w) dont la différence des gradients entre le couple (u,w) et le couple (u,v) est négative.

On se rend compte que les liens communs aux deux couples OD (u,v) et (u,w) ne sont pas pris en compte dans le calcul de la variation de coût (2.28). On évite ainsi le problème énoncé dans la remarque 2.1 ce qui permet d'obtenir la convergence. De plus, le fait de savoir que la variable p_{uv} choisie est supérieure à δ^k permet d'éviter d'autant plus les problèmes de convergence. Lors de nos tests, cette heuristique n'a jamais été bloquée par les contraintes et a toujours convergé vers le minimum global du problème.

Remarque 2.2 *Nous pouvons faire un parallèle avec les méthodes de descente. Une itération de l'algorithme proposé consiste à trouver les deux couples (u,w) et (u,v) tels que la différence entre le gradient du premier et le gradient du second soit la plus petite possible. Ensuite, la direction de descente choisie pour l'itération est le vecteur d^k tel que $d^k_{uv} = -1$, $d^k_{uw} = 1$ et $d^k_{ij} = 0$ pour les autres couples ij. Le pas effectué dans cette direction est δ^k et la solution suivante est donc le voisin $\mathbf{P}(u,v,w)$. L'heuristique proposée décrite par des techniques de recherche locale peut donc aussi être vue comme une méthode de descente mieux adaptée au problème traité que les classiques méthodes du gradient projeté ou de « Frank and Wolfe ». Cependant, le fait de connaître à l'avance le pas qui sera utilisé permet d'éviter de choisir un couple (u,v) tel que sa variation le rendrait négatif. Ce n'est donc pas le meilleur triplet global qui est choisi mais le meilleur triplet permettant d'appliquer la variation demandée.*

2.4.3 Initialisation

L'efficacité de l'algorithme proposé dépend grandement de la solution initiale utilisée. Contrairement aux solutions aisément calculables comme une solution équirépartie ($p_{ij} = \frac{1}{N-1}$), la solution initiale admissible proposée ici permet une convergence rapide de l'heuristique proposée.

La solution initiale doit être reliée à la formulation du voisinage utilisé. Aussi, nous proposons de la construire en plaçant itérativement et à moindre coût des fragments (quantums) de quantité Q sur les

différents couples. Partant d'un vecteur solution nul, chaque itération incrémente de Q la valeur p_{ij} du couple OD (i, j) tel que l'augmentation de coût soit minimum. L'algorithme 1, nommé *Mille Feuilles*, illustre le pseudo code de la procédure construisant la solution initiale admissible.

algorithm 1 Mille Feuilles

```
1:  procedure InitMilleFeuille
2:      ∀i = 1..c   p_ij = 0 , ∀i = 1..N   q(i) = 0 , Δ_min = 0 , Γ = 0
3:      while Σ_i q(i) < N/Q do                ▷ Tant que tous les quantums ne sont pas placés
4:          for i = 1 to N do                  ▷ Pour chaque origine
5:              if q(i) < 1/Q then
6:                  for j = 1 to N , j ≠ i do   ▷ Pour chaque destination différentes de i
7:                      Evaluer ΔΓ_ij avec δ_ij = δ_ij + Q
8:                      if ΔΓ_ij < Δ_min then
9:                          i* = i , j* = j , Δ_min = ΔΓ_ij
10:                     end if
11:                 end for
12:             end if
13:         end for
14:         p_i*j* = p_i*j* + Q                ▷ Ajouter Q là où cela augmente le moins le coût
15:         q(i*) = q(i*) + 1                  ▷ Compte combien de quantums ont été ajoutés à l'origine i
16:         Calculer Γ(P)                      ▷ Met à jour le coût
17:     end while
18: end procedure
```

La boucle principale de l'algorithme 1 parcourt tous les couples OD (i, j) possibles (ligne 3 et ligne 5) et évalue la variation de coût suite à la modification de $p_{i,j}$ incrémenté de Q. L'équation (2.27) utilisée avec $\delta_{ij} = +Q$ permet une évaluation rapide. Suite à cette boucle principale, le couple OD i^*j^* qui entraîne la plus petite augmentation de coût est retenu et appliqué.

Les valeurs $q(i)$ représentant le nombre de quantums placés sur la source i sont maintenues à jour et utilisées à la ligne 4 de l'algorithme pour éviter de placer trop de quantums sur une même origine. De plus, la boucle principale stoppe quand la somme de tous les $q(i)$ est égale à $\frac{N}{Q}$ ce qui veut dire que pour chaque origine i, $q(i) = \frac{1}{Q}$. Dès lors, en notant n_{ij} le nombre de quantum placés sur le couple OD ij, nous pouvons vérifier la contrainte de conservation des flots pour une origine i donnée :

$$\sum_{j \neq i} p_{ij} = \sum_{j \neq i} (n_{ij} \times Q) = Q \sum_{j \neq i} n_{ij} = Q \times q(i) = 1$$

Cet algorithme permet donc d'obtenir une solution initiale admissible telle que la manière dont elle est construite s'adapte très bien à notre voisinage. En effet, dans notre heuristique principale permettant une résolution du problème (2.14), la valeur de δ^k définissant en partie le voisinage à l'itération k sera initialisée à une valeur inférieure à Q. La solution initiale proposée garantissant qu'aucune solution ne peut être meilleure dans son voisinage si $\delta^k \geq Q$.

2.4.4 Algorithme des Fanouts Généralisés : GFO

Nous proposons dans cette partie une description précise de l'algorithme basé sur des techniques de recherche locale que nous avons développé. Nous l'avons nommé GFO pour « Generalized FanOut ». Le pseudo-code de cet algorithme est décrit par l'algorithme 2.

algorithm 2 GFO

1: **procedure** MATRICEESTIMATION
2: $\mathbf{P}^0 = \text{MilleFeuille}(Q)$, $\Gamma^* = \Gamma(\mathbf{P}^0)$ $k = 0$, $\delta_0 = \frac{Q}{2}$ ▷ Initialisation
3: **while** not needToStop() **do**
4: better = $false$
5: **for all** $\mathbf{P} \in V_k$ **do** ▷ Pour tous les voisins de $\mathbf{P}^k$
6: Evaluate $\Gamma(\mathbf{P})$
7: **if** $\Gamma(\mathbf{P}) < \Gamma^*$ **then** $\Gamma^* = \Gamma(\mathbf{P})$, $\mathbf{P}_{min} = \mathbf{P}$, better $= true$
8: **end if**
9: **end for**
10: **if** better **then**
11: $\mathbf{P}^{k+1} = \mathbf{P}_{min}$, $\delta^{k+1} = \delta^k$
12: **else**
13: $\mathbf{P}^{k+1} = \mathbf{P}^k$, $\delta^{k+1} = \frac{\delta^k}{2}$
14: **end if**
15: **end while**
16: **end procedure**

Une première phase d'initialisation (ligne 1) permet de créer la solution initiale à l'aide de l'algorithme « Mille Feuilles » (Algorithme 1). La valeur de Q qui est utilisée dans notre implémentation est 2.10^{-4} et ce choix provient principalement de tests effectués. La manière dont est construite la solution initiale nous assure qu'en prenant $\delta^0 \geq Q$, aucune solution de coût inférieur ne sera trouvée dans le voisinage de P^0. C'est pourquoi nous démarrons notre recherche avec une valeur $\delta^0 = \frac{Q}{2}$.

Ensuite, dans la boucle principale entre les lignes 2 et 14, l'exploration du voisinage de la solution courante est effectuée (lignes 4 à 8). Le booléen *better* nous permet de savoir si un meilleur coût a été trouvé lors de cette exploration. Dès lors, deux cas se présentent (lignes 9 à 13) :

Cas 1 : Soit *better* vaut vrai : dans ce cas, une solution qui diminue strictement le coût a été trouvée dans le voisinage. Cette solution est $\mathbf{P}_{min}$ donc on choisi $\mathbf{P}^{k+1} = \mathbf{P}_{min}$ et $\delta^{k+1} = \delta^k$. En effet, comme une meilleure solution a été trouvée, il n'est pas nécessaire de diminuer le pas d'exploration δ^k.

Cas 2 : Soit *better* vaut faux : cela veut dire qu'aucun des voisins ne permet une diminution stricte du coût. Dans ce cas, nous allons garder la solution courante $\mathbf{P}^{k+1} = \mathbf{P}^k$ mais nous allons diminuer la valeur du pas $\delta^{k+1} = \frac{\delta^k}{2}$.

La fonction *needToStop()* retournera vrai si et seulement si une des conditions suivantes est vérifiée

Condition 1 : δ^k devient trop petit :

$$\delta^k \leq \frac{10^{-6}}{N-1} \tag{2.29}$$

Condition 2 : L'erreur relative maximale sur les charges est inférieure à 1% :

$$\max_{\substack{1 \leq l \leq L \\ 1 \leq t \leq \tau}} \frac{|y_l^t - \overline{y_l^t}|}{y_l^t} \leq 1\% \tag{2.30}$$

Condition 3 : Convergence absolue :

$$\| \quad \frac{\partial \Gamma}{\partial p_{uw}}(\mathbf{P}^k) - \frac{\partial \Gamma}{\partial p_{uv}}(\mathbf{P}^k) \quad \| \leq \epsilon \quad \forall \mathbf{P}(u,v,w) \in V^k \tag{2.31}$$

Remarque 2.3 *L'équation (2.28) montre que si la différence des gradients entre les couples (u,w) et (u,v) est nulle pour tous les voisins $\mathbf{P}(u,v,w) \in V^k$ alors l'algorithme a convergé. Techniquement, cette convergence est difficile à atteindre car elle nécessite des temps de calculs ne supportant pas le passage à l'échelle. Il est cependant important de tester ce critère de convergence car c'est le seul qui soit exact. L'utilisation d'une valeur positive ϵ petite permet donc de rendre ce test effectif.*

Cet algorithme a été développé et testé dans le logiciel NEST. Le protocole de test ainsi que les résultats obtenus sont présentés juste après.

2.5 Tests et Résultats

Des données SNMP réelles ainsi que les trafics les ayant générés sont difficiles à trouver. Nous avons décidé de tester notre approche sur des données simulées. Pour cela, nous avons simulé des matrices de trafic respectant notre hypothèse principale de faible variation des fanouts sur une courte durée, puis nous avons propagé les trafics au sein d'un réseau virtuel en utilisant NEST. Le protocole de test que nous avons suivi peut donc se résumer ainsi

1. Construire un réseau et récupérer sa matrice de routage A (outil NEST).
2. Générer des matrices de trafic X^t
3. Propager ces trafics et récupérer les charges des liens $\mathbf{Y}^t$ (outil NEST).
4. Exécuter l'algorithme GFO avec A et $\mathbf{Y}^t$ comme données initiales
5. Comparer la solution obtenue avec l'original X^t

Nous expliquerons tout d'abord dans ce chapitre comment nous construisons des matrices de trafic respectant une variation plus ou moins faible des fanouts, puis dans une seconde partie, nous donnerons les différentes caractéristiques des réseaux de test, enfin nous donnerons les résultats numériques obtenus.

2.5.1 Génération des matrices de trafic originales

Pour pouvoir générer autant de matrices de trafic que de mesures nécessaires, une première matrice X^0 est construite. Ensuite, pour chaque instant de mesure t, nous perturbons la matrice X^0 pour obtenir une matrice X^t. La matrice X^0 ne sera donc pas utilisée lors de nos estimations, seules les mesures $t = 1..\tau$ seront utilisées.

La plupart des articles portant sur l'analyse des trafics affirment que seulement 30% des flots génèrent 80% de la charge totale sur le réseau. De plus, il existe le plus souvent des gros flots, des flots moyens et des petits flots. Nous avons donc décidé de générer notre matrice de trafic initiale selon une distribution multimodale : une somme pondérée de distributions gaussiennes. En notant $G(\lambda, \sigma)$ une distribution gaussienne de moyenne λ et d'écart type σ, alors la distribution de probabilité des x_{ij}^0 est :

$$P = \sum_{k=1}^{3} p_k G(\lambda_k, \sigma_k)$$

$$\begin{pmatrix} p_1 = 0.3 & \lambda_1 = 50000 & \sigma_1 = \frac{\lambda_1}{10} = 5000 \\ p_2 = 0.2 & \lambda_2 = 10000 & \sigma_2 = \frac{\lambda_2}{10} = 1000 \\ p_3 = 0.5 & \lambda_3 = 500 & \sigma_3 = \frac{\lambda_3}{10} = 50 \end{pmatrix}$$

Les gaussiennes ou lois normales $G(\lambda, \sigma)$ ont des particularités bien connues : 99.7% des valeurs sont comprises dans l'intervalle $\lambda \pm 3\sigma$, 95% des valeurs sont comprises dans l'intervalle $\lambda \pm 1.96\sigma$ et 68% des valeurs sont comprises dans l'intervalle $\lambda \pm \sigma$.

50% des flots sont des petits c'est à dire avec une demande moyenne de 500Kbps et des valeurs proches de cette moyenne, 20% seront des flots moyens avec une demande moyenne de 10000Kbps et des valeurs un peu plus étalées autour de cette moyenne, et 30% des flots seront des gros avec une demande moyenne de 50000Kbps et des valeurs très étalées.

La figure 2.7 donne une illustration de la fonction de distribution cumulative (cdf) de cette loi de probabilité.

A partir de cette première matrice X^0, on récupère des valeurs initiales pour les trafics entrant en chaque nœud ($\mathbf{E}^0$) et pour les fanouts ($\mathbf{P}^0$) :

$$\begin{aligned} E_i^0 &= \sum_{j \neq i} x_{ij}^0 \\ p_{ij}^0 &= \frac{E_i^0}{x_{ij}^0} \end{aligned}$$

Ces valeurs seront en fait les moyennes respectives de ces paramètres au cours du temps. Ensuite, pour obtenir une matrice de trafic pour chaque instant t, la procédure suivante est exécutée :

1. Evolution des trafics entrant dans le réseau par le nœud i par l'ajout d'un bruit gaussien centré :

$$\forall i = 1..N \qquad E_i^t = E_i^0 + G(0, \frac{E_i^0}{10})$$

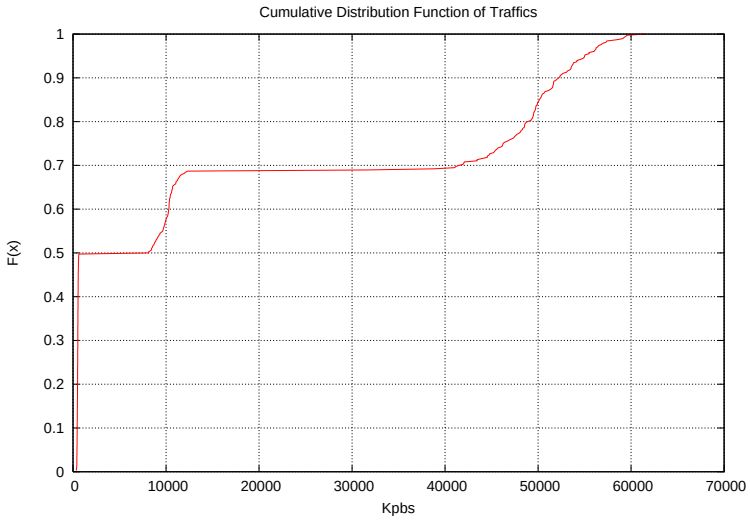


FIG. 2.7 – cdf de la loi de probabilité des trafics

Plus les trafics sont différents à chaque mesure, plus l'information ajoutée est grande. Avec de telles valeurs, les variations relatives appliquées sur les trafics E_i^t par rapport à E_i^0 sont en moyenne nulles et principalement comprises entre $\pm$ 10% (pour 65% des cas) et majoritairement comprises entre $\pm$ 30% (pour 99.7% des cas). Cette hypothèse limite donc l'information apportée par chaque mesure, des variations plus grandes seraient plus favorables mais cela ne nous semble pas réaliste d'augmenter ces variations.

2. Evolution des fanouts par l'ajout d'un bruit gaussien centré puis normalisation :

$$\forall ij = 1..c \quad z_{ij}^t = \quad p_{ij}^0 + G(0, p_{ij}^0 \times pVar)$$
$$\forall ij = 1..c \quad p_{ij}^t = \quad \frac{z_{ij}^t}{\sum_k z_{ik}^t}$$

L'analyse des écarts relatifs des fanouts par rapport à leur moyenne peut être faite de la même manière que pour les trafics E_i^t. Ainsi, ces variations seront comprises entre $\pm$ pVar pour 65% des cas et majoritairement comprises entre $\pm$ 3pVar (pour 99.7% des cas). L'hypothèse de faibles variations des fanouts suppose que ces variations sont de l'ordre du pourcent. C'est pourquoi la valeur de pVar sera souvent 1%. Nous testerons tout de même notre algorithme pour des valeurs supérieures.

3. Calcule de la matrice de trafic à l'instant t avec :

$$\forall ij = 1..c \qquad x_{ij}^t = p_{ij}^t \times E_i^t$$

Nous connaissons donc les trafics réels dont sont issues les charges sur lesquelles nous travaillerons. Même si ces trafics sont hypothétiques et qu'un travail sur données réelles apporterait une meilleure validation de la méthode et de l'hypothèse, le fait de les connaître exactement permet de tester l'efficacité et la robustesse de notre méthode.

Suite à cette étape de génération, τ matrices de trafic sont disponibles. Nous les propageons sur le réseau associé pour obtenir une série de τ vecteurs de charges sur les liens. Ces vecteurs de charges seront nos données SNMP simulées.

2.5.2 Réseaux de test

Les tests effectués ont été faits sur des réseaux de tailles et de topologies différentes. Ces réseaux sont en fait construits à l'aide du logiciel NEST ce qui nous permet de tester notre algorithme sur n'importe quel type de topologie. Six réseaux sont utilisés pour nos tests et ils sont illustrés sur la figure 2.8. Le tableau 2.1 résume les principales propriétés de ces réseaux. Cependant, par soucis de clarté, nous allons tout d'abord donner des résultats obtenus sur les réseaux *net4* et *net5* afin de les commenter de manière approfondie, puis nous exposerons l'ensemble de nos résultats.

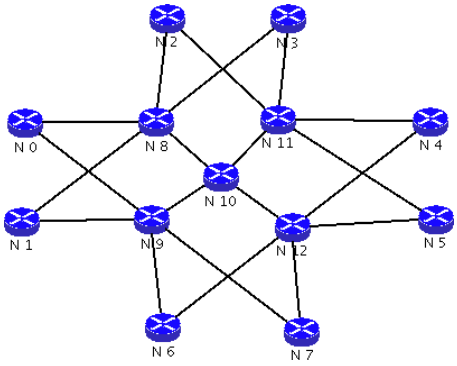
Nom	#Routeurs	#Routeurs Frontières	#OD	#Interfaces
net1	13	8	56	40
net2	18	10	90	56
net3	20	12	132	66
net4	28	15	210	98
net5	38	20	380	142
net6	45	35	1190	166

TAB. 2.1 – Particularité des réseaux testés

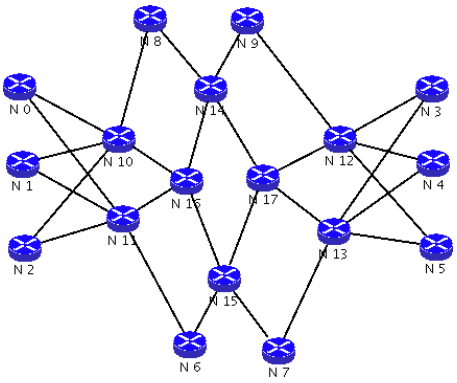
Les matrices de routage associées sont calculées à l'aide de NEST suite à une configuration du protocole OSPF avec toutes les métriques égales à 1. Suite aux tests effectués sur les méthodes de descente, nous nous sommes rendu compte que l'utilisation de routes uniques était l'hypothèse la plus défavorable. Aussi, même si un routage par partage de charge sur les plus courts chemins est celui utilisé par défaut dans NEST, nous avons limité son utilisation à une seule route. Si l'on regarde attentivement les rapports entre les nombres de couples OD et les nombres d'interfaces, on se rend compte que tous les problèmes traités sont extrêmement sous-contraints avec un rapport supérieur à 2 pour la plupart d'entre eux.

2.5.3 Résultats numériques

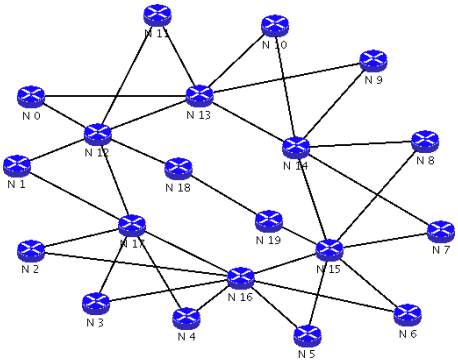
Il existe de nombreux travaux portant sur l'estimation des matrices de trafic et dans la plupart de ces travaux, deux critères principaux sont souvent utilisés pour pouvoir évaluer l'efficacité des méthodes proposées



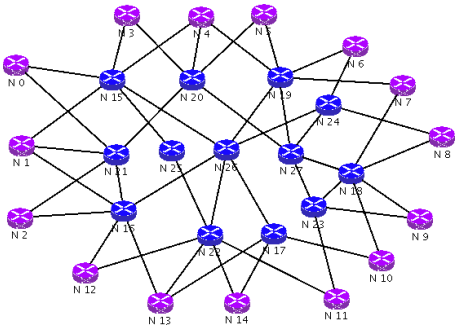
(a) *net1*



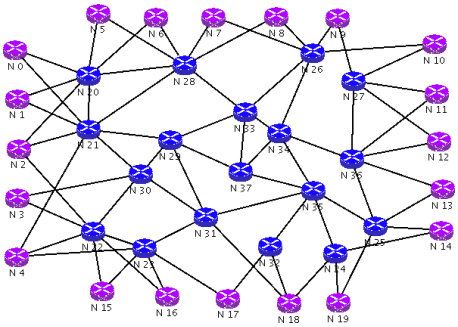
(b) *net2*



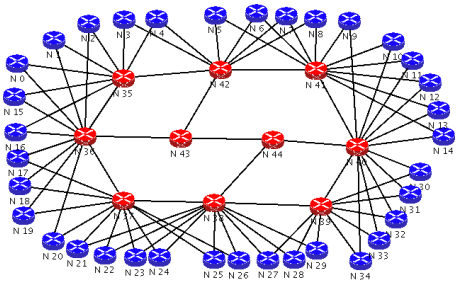
(c) *net3*



(d) *net4*



(e) *net5*



(f) *net6*

FIG. 2.8 – Topologies de tests

– L'erreur spatiale pour chaque couple OD uv : elle résume l'erreur sur le couple OD uv au cours du temps.

$$errS_{uv} = \frac{\sqrt{\sum_{t=1}^{\tau} (x_{uv}^t - p_{uv}E_u^t)^2}}{\sqrt{\sum_{t=1}^{\tau} {x_{uv}^t}^2}} \tag{2.32}$$

– L'erreur temporelle pour chaque mesure t : elle résume l'erreur globale sur tous les couples OD pour un instant t.

$$errT_t = \frac{\sqrt{\sum_{ij=1}^{c} (x_{ij}^t - p_{ij}E_i^t)^2}}{\sqrt{\sum_{ij=1}^{c} {x_{ij}^t}^2}} \tag{2.33}$$

De notre côté, étant donnée l'approche choisie, nous proposons de regarder de plus près l'estimation de la matrice de trafic moyenne. Notons pour cela $\overline{E_i}$ la valeur moyenne au cours des mesures des variables E_i^t. De la même manière, notons $\overline{p_{ij}}$ la moyenne des p_{ij} réels sur les mesures et $\overline{x_{ij}}$ celle des trafics réels x_{ij}. Il nous parait alors intéressant de chiffrer l'erreur faite sur les valeurs moyennes des trafics et des fanouts puisque nous cherchons à minimiser la moyenne des erreurs quadratiques sur les charges.

Nous considérerons donc l'écart relatif moyen sur l'ensemble des couples OD entre les trafics moyens réels et les trafics moyens estimés :

$$ER_X = \frac{1}{c}\sum_{ij=1}^{c}(1 - \frac{\frac{1}{\tau}\sum_{t=1}^{\tau} p_{ij}E_i^t}{\overline{x_{ij}}}) = \frac{1}{c}\sum_{ij=1}^{c}(1 - \frac{p_{ij}\overline{E_i}}{\overline{x_{ij}}}) \tag{2.34}$$

ainsi que l'écart relatif moyen sur l'ensemble des couples OD entre les fanouts moyens réels et les fanouts estimés :

$$ER_P = \frac{1}{c}\sum_{ij=1}^{c}(1 - \frac{p_{ij}}{\overline{p_{ij}}}) \tag{2.35}$$

Enfin, nous proposons également le calcul de la moyenne sur l'ensemble des couples OD et sur l'ensemble des mesures des écarts relatifs entre les trafics réels et estimés. Nous avons noté ER cette erreur et nous avons donc :

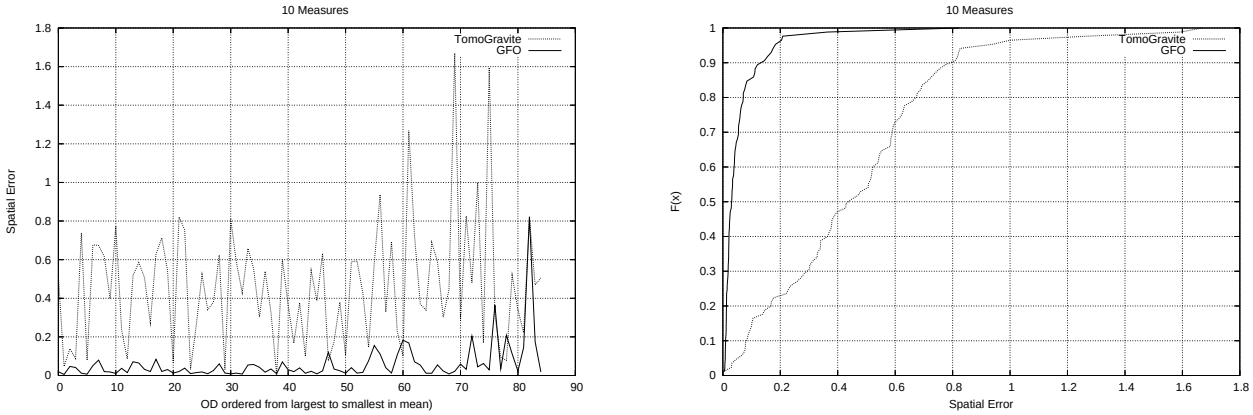
$$ER = \sum_{t=1}^{\tau}\frac{1}{\tau}\sum_{uv=1}^{c}\frac{1}{c}\frac{(x_{uv}^t - p_{uv}E_u^t)}{x_{uv}^t} \tag{2.36}$$

Remarque 2.4 *Les « petits » trafics sont très difficiles à estimer car leur impact est très faible au niveau des charges. Il est donc usuel de ne tenir compte que des plus « gros » flots lors de l'évaluation des différents critères proposés. Ainsi, nous ne donnerons des résultats que pour les flots, triés par ordre décroissant, dont la somme des charges représente 95% de la charge totale. Si nous revenons aux trafics que nous traitons, cela veut dire qu'une partie des flots dont la moyenne est autour de 500Kbps peuvent être mal estimés sans que nos résultats n'en pâtissent. L'impact de ces flots sur la charge des interfaces est minime par rapport aux autres flots dont la moyenne est au moins 20 fois supérieure. Cependant, il doit être noté que cette pratique peut fausser les résultats si l'ensemble des flots ont à peu près la même valeur de trafic.*

La méthode de référence dans le domaine est la tomogravité de Zhang et Al. [70] car celle-ci s'appuie sur des hypothèses qui sont selon l'auteur souvent vérifiées pour les réseaux de transit. Nous proposons de comparer notre algorithme à la tomogravité dont les spécifications sont données dans l'état de l'art (2.1.2), et cela même si les matrices proposées ne suivent pas forcement le modèle proposé par Zhang et Al.

Pour pouvoir avoir un point de comparaison avec l'ensemble des méthodes existantes, il est possible de se référer à deux articles les comparant. L'article de Soule et Al. [82] donne des résultats sur la tomogravité, la méthode de changement de métrique, les fanouts, PCA et Kalman. On peut également trouver dans l'article de Vaton et Al [84] de bonnes comparaisons sur toutes les méthodes statistiques basées sur des algorithmes EM.

Résultats pour $pVar = 1\%$



(a) Erreur Spatiale sur *net4*

(b) Cdf de l'Erreur Spatiale sur *net4*

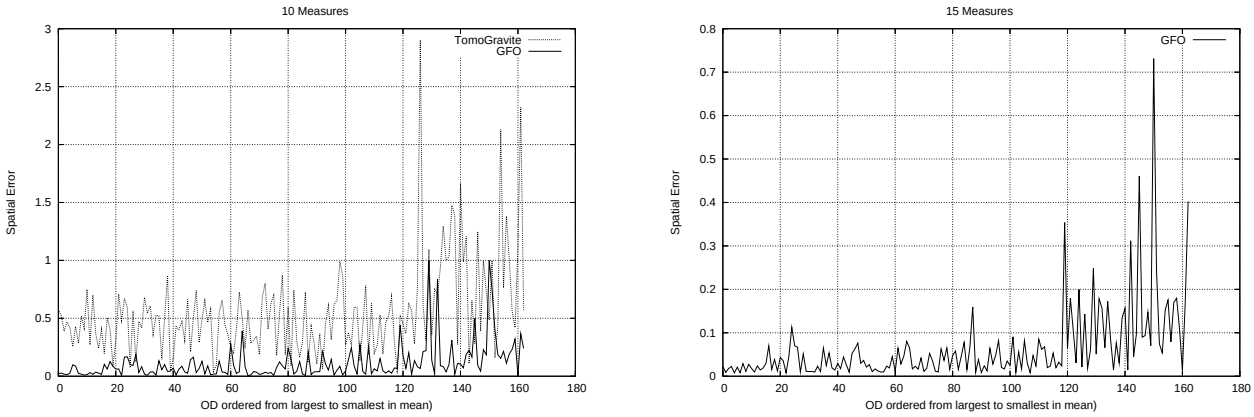
FIG. 2.9 – Résultats pour *net4* avec 10 mesures

Dans les figures 2.9(a) et 2.10(a) sont illustrées les erreurs spatiales respectives pour *net4* et *net5* avec $\tau = 10$ mesures. Les erreurs obtenues sont vraiment très bonnes comparées aux résultats que l'on

peut trouver dans la littérature. Nous n'excédons pas les 20% d'erreur excepté pour les plus petits trafics pour *net4*. La figure 2.9(b) montre d'une manière plus visible la fonction cumulative de distribution pour cette erreur. En ce qui concerne le second réseau de test, les résultats sont un peu moins bons mais cela est dû à un trop faible nombre de mesures compte tenu de la taille du réseau. En terme d'erreur relative, on trouve

Sur *net4* : $ER = 20.5\%$, $ER_X = 0.8\%$, $ER_P = 8.3\%$
Sur *net5* : $ER = 27.5\%$, $ER_X = 1.3\%$, $ER_P = 11.8\%$

Cela montre que même si l'erreur relative moyenne sur les mesures et les trafics reste élevée, l'erreur sur les fanouts moyens est faible et l'erreur sur les trafics moyen l'est encore plus. L'estimation de la matrice moyenne est donc efficace pour ces deux exemples.



(a) Erreur Spatiale sur *net5* avec 10 mesures

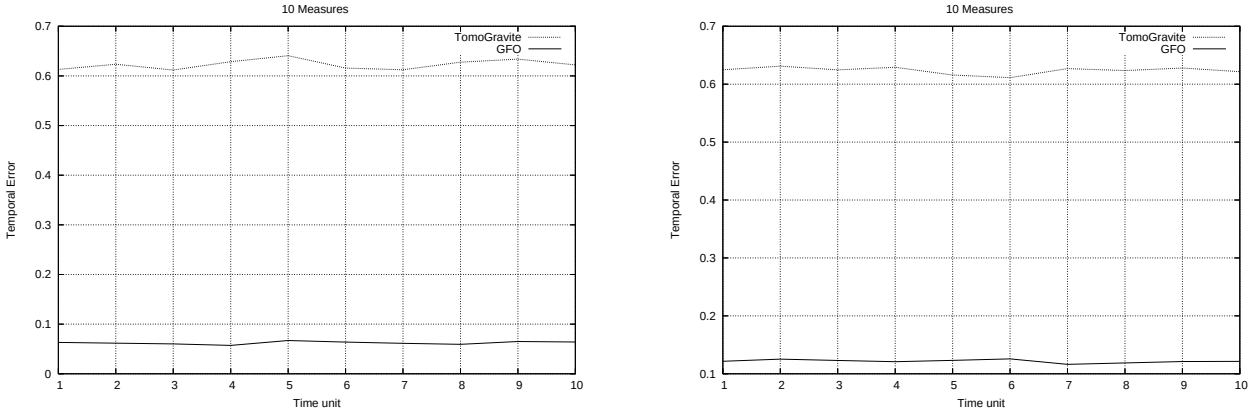
(b) Erreur Spatiale sur *net5* avec 15 mesures

FIG. 2.10 – Résultats pour *net5* avec 10 et 15 mesures

Si l'on regarde plus précisément la figure 2.10(b) où nous n'avons tracé que l'erreur spatiale pour $\tau = 15$ mesures, nous pouvons voir que les erreurs sont la plupart du temps inférieures à 10%. Les valeurs autour de 20%-30% sont atteintes pour les plus petits flots. En ayant augmenté le nombre de mesures, nous trouvons en terme d'erreur moyenne des valeurs plus faibles comparées à celles obtenues avec 10 mesures : $ER = 26.2\%$, $ER_X = 0.8\%$, et $ER_P = 11.2\%$. Nous nous rendons compte ici que le nombre de mesures a une importance capitale et qu'il dépend de la taille du réseau. Cette dernière affirmation qui semblait évidente nous pousse vers une question capitale : comment déterminer le nombre de mesures optimal permettant d'obtenir les meilleurs résultats ?

Les figures 2.11(a) et 2.11(b) contiennent un tracé de l'erreur temporelle pour $\tau = 10$ mesures respectivement sur *net4* et *net5*. Nous pouvons alors nous rendre compte que GFO se comporte comme la tomogravité en terme d'estimation dans le temps : aucun instant n'est mieux estimé qu'un autre. La courbe est relativement plate ce qui nous permet d'affirmer que nous obtenons une solution qui est aussi bonne pour tous les instants de mesure. L'optimisation réalisée étant très proche d'une interpolation linéaire, ce résultat n'est pas étonnant mais mène à deux commentaires importants

- Les solutions proposées par les méthodes de troisième génération ont une erreur temporelle croissante au cours du temps entre deux calibrations successives de leur modèle. La solution que nous proposons permet une erreur constante au cours du temps pour la période d'étude considérée.
- Nous avons expliqué dans la partie 2.2 que nous voulons estimer les fanouts sur une courte période de temps sur laquelle ils sont censés peu varier. Cet objectif est atteint sur les exemples traités car l'erreur est presque constante au cours du temps.



(a) Erreur temporelle sur *net4*

(b) Erreur temporelle sur *net5*

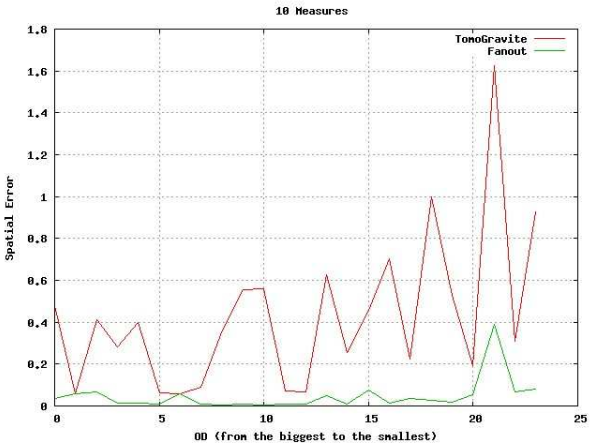
FIG. 2.11 – Erreur temporelle pour 10 mesures

Autres résultats pour $pVar = 1\%$

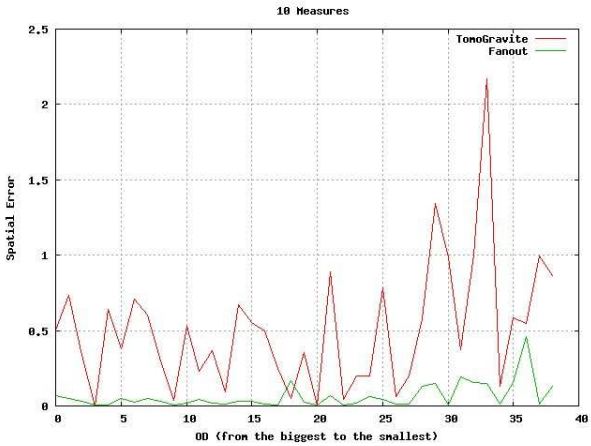
Nous proposons ici quelques résultats supplémentaires obtenus pour une valeur de $pVar$ égale à 1% ainsi qu'un ensemble de remarques généralisées à l'ensemble des tests effectués.

- Pour des réseaux de taille modérée tels que *net1*, *net2*, et *net3*, nous nous rendons compte que 10 mesures sont suffisantes pour obtenir des résultats tout à fait satisfaisants. Cela peut être constaté sur les figures 2.12(a), 2.12(b), 2.13(a) qui illustrent très bien ce point. L'erreur spatiale ne dépasse que très rarement 5% sauf pour des petits flots pour les deux premiers scénarios. Pour le troisième, l'erreur reste toujours inférieure à la limite tolérée par les opérateurs qui se situe autour des 20%.

- Pour des réseaux de taille plus conséquente comme *net6* qui contient 30 nœuds, l'erreur ne devient réellement acceptable qu'à partir de 30 mesures comme on peut le voir sur la figure 2.13(b). La limitation est bien visible si l'on regarde attentivement les résultats obtenus pour 10 ou 20 mesures qui sont illustrés sur les figures 2.14(a) et 2.14(b). La question de savoir si l'obtention de ces 30 mesures ne joue pas sur la qualité même de la solution peut alors être posée. Si les fanouts ont trop varié durant l'ensemble de ces mesures, l'estimation risque d'être faussée. Pour répondre à cette question, il est nécessaire de faire plus de tests sur des données réelles pour vérifier que ce

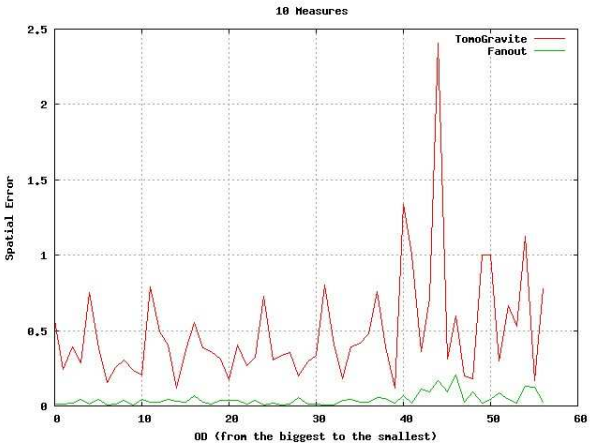


(a) Erreur spatiale sur *net1* avec 10 mesures

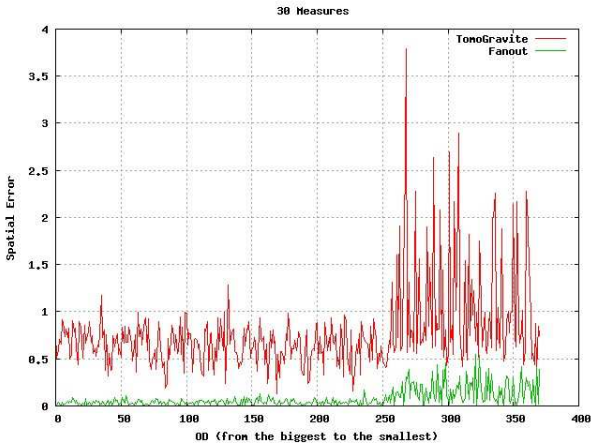


(b) Erreur spatiale sur *net2* avec 10 mesures

FIG. 2.12 – Erreur spatiale pour pVar = 1%



(a) Erreur spatiale sur *net3* avec 10 mesures



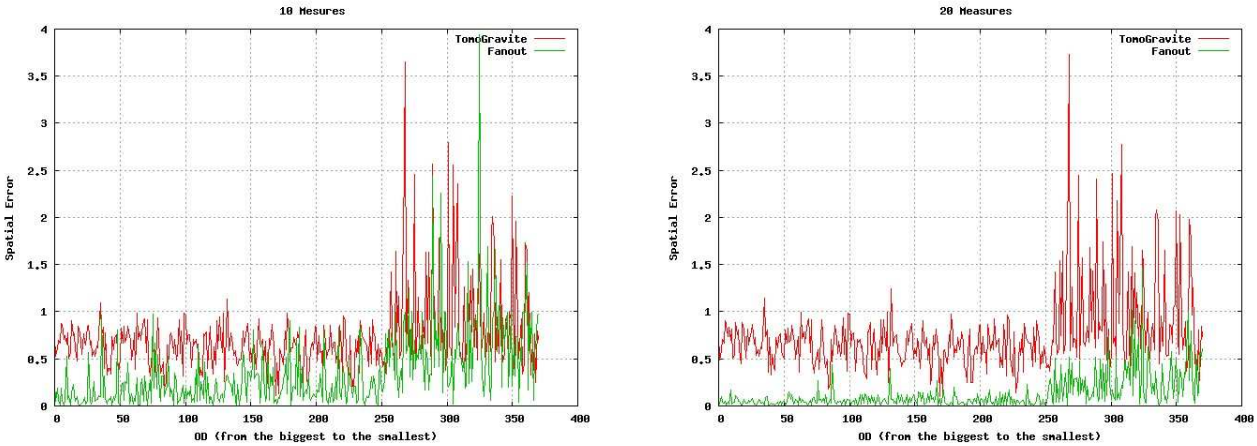
(b) Erreur spatiale sur *net6* avec 30 mesures

FIG. 2.13 – Erreur spatiale pour pVar = 1%

nombre de mesures est faisable.

Cependant, aucune contrainte ne nous force à utiliser des mesures successives, il est donc possible d'utiliser des mesures faites un autre jour par exemple en ajoutant l'hypothèse que les fanouts ont peu variés entre ces deux séries de mesures. Nous pouvons nous servir de la cyclo-stationnarité montrée par Papagiannaki et Al. par exemple.

► Les résultats de gravité que nous avons tracés montrent bien les limites d'une approche spatiale comme celles détaillées lors de notre état de l'art. Les matrices générées sont purement aléatoires quand à la valeur de leur demande et dans ce cas, les hypothèses de la tomogravité ne sont pas



(a) Erreur spatiale sur *net6* avec 10 mesures

(b) Erreur spatiale sur *net6* avec 20 mesures

FIG. 2.14 – Erreur spatiale trop élevée sur *net6* pour moins de mesures

vérifiées. Cela ne veut pas dire que les travaux réalisés par Zhang sont faux. Il existe certainement des exemples de réseaux régis par les lois de la gravité cependant, une telle hypothèse est trop limitative à nos yeux.

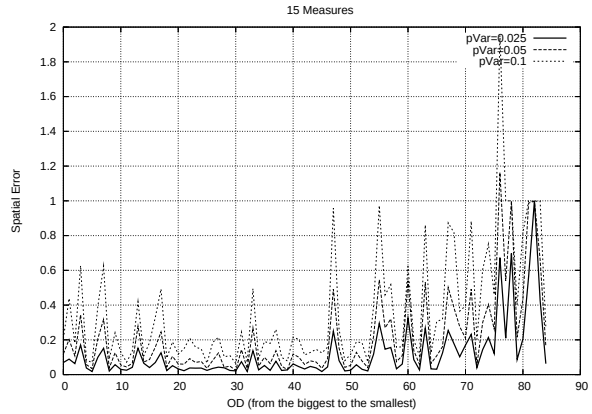
Résultats pour $pVar > 1\%$

Nous avons également testé notre algorithme pour des valeurs de $pVar$ supérieur à 1%. Comme nous travaillons sous l'hypothèse d'une variation faible des fanouts, nous avons limité la valeur de $pVar$ à 10%. Nous avons également augmenté le nombre de mesures à $\tau = 15$ pour obtenir un comportement assez surprenant sur lequel nous reviendrons ensuite.

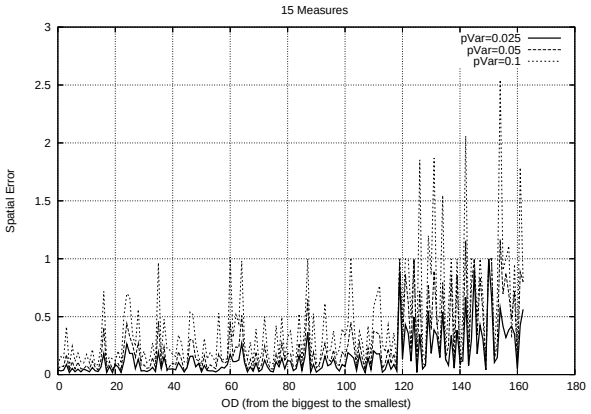
Les figures 2.15(a) et 2.15(b) montrent les résultats obtenus. Le premier commentaire que nous pouvons faire est que l'erreur spatiale augmente lorsque $pVar$ augmente. Cela parait logique mais il est nécessaire de voir dans quelle mesure notre algorithme reste efficace lorsque les variations sur les fanouts deviennent plus importantes. On remarque alors que jusqu'à $pVar = 5\%$, les résultats obtenus restent corrects. Au delà de cette valeur, les estimations sont vraiment mauvaises.

La figure 2.15(a) montre une particularité intéressante sur les résultats obtenus lorsque $pVar$ augmente. En effet, les résultats sur *net4* étaient relativement bons pour $\tau = 10$ mesures, cependant, en augmentant le nombre de mesures à 15, nous obtenons l'effet inverse de celui espéré : certains gros trafics sont mal estimés ce qui rend l'estimation des matrices inutilisable par les opérateurs. Cela doit être dû à la divergence apportée par une variation plus importante des fanouts au cours du temps. Plus on utilise de mesure plus cette erreur s'accentue ce qui justifie ces résultats.

En s'intéressant plus particulièrement aux figures 2.16(a) et 2.16(b), nous pouvons distinguer une particularité étonnante du comportement de notre algorithme face à des valeurs croissantes de $pVar$.



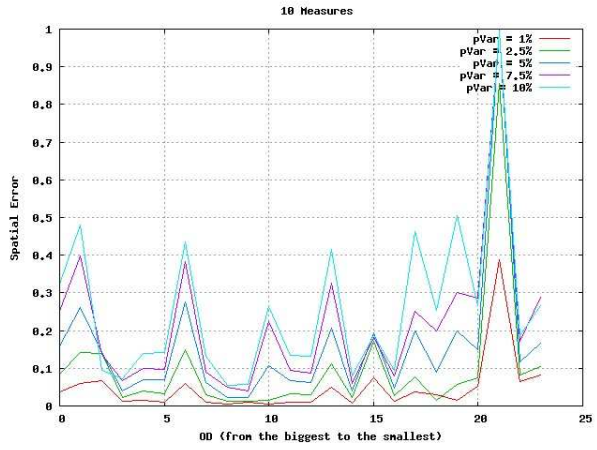
(a) Erreur Spatiale sur *net4* pour différentes valeurs de $pVar$



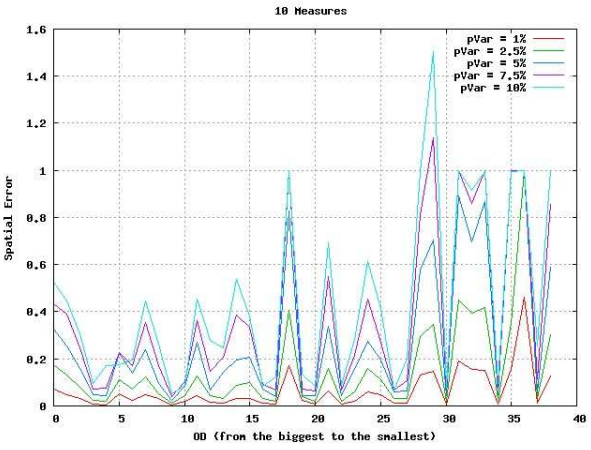
(b) Erreur Spatiale sur *net5* pour différentes valeurs de $pVar$

FIG. 2.15 – Résultats pour différentes valeurs de $pVar$

En effet, ces topologies ont été choisies volontairement de part leurs faibles nombres d'OD ce qui nous permet de mieux distinguer ce qu'il se passe de manière visuelle. On remarque en fait que certains couples OD sont mal estimés et que ce ne sont pas forcément les plus petits. Mais surtout, lorsque $pVar$ augmente, ces couples OD mal estimés le sont de plus en plus ce qui donne une série de courbe avec des pics de plus en plus hauts. Cela est étonnant car plus $pVar$ augmente plus notre estimation s'éloigne d'une valeur moyenne acceptable mais cela se fait toujours pour les mêmes couples OD. On s'attendrait à ce que les couples OD mal estimés puissent évoluer mais cela ne se produit pas. Ce dernier point est certainement un résultat de notre approche qui a tendance à favoriser certains couples OD par rapport à d'autres. Cela reste un point obscur à traiter en priorité.



(a) Erreur Spatiale sur *net1* pour différentes valeurs de $pVar$



(b) Erreur Spatiale sur *net2* pour différentes valeurs de $pVar$

FIG. 2.16 – Particularité des erreurs quand $pVar$ augmente

2.6 Conclusion

L'estimation des matrices de trafic est un sujet qui a été traité par de nombreux chercheurs mais qui reste tout de même un problème relativement jeune car les premiers travaux de Vardi datent de 1996. De nombreuses approches ont été proposées et nous proposons au début de ce chapitre un résumé rapide des techniques les plus connues.

Dans le cadre des nouvelles techniques d'estimation qui se basent sur des hypothèses spatiales et temporelles sur l'évolution des trafics, nous avons proposé d'utiliser une propriété encore inexploitée du comportement des trafics sous certaines hypothèses. Nous avons donc supposé que les rapports entre les trafics (les fanouts) sont quasi-stationnaires pour des durées variant de une à deux heures. Partant de cette hypothèse, nous avons proposé une nouvelle modélisation du problème ainsi qu'une heuristique permettant de le résoudre. Les résultats obtenus sont bons sous certaines réserves et hypothèses.

Une comparaison des résultats obtenus par notre algorithme avec ceux proposés par la tomogravité ([70]) a montré une amélioration nette de la qualité de la solution. Cependant, nous pensons que ces bons résultats obtenus ne sont possibles que par la nature même du réseau sur lesquels les tests ont été effectués. De la même manière, nous affirmons que notre approche du problème n'est pas toujours adaptée mais que son utilisation dans le cadre des réseaux d'opérateur devrait s'avérer efficace.

En effet, la propriété dont nous partons pour développer notre solution peut être résumée ainsi : les proportions de trafics entre deux parties communicantes varient peu. Et effectivement, dans le cadre des réseaux d'opérateur, de nombreux clients louent les services de l'opérateur pour établir leurs communications propres. La constance des proportions étudiées semble donc valide étant donné qu'un client ne communiquera jamais avec les autres clients de l'opérateur. De plus, cette approche est validée par des mesures et analyses effectuées par Papagiannaki dans [79].

En terme de perspectives, nous pouvons dégager deux axes importants dans lesquels il nous semble intéressant de poursuivre.

1. Tester et analyser les résultats de cette approche sur des données réelles. Pour cela, il faudrait pouvoir bénéficier des mesures faites par certains opérateurs sur leur réseau. Les mesures actuellement disponibles ne nous permettent pas de valider ces résultats étant donné qu'elles sont faites sur des réseaux de transit permettant l'interconnexion de nombreux système autonomes. Dans de tels réseaux, la tomogravité donne elle de très bons résultats.
2. Approfondir notre étude pour des valeurs plus élevées de $pVar$. Le comportement étonnant de nos indicateurs pour des valeurs de $pVar$ croissantes devrait pouvoir nous guider pour améliorer les solutions trouvées. Il est important à notre avis de vérifier que ce comportement est le même avec d'autres techniques de résolution ou d'optimisation.

CHAPITRE 3

Optimisation des métriques IP dans les réseaux IP/MPLS

Un concept clé de l'ingénierie de trafic est celui de l'optimisation du routage, qui permet une utilisation plus efficace des ressources réseaux existantes en adaptant le routage aux trafics supportés par le réseau. L'optimisation du routage permet une amélioration globale des performances, en particulier pour les réseaux ayant une distribution de trafic non uniforme. Il est cependant bien connu que, à cause du principe même du routage dans ces réseaux, l'ingénierie de trafic dans les réseaux IP est un problème complexe à résoudre.

Dans le chapitre 1, nous avons expliqué que le calcul des routes dans un réseau IP s'effectue grâce à des protocoles de routage. OSPF (Open Shortest Path First) et ISIS (Intermediate System to Intermediate System) sont actuellement les deux protocoles de routage intra-domaine les plus utilisés dans l'Internet. Ces protocoles routent un flot de trafic en utilisant le(s) plus court(s) chemin(s) entre la source et la destination, avec éventuellement un partage de charge équitable au niveau d'un nœud si plusieurs plus courts chemins existent.

Le poids des interfaces, et par conséquent les plus courts chemins utilisés par le routage, peuvent être modifiés par l'administrateur du réseau. Ces poids peuvent être mis à 1 pour minimiser le sauts (nombre de routeurs traversés) jusqu'à la destination, ou bien proportionnel à la distance physique pour minimiser le délai de propagation. Généralement l'objectif est de minimiser la congestion sur les interfaces réseau, l'heuristique standard recommandée par Cisco consiste à prendre ces poids inversement proportionnels à la capacité des interfaces de transmission. Il est cependant reconnu que de telles heuristiques conduisent souvent à une utilisation assez pauvre des ressources du réseau, et qu'elles s'accommodent mal de variations brusques de trafic ou de pannes d'équipements.

On a longtemps considéré que des protocoles comme OSPF ou ISIS n'étaient pas assez flexibles pour permettre une ingénierie de trafic efficace. C'est une des raisons de l'introduction des technologies

de commutation d'étiquettes comme MPLS (Multi-Protocol Label Switching) qui permettent de définir le chemin de chaque flot dans le réseau.

Récemment, la communauté scientifique, sous l'impulsion de Fortz et Thorup en particulier, s'est intéressée de plus en plus au problème d'optimisation des métriques de routage IP. On sait désormais que ce problème, connu pour être NP-difficile, est d'une importance pratique considérable puisqu'on a pu montrer qu'une affectation suffisamment intelligente des métriques de routage permet souvent d'obtenir un routage aussi performant que le routage MPLS sans coût financier supplémentaire, parfois même avec des performances proches de celles d'un routage optimal par partage de charge. Les relations étroites entre un IGP (« Interior Gateway Protocol ») et MPLS au niveau des chemins utilisés dans la propagation MPLS accentuent d'autant la nécessité de pouvoir contrôler le routage IP à l'aide d'une ingénierie efficace.

Ainsi, depuis quelques années, sont parus dans la littérature scientifique plusieurs algorithmes d'optimisation des métriques IP. Dans la suite de ce chapitre, nous donnerons rapidement quelques informations sur le contenu scientifique des travaux réalisés sur ce sujet. Puis nous présenterons en détail nos travaux et contributions. Partant des travaux de Fortz et Thorup, nous proposons un algorithme permettant :

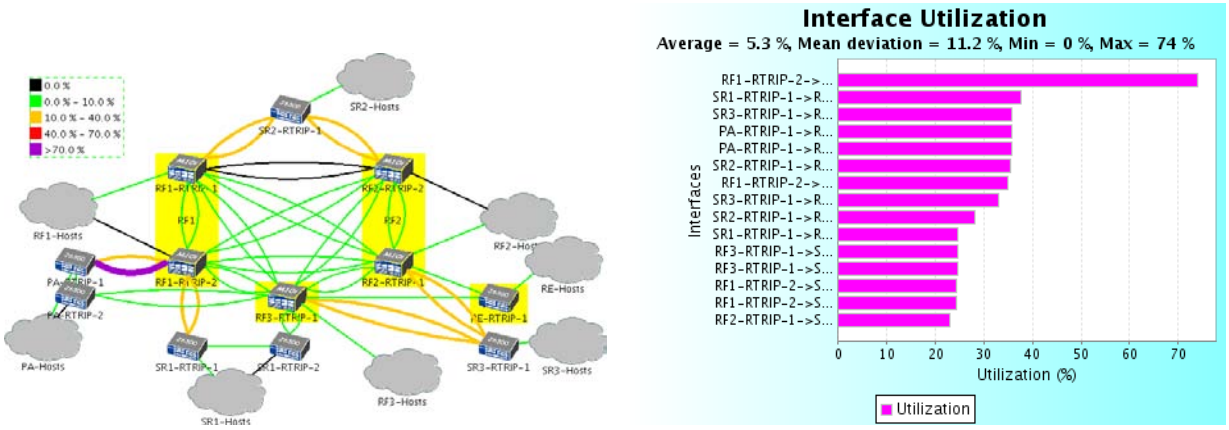
- d'améliorer les performances globales d'un réseau IP en réduisant la congestion des interfaces et en minimisant les délais et les taux de pertes de bout-en-bout des flots,
- de dégager plus de bande-passante résiduelle sur les chemins empruntés par les flots pour :
  - augmenter la capacité du réseau à tolérer sans dégradation forte du service des variations brusques du trafic sur une courte période de temps, et
  - réduire la fréquence des modifications du plan de routage dû à des augmentations prévisibles de trafic,
- et d'augmenter la robustesse du réseau en prenant en compte la défaillance des équipements dans le processus d'optimisation.

Les contributions apportées portent sur un algorithme plus efficace et sur une vision incrémentale des modifications de métriques proposées par cet algorithme. Lors de nos travaux, nous avons en permanence gardé à l'esprit le fait que notre algorithme puisse être utilisé par des opérateurs. Pour cela, nous sommes parti du postula que changer un plan de routage complet n'était pas réalisable sur un réseau opérationnel. Une approche incrémentale résultant en une suite de modifications améliorant l'état du réseau a donc été développée.

3.1 Etat de l'art

Un des piliers de l'ingénierie de trafics est l'utilisation optimisée des ressources disponibles. Ainsi, bien répartir les volumes des trafics que le réseau doit écouler est une thématique de recherche importante sur laquelle la communauté scientifique s'est penchée.

Comme nous l'avons expliqué dans le chapitre 1, les réseaux IP ont maintenant besoin d'être planifiés, et cela passe par un choix judicieux des routes utilisées. Malgré la mise en place de techniques telles que IntServ ou DiffServ, aucune QoS ne peut être garantie si les ressources sont mal utilisées. La figure 3.1 illustre cette nécessité, en effet sur cette figure, les ressources du réseau sont largement suffisantes étant donné que l'utilisation moyenne des interfaces est très faible (moins de 6%). Avec une interface chargée à plus de 70%, la seconde plus chargée étant à moins de 40%, le routage proposé (métriques égales à 1) engendre une mauvaise utilisation de ces ressources et donc une QoS très pauvre quels que soient les mécanismes de gestion de la QoS qui sont mis en place.



(a) Visualisation graphique de l'utilisation des interfaces

(b) Statistique de l'utilisation des interfaces

FIG. 3.1 – Mauvaise utilisation des ressources

La répartition optimale des flots sur le réseau est un problème bien connu sous le nom de « Multi Commodity Flow Problem » . Partant d'un ensemble de flots et d'un ensemble de routes possibles pour chacun d'entre eux, l'idée est de trouver le (ou les) chemin(s) sur lequel (lesquels) placer chacun des flots. Pour ce faire, il est nécessaire de proposer une évaluation des solutions traitées et le plus souvent, le délai moyen de bout en bout est le critère retenu. L'ensemble de ces problèmes est connu pour être NP difficile ([36]) aussi des techniques basées principalement sur les méthodes de descentes admissibles ou sur des heuristiques permettent de les résoudre (voire [50]). Une extension du problème où le nombre de chemins est contraint est proposé par Duhamel et Mahey [86]. Les lecteurs intéressés pourront de référer aux travaux de Pioro et Al. dans [40] pour plus de détails sur l'ensemble des travaux réalisés sur ce sujet.

Néanmoins, mettre en place un tel routage n'est pas chose aisée. Il n'y a aucune garantie que le routage trouvé précédemment puisse être mis en place à l'aide des protocoles de routage IP qui utilisent les plus courts chemins. Ce second problème est lui connu sous le nom de « Problème Inverse des Plus Courts Chemins » . Didier Burton l'a longuement étudié lors de sa thèse [29]. Une autre formulation du problème existe dans laquelle les auteurs cherchent à maximiser le nombre de chemins précédemment calculés qui sont effectivement mis en place par les métriques choisies (voir les travaux de Pioro et Al. dans [64] et ceux de Ameur et Al. dans [60]).

Pour contourner l'ensemble de ces problèmes, une approche différente peut être choisie : supposons que l'on ait déjà un plan de routage défini par un ensemble de métriques donné. Quelles sont alors les

modifications de métrique permettant d'optimiser un critère de performance ? Ce critère de performance peut porter sur la QoS des flots et/ou sur l'utilisation des ressources du réseau. De plus, l'optimisation étant faite sur le fonctionnement nominal du réseau, qu'en est-il en régime de panne ? Il faut garantir des performances « acceptables » dans ce cas de figure également.

On peut trouver une définition de ce problème appelé « Flow Allocation Problem » (FAP) dans les travaux de Bertsekas et Galager [13] et [15]. Bertsekas cherche l'ensemble des métriques IP permettant de satisfaire des contraintes d'utilisation sur les interfaces inférieures à 100%. Ces travaux aboutissent sur un problème linéaire qu'il prouve être NP difficile. De nombreux auteurs ont alors proposé des heuristiques permettant de résoudre ce problème (recuit simulé, ajustement de métrique). Des techniques basées sur la relaxation lagrangienne sont également proposées par Pioro dans [64]. L'ensemble de ces travaux ont des résultats limités, la raison principale étant la difficulté et les temps de calculs nécessaires pour relier les changements de métriques aux évolutions de charges sur les interfaces.

Plus tard, Fortz et Thorup ont relancé la recherche sur ce sujet qui fut quelque peu délaissée de part l'adhésion mondiale à MPLS. Ils publient alors deux articles [51] en 2000 et [62] 2 ans plus tard proposant une méta-heuristique basée sur des notions de recherche locale menant à une solution de bonne qualité et ce dans des temps de calculs relativement courts comparés aux travaux plus anciens. Pour cela, ils proposent une structure de voisinage, s'inspirant fortement des particularités du routage, où un voisin est obtenu à partir de la solution courante

- par un changement aléatoire sur une et une seule métrique,
- et par introduction de partage de charge.

Fortz et Thorup créent ainsi un voisinage composé d'un côté de voisins aléatoires permettant de sortir des minimas locaux et de voisins ajoutant du partage de charge de l'autre. Comme ce voisinage est relativement important, l'exploration est aléatoire et porte sur 20% de ce voisinage. Les auteurs mettent également en place une recherche dite tabou permettant une sortie plus efficace des minimums locaux.

Les deux points clés associés à l'évaluation des voisins sont le routage et la propagation. Pour éviter des temps de calculs inacceptables, Fortz et Thorup proposent d'utiliser des techniques de propagation dynamique et de reroutage ciblé. L'idée étant que quelques changements de métriques ne vont pas faire changer tout le routage et en conséquence toutes les charges sur le réseau. Les deux auteurs obtiennent alors des résultats convaincants montrant que dans certains cas, le routage OSPF optimisé peut être très proche de la solution de répartition optimale des flots.

Partant de leurs travaux, nous proposons une modification de la structure du voisinage et du fonctionnement de l'algorithme menant à deux contributions qui nous semblent intéressantes : une diminution des temps de calcul de part un voisinage plus petit, et une solution exploitable par les opérateurs de part son aspect incrémental.

3.2 Modélisation du problème

Nous proposons dans cette partie un ensemble de notations et la modélisation nous permettant ainsi de poser de manière claire le problème que nous cherchons à résoudre. Nous donnons également une définition du coût du réseau.

A quelques différence près que nous avons volontairement introduites pour conserver les notations précédemment utilisées dans le chapitre 2, la modélisation proposée est identique à celle utilisée par Fortz et Thorup dans leurs travaux ([51] et [62]). Cette modélisation permet principalement une définition aisée de la propagation dynamique des flots et de la mise à jour ciblée des routes suite à des modifications de métriques.

3.2.1 Modélisation de la topologie

Le réseau sera modélisé sous la forme d'un graphe orienté dont les nœuds correspondent aux routeurs et les arcs aux interfaces de communication des routeurs. On note N le nombre de nœuds du graphe et L le nombre d'arcs du graphe. A chaque arc a du graphe est associé un poids w_a qui représente la métrique de l'interface correspondante et qui peut prendre toute valeur dans l'intervalle $\Omega = \left[1, 2^{16} - 1\right]$. Étant donné une solution admissible $\mathbf{w} = (w_1, \ldots, w_L)$, on lui associe

- $D_i^x(\mathbf{w})$, la distance du nœud i au nœud destination, x,
- $\delta_{i,j}^x(\mathbf{w})$, qui vaut 1 si l'arc (i, j) est sur un plus court chemin vers la destination x et 0 sinon,
- et $n_i^x(\mathbf{w}) = \sum_j \delta_{i,j}^x(\mathbf{w})$, qui représente le nombre d'arcs sortants du nœud i qui sont sur un plus court chemin vers le nœud destination x.

Ces notations permettent de représenter les plus courts chemins (PCC) dans un graphe. Elles sont illustrées sur la figure 3.2 dans laquelle la destination x est 3, et où les métriques sont indiquées sur les arcs du graphe.

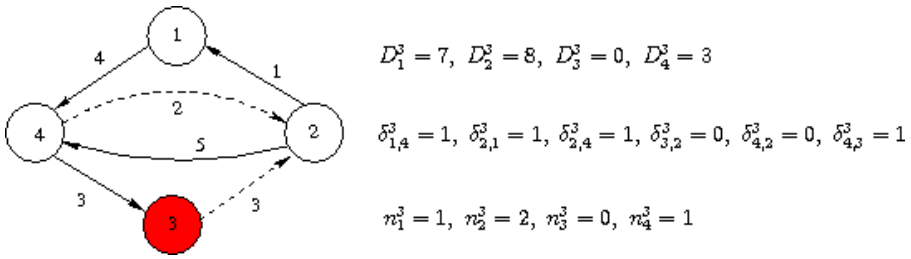


FIG. 3.2 – Représentation des plus courts chemins dans un graphe

3.2.2 Modélisation du trafic

Le réseau écoule un ensemble de K flots de communication. Chaque flot $k = 1, \ldots, K$ est caractérisé par sa source $s(k)$, sa destination $t(k)$ et sa demande en bande-passante d^k. Chaque flot est considéré comme étant un couple Origine/Destination unique. On a donc $K \leq N^2$.

Remarque 3.1 *Dans la pratique les réseaux IP écoulent généralement un nombre de flots très supérieur à N^2 si on prend en compte toutes les caractéristiques d'une communication : protocoles applicatifs et de transport, ports sources et destinations, type de service, codec, paramètres de QoS, etc. Sans perte de généralité, nous considérons ici l'agrégation de tous les flots ayant mêmes origine et destination en un flot unique. L'origine et la destination sont en effet les seules informations impactant le routage. Les prises en compte des mécanismes DiffServ et du routage par ToS (Type Of Service) peuvent être vues comme des extensions de l'algorithme que nous proposons.*

On notera

λ_i^x le trafic direct émis au nœud i pour la destination x :

$$\lambda_i^x = \sum_{\substack{k, \\ s(k)=i, \\ t(k)=x}} d^k$$

$\gamma_i^x(\mathbf{w})$ le trafic reçu au nœud i, direct et en transit, pour la destination x. En supposant un partage de charge équitable ($\frac{1}{N}$ sur chaque chemin s'il en existe N), on a alors :

$$\gamma_i^x(\mathbf{w}) = \lambda_i^x + \sum_{j \neq x} \frac{\delta_{j,i}^x(\mathbf{w})}{n_j^x(\mathbf{w})} \gamma_j^x(\mathbf{w})$$

La charge de l'interface (i, j) s'écrit alors :

$$Y_{i,j}(\mathbf{w}) = \sum_{x=1}^{N} \frac{\delta_{i,j}^x(\mathbf{w})}{n_i^x(\mathbf{w})} \gamma_i^x(\mathbf{w})$$

Ainsi, étant donnée un vecteur de métriques $\mathbf{w} = (w_1, \ldots, w_L)$, nous pouvons calculer récursivement à partir des formules précédentes le trafic $\gamma_i^x(\mathbf{w})$ reçu au nœud i pour la destination x et la charge $Y_{i,j}(\mathbf{w})$ sur chaque interface (i, j) à partir des variables $\delta_{i,j}^x(\mathbf{w})$ et $n_i^x(\mathbf{w})$ contenant l'information sur les plus courts chemins pour la destination x.

3.2.3 Fonctions coûts additives

Soit un vecteur de métriques $\mathbf{w} = (w_1, \ldots, w_L)$. Pour pouvoir formuler un problème d'optimisation, nous devons définir le coût $\Phi(\mathbf{w})$ de cette solution. Classiquement, un coût de forme additive peut être utilisé :

$$\Phi(\mathbf{w}) = \sum_{l=1}^{L} \Phi_l(\mathbf{w})$$

où $\Phi_l(\mathbf{w})$ représente la contribution de l'interface l au coût global. Nous proposons dans la suite plusieurs critères d'optimisation pour cette forme de coût.

Optimisation de la QoS

Nous proposons ci-dessous deux formulations du coût des interfaces, toutes les deux basées sur des formules issues de la théorie des files d'attente.

Coût basé sur la formule M/M/1/∞

De par la formule de Little, on sait qu'il est équivalent de minimiser le délai moyen de bout en bout des paquets et le nombre moyen de paquets dans le réseau.

Considérons une interface (i,j) de capacité $C_{i,j}$. Pour minimiser le nombre moyen de paquets sur cette interface, nous utilisons la formule $M/M/1/\infty$ avec une pénalité permettant d'évaluer le coût pour des solutions non-admissibles au sens des capacités (i.e. $Y_{i,j} \geq C_{i,j}$). En notant,

$$\overline{Y}_{i,j} = K_1\, C_{i,j}$$

nous pouvons définir le coût associé à l'interface (i,j) :

$$\Phi_{i,j}(Y_{i,j}) = \begin{cases} \frac{Y_{i,j}}{C_{i,j}-Y_{i,j}} & \text{si } Y_{i,j} \leq \overline{Y}_{i,j} \\ \frac{\overline{Y}_{i,j}}{C_{i,j}-\overline{Y}_{i,j}} + K_2 \frac{C_{i,j}}{\left[C_{i,j}-\overline{Y}_{i,j}\right]^2} \left[Y_{i,j} - \overline{Y}_{i,j}\right] & \text{si } Y_{i,j} > \overline{Y}_{i,j} \end{cases} \tag{3.1}$$

où $K_1 \in]0,1[$ et $K_2 > 1$ sont deux paramètres ajustables. On remarquera que la pénalité appliquée dans le cas $Y_{i,j} > \overline{Y}_{i,j}$ correspond à la dérivée du nombre moyen de paquets au seuil de charge $\overline{Y}_{i,j}$ multipliée par le coefficient K_2 (cf. figure 3.3).

Coût basé sur la formule M/M/1/K

Un coût basé sur la formule M/M/1/K ([8]) présente l'avantage de permettre d'optimiser à la fois le délai moyen des paquets mais également les taux de perte. Cette formule étant valide quel que soit le taux d'utilisation de l'interface, nous n'avons pas à faire intervenir une pénalité comme dans le coût précédent.

En notant $K_{i,j}$ la capacité en nombre de paquets du buffer de l'interface (i,j), nous proposons donc de définir le coût associé à cette interface par :

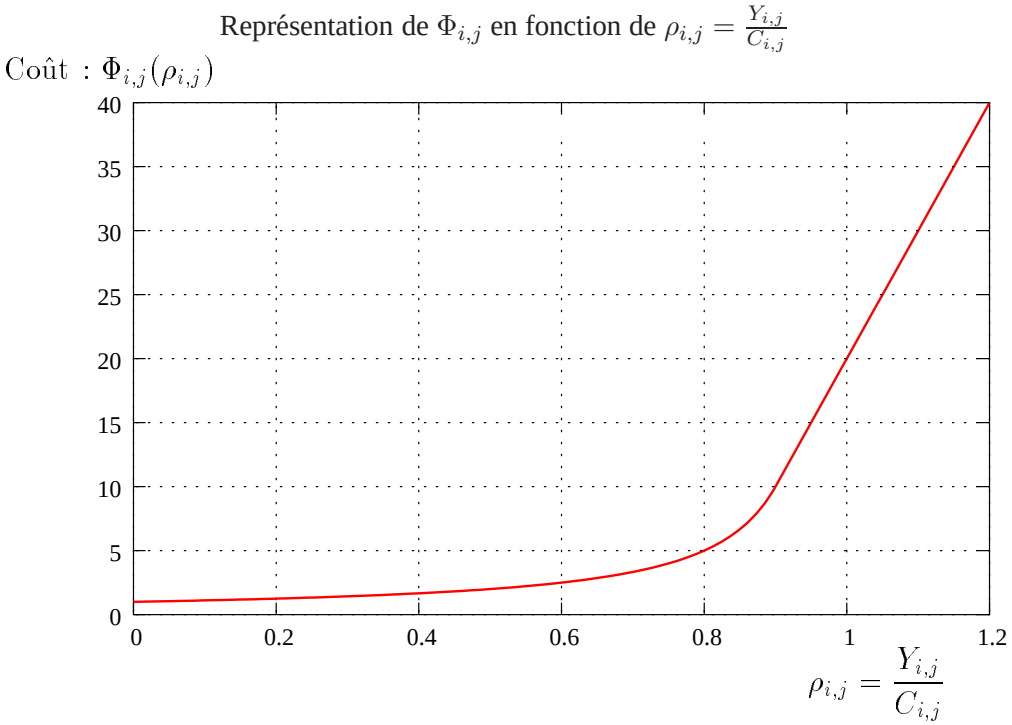


FIG. 3.3 – Coût basé sur la formule M/M/1 avec $K_1 = 0.9$, $K_2 = 1$ et $C_{i,j} = 1$.

$$\Phi_{i,j}(Y_{i,j}) = \frac{C_{i,j}}{K_{i,j}} W_{i,j} \; + \; P_{i,j} \tag{3.2}$$

où, en notant $\rho_{i,j} = Y_{i,j}/C_{i,j}$:

$$W_{i,j} = \frac{1}{C_{i,j} - Y_{i,j}} \quad \text{et} \quad P_{i,j} = 1 \; - \; \frac{1 - {\rho_{i,j}}^{K_{i,j}}}{1 - {\rho_{i,j}}^{K_{i,j}+1}}$$

Optimisation de la capacité résiduelle totale

La capacité résiduelle d'un réseau est un critère important, qui permet notamment de mesurer sa capacité à tolérer des variations brusques du trafic. Une capacité résiduelle importante permet également d'assurer que le réseau pourra supporter l'évolution de la demande en trafic sur une période de temps assez longue. Pour optimiser ce critère, nous proposons le coût suivant :

$$\Phi_{i,j}(Y_{i,j}) = \begin{cases} K_1 \, Y_{i,j}^2 & \text{si } Y_{i,j} \leq C_{i,j} \\ K_1 \, Y_{i,j}^2 \; + \; K_2 \, [C_{i,j} - Y_{i,j}]^2 & \text{si } Y_{i,j} > C_{i,j} \end{cases} \tag{3.3}$$

On remarquera que la pénalité appliquée dans le cas $Y_{i,j} > C_{i,j}$ permet de « forcer » la méthode d'optimisation à converger vers une solution admissible au sens des capacités.

Optimisation de l'utilisation maximale des interfaces

Nous pouvons également formuler notre coût de manière non additive. Ainsi, nous proposons par exemple de considérer l'utilisation maximale des interfaces sur le réseau :

$$\Phi(\mathbf{w}) = \max_{(i,j)} \ \Phi_{i,j}(\mathbf{w}) = \max_{(i,j)} \ \frac{Y_{i,j}(\mathbf{w})}{C_{i,j}} \tag{3.4}$$

De part les tests effectués sur l'ensemble des critères proposés, nous nous sommes rendus compte que l'utilisation maximale des interfaces est le critère le plus intéressant. Les coûts de forme additive permettent un traitement plus facile dans le cadre de l'optimisation, en effet, ils sont séparables et souvent dérivables ce qui permet d'appliquer plusieurs techniques d'optimisation. Néanmoins, l'approche retenue pour traiter notre problème étant une heuristique basée sur des techniques de recherche locale, la formulation de coût sous forme de maximum ne nous gênera pas. En plus, minimiser l'utilisation maximale des interfaces permet de minimiser les sources de congestions. Dans la suite de ce chapitre, nous parlerons donc le plus souvent de l'utilisation maximale des interfaces lorsque nous évoquerons le critère utilisé. Cependant, ce critère peut toujours être remplacé sans changer de manière fondamentale la résolution du problème proposé.

3.2.4 Formulation mathématique du problème

Le problème à résoudre est le suivant :

$$\underset{\mathbf{w}\in\Omega^L}{\text{Minimiser}} \left\{ \Phi(\mathbf{w}) = \max_{(i,j)} \ \frac{Y_{i,j}(\mathbf{w})}{C_{i,j}} \right\}$$

avec :

$$\begin{aligned} Y_{i,j}(\mathbf{w}) &= \sum_{x=1}^{N} \frac{\delta_{i,j}^x(\mathbf{w})}{n_i^x(\mathbf{w})} \gamma_i^x(\mathbf{w}) \quad \forall i,j = 1 \dots N \\ \gamma_i^x(\mathbf{w}) &= \lambda_i^x + \sum_{j \neq x} \frac{\delta_{j,i}^x(\mathbf{w})}{n_j^x(\mathbf{w})} \gamma_j^x(\mathbf{w}) \quad \forall i,x = 1 \dots N \end{aligned} \tag{3.5}$$

Les paramètres $\delta_{i,j}^x(\mathbf{w})$ et $n_i^x(\mathbf{w})$ étant obtenus directement à partir du vecteur de métriques $\mathbf{w}$ par résolution d'un problème des plus courts chemins.

3.3 Résolution du problème

Pour résoudre le problème précédent, nous proposons d'utiliser un algorithme basé sur des techniques de recherche locale dont l'originalité principale réside dans sa structure de voisinage. Le voisi-

nage $V(\mathbf{w})$ d'une solution $\mathbf{w} = (w_1, \ldots, w_L)$ est défini par :

$$V(\mathbf{w}) = \left\{\mathbf{w}^1, \mathbf{w}^2, \ldots, \mathbf{w}^L\right\} \tag{3.6}$$

où :

$$\mathbf{w}^i = (w_1, w_2, \ldots, w_i + \Delta_i, \ldots, w_L) \tag{3.7}$$

avec :

$$\Delta_i = argmin_{\Delta \geq 1} \left[Y_i\left(w_1, \ldots, w_i + \Delta, \ldots, w_L\right) < Y_i(\mathbf{w})\right] \tag{3.8}$$

Ainsi le voisinage d'une solution $\mathbf{w}$ contient exactement L solutions (équation 3.6). Chacune de ces solutions est associée à un interface i (équation 3.7). Elle est obtenue en augmentant la métrique de cette interface de la quantité minimale permettant de réduire la charge de cette interface, c'est à dire de dévier du trafic de cette interface (équation 3.8). On remarquera que cette structure de voisinage permet également de générer automatiquement les situations de partage de charge puisque si une seule interface sortante est utilisée, la modification minimale de métrique permettant de dévier du trafic est celle qui introduit du partage de charge.

3.3.1 Génération du voisinage

Pour une solution $\mathbf{w} = (w_1, \ldots, w_L)$, la génération de son voisinage consiste, pour chaque interface i de poids w_i, à déterminer la variation minimale Δ_i de sa métrique qui permettra de dévier du flot de cette interface. Notons F_i l'ensemble des flots passant par l'interface i. On peut distinguer deux cas :

1. $F_i = \emptyset$: dans ce cas, Δ_i ne peut être définie puisqu'on ne peut pas dévier du flot de l'interface i.

2. $F_i \neq \emptyset$: si l'on veut dévier au moins un des flots de i, il faut donc que l'interface i ne fasse plus partie des plus courts chemins (PCC) d'au moins un des flots de F_i. On procède de la façon suivante :

   (a) On supprime l'interface i du graphe : $\mathbf{w}' = (w_1, \ldots, w_{i-1}, \infty, w_{i+1}, \ldots, w_L)$.
   (b) Mise à jour des tables de distances $D_u^v(\mathbf{w}')$ par un algorithme des plus courts chemins.
   (c) On calcule :

   $$d_{min} = \min_{f \in F} \left[D_{s(f)}^{t(f)}(\mathbf{w}') - D_{s(f)}^{t(f)}(\mathbf{w})\right]$$

   qui correspond à la variation minimale de distance entre la source $s(f)$ et la destination $t(f)$ de chaque flot f passant par l'interface i.

   (d) On définit Δ_i de la façon suivante :

$$\Delta_i = \begin{cases} 1 & \text{si } d_{min} = 0 \\ d_{min} & \text{si } 0 < d_{min} < \infty \\ \infty & \text{si } d_{min} = \infty \end{cases}$$

L'ensemble de flots F_i est obtenu directement lors de la propagation des flots, nécessaire pour calculer le coût d'une solution. La suppression de l'interface i est réalisée en lui affectant une métrique infinie. La mise à jour des tables de distances, suite à cette modification, est obtenue par un algorithme de plus courts chemins dynamique que nous décrivons plus tard et qui est basé sur les travaux de Ramalingam et Reps [36].

Dans la solution $\mathbf{w}^i = (w_1, w_2, \ldots, w_i + \Delta_i, \ldots, w_L)$, il existera au moins un flot de F_i qui sera dévié tout ou partie de l'interface i :

- Si $d_{min} = 0$, cela signifie qu'il n'y avait pas unicité du plus court chemin pour au moins un des flots de F_i (partage de charge). En posant $\Delta_i = 1$, ce flot sera intégralement dévié de l'interface i,
- Si $0 < d_{min} < \infty$, la solution $\mathbf{w}^i$ introduit du partage de charge pour au moins un des flots de F_i,
- Si $d_{min} = \infty$, il n'existe pas d'autre chemin ne passant pas par l'interface i pour les flots de F_i. On supprime donc cette interface de celles dont la métrique peut être modifiée.

3.3.2 Algorithme d'optimisation des métriques

L'algorithme de recherche locale mis en œuvre est l'algorithme 3. Cet algorithme n'assure pas forcément une décroissance monotone du coût pour pouvoir « sortir » des optima locaux. La technique mise en place n'est pas du type tabou mais simplement une exploration limitée de solutions plus coûteuses.

Le test de convergence permettant de terminer l'algorithme est basé sur 3 critères :

1. Nombre maximum d'itérations (Q_1),
2. Nombre maximum d'augmentations successives du coût (Q_2),
3. Voisinage de la solution courante vide : $\Delta_i = \infty, \forall i = 1 \ldots L$.

L'initialisation de l'algorithme consiste à calculer les PCC, et les paramètres $D_u^v(\mathbf{w})$, $\delta_{u,v}^x(\mathbf{w})$, $n_u^v(\mathbf{w})$ associés, pour la solution initiale $\mathbf{w}$ grâce à l'algorithme de Dijkstra. On propage ensuite les flots sur ces PCC pour obtenir le trafic reçu en chaque nœud u pour chaque destination v, $\gamma_u^v(\mathbf{w})$, et donc le trafic sur chaque interface (u, v), $Y_{u,v}(\mathbf{w})$. On en déduit immédiatement le coût $\Phi(\mathbf{w})$ de la solution initiale.

La boucle principale de l'algorithme effectue une exploration du voisinage de la solution courante $\mathbf{w}$. Pour chaque interface i, on calcule Δ_i en posant $w_i = \infty$ et en effectuant un calcul de PCC. Si $\Delta_i \neq \infty$, on évalue la solution voisine $\mathbf{w}^i$ en déterminant l'impact sur les PCC de la modification $w_i \rightarrow w_i + \Delta_i$ sur la métrique de l'interface i, puis en propageant les flots sur ces PCC. On détermine ainsi la solution voisine $\mathbf{w}^i$ de coût minimal.

De manière pratique, étant donné qu'il existe souvent des métriques identiques ou des routes de distances identiques, la valeur de Δ_i que l'on trouve est souvent égale à 1. Nous avons donc décidé d'explorer tout d'abord le voisin correspondant à l'interface i en appliquant $\Delta_i = 1$. Si ce changement n'a aucun impact sur la répartition des charges, alors nous cherchons la valeur de Δ_i définie plus haut.

algorithm 3 Optimisation des Métriques

procedure METRICOPTIMISATION
  Initialisation
  Lecture de la solution initiale $\mathbf{w} = (w_1, \ldots, w_L)$
  Calcul des plus courts chemins $\rightarrow$ $D_u^v(\mathbf{w})$, $\delta_{u,v}^x(\mathbf{w})$, $n_u^v(\mathbf{w})$ $\quad u, v, x = 1 \ldots N$
  Propagation des flots sur les plus courts chemins $\rightarrow$ $\gamma_u^v(\mathbf{w})$, $Y_{u,v}(\mathbf{w})$ $\quad u, v = 1 \ldots N$
  Calcul du coût $\Phi(\mathbf{w})$ de la solution initiale.
  $\mathbf{w}^* = \mathbf{w}$ ▷ Initialisation de la solution de coût minimum

  Boucle Principale
  while Convergence() = false **do**
    $\Phi_{min} = \infty$
    for $i = 1 \ldots L$ **do**
      Calcul de $\Delta_i \quad \rightarrow \quad \mathbf{w}^i = (w_1, w_2, \ldots, w_i + \Delta_i, \ldots, w_L)$
      Calcul des plus courts chemins $\rightarrow$ $D_u^v(\mathbf{w}^i)$, $\delta_{u,v}^x(\mathbf{w}^i)$, $n_u^v(\mathbf{w}^i)$ $\quad u, v, x = 1 \ldots N$
      Propagation des flots $\rightarrow$ $\gamma_u^v(\mathbf{w}^i)$, $Y_{u,v}(\mathbf{w}^i)$ $\quad u, v = 1 \ldots N$
      Calcul du coût $\Phi(\mathbf{w}^i)$
      if $\Phi(\mathbf{w}^i) \leq \Phi_{min}$ **then**
        $\mathbf{w}_{next} = \mathbf{w}^i$ et $\Phi_{min} = \Phi(\mathbf{w}^i)$
      end if
    end for
    $\mathbf{w} = \mathbf{w}_{next}$
    if $\Phi(\mathbf{w}) < \Phi(\mathbf{w}^*)$ **then**
      $\mathbf{w}^* = \mathbf{w}$ ▷ Copie de la solution de coût minimum
    end if
  end while
end procedure

Cette modification a accéléré de manière significative l'algorithme (division des temps de calcul par 3 en moyenne) en évitant le calcul de la variation minimale un bon nombre de fois.

3.3.3 Approche incrémentale ou globale

L'algorithme 3 propose donc un ensemble de modifications itératives à appliquer sur la solution initiale pour améliorer le critère utilisé. Nous proposons alors de l'utiliser de deux manières bien distinctes :

1. **Algorithme incrémental basé sur la topologie existante**

   Pour cela, nous utilisons l'algorithme incrémental d'optimisation des métriques proposé précédemment en utilisant les métriques de la topologie traitée comme solution initiale. Dès lors, la solution proposée est directement applicable par un opérateur souhaitant améliorer le critère optimisé et ce à moindre coût car le nombre de changement de métrique est restreint aux seules variations proposées. Cet algorithme sera noté A_{INCR} et nous utiliserons les paramètres $Q_1 = 200$

et $Q_2 = 5$ (ces valeurs proviennent de nombreux tests).

Toutes les modifications proposées par l'algorithme améliorent itérativement et de manière stricte le coût du réseau. Ces modifications contiendront donc éventuellement plusieurs changements de métriques étant données les augmentations de coûts tolérées par l'algorithme dans certains cas bien précis.

D'un point de vue pratique, l'opérateur peut donc choisir le nombre de modifications qu'il souhaite appliquer sur son réseau. Quel que soit ce nombre, à partir du moment où l'ordre proposé est respecté, le coût du réseau sera diminué.

2. Algorithmes « multi-start »

Nous proposons également deux algorithmes dont la solution proposée est souvent de meilleure qualité que celle de l'algorithme précédent. Cependant, la solution proposée est plus difficile à mettre en place pour un opérateur et les temps de calculs sont plus longs que pour la solution précédente. Les deux algorithmes « multi-start » proposés sont

- $A_{\mathrm{MS}}^{\mathrm{stop}}$ qui effectue l'algorithme incrémental sur trois solutions initiales différentes : la solution courante, une solution où toutes les métriques sont égales à 1, et une solution où les métriques sont inversement proportionnelles à leur capacité. Cet algorithme sera configuré avec $Q_1 = 100$ et $Q_2 = 0$, et
- $A_{\mathrm{MS}}^{\mathrm{min}}$ qui est le même que $A_{\mathrm{MS}}^{\mathrm{stop}}$ mais avec les paramètres $Q_1 = 100$ et $Q_2 = 5$. C'est à dire que nous autorisons une recherche permettant de sortir des minimums locaux.

Les métriques solutions proposées par ces deux algorithmes peuvent être toutes différentes de celles configurées sur le réseau initial. Dans ce cas, il est peu probable qu'un opérateur accepte d'appliquer l'ensemble de ces changements de peur de déstabiliser son réseau. De plus, l'aspect incrémental des améliorations proposées est perdu, il faut donc appliquer la totalité des changements pour améliorer le réseau. Ces deux solutions ne seront donc jamais utilisées sur un réseau fonctionnel, cependant, leur intérêt principal est associé à la configuration d'un réseau encore inexistant.

Pour chacun de ces algorithmes, le calcul du coût d'une solution est immédiat quand on connaît la charge des interfaces. Par contre, ils font fréquemment appel à deux opérations plus complexes décrites dans les parties suivantes :

– Le calcul de l'impact sur les PCC de l'augmentation de métrique d'une interface, et
– la propagation des flots sur les plus courts chemins.

3.3.4 Algorithme des plus courts chemins dynamiques

La modification de la métrique d'une seule interface n'impacte souvent qu'un petit nombre de PCC. Des algorithmes, dits de plus courts chemins dynamiques, permettent de traiter ce type de problèmes plus efficacement que les algorithmes de Dijkstra ou de Bellman-Ford. Nous avons utilisé une amélioration de l'algorithme de Ramalingam et Reps [36], décrite en détail dans [69]. La structure simple

de notre voisinage permet de se retrouver en permanence face à l'augmentation d'une et une seule métrique. C'est ce qui nous permet de simplifier d'autant plus l'algorithme des plus courts chemins dynamiques. Ce gain de temps non négligeable est une des forces de l'algorithme développé.

On suppose que le calcul complet de toutes les tables de distance a été fait. On se place ici dans le cas où la métrique d'une interface (s,t) a augmenté d'une valeur Δ. On note $\mathbf{w}$ le vecteur de métriques initial et $\mathbf{w}'$ celui obtenu après modification. Au début de l'algorithme, on a :

$$D_u^v(\mathbf{w}') = D_u^v(\mathbf{w}),\ \delta_{u,v}^x(\mathbf{w}') = \delta_{u,v}^x(\mathbf{w}),\ n_u^v(\mathbf{w}') = n_u^v(\mathbf{w}) \quad u,v,x = 1\dots N$$

Pour illustrer les différentes étapes de l'algorithme, nous utiliserons le réseau de la figure 3.4(a). La destination pour laquelle on veut recalculer les PCC est le nœud F. Sur cette figure sont indiquées, pour chaque interface (u,v), sa métrique w_{uv} et son utilisation vers F, $\delta_{u,v}^F(\mathbf{w})$, ainsi que, pour chaque nœud u, sa distance à F : $D_u^F(\mathbf{w})$. On suppose que l'interface modifiée est $(s,t) = (D,G)$ et que sa métrique passe de 1 à 3 ($\Delta = 2$).

Première étape : propagation amont du changement.

Cette première étape permet de remonter le long des chemins utilisés pour joindre F et d'identifier les nœuds et les interfaces impactées par la modification $w_{s,t} \rightarrow w_{s,t} + \Delta$.

A1. On vérifie que ce changement fera évoluer des distances vers F. Pour cela, il faut que : $\delta_{s,t}^F(\mathbf{w}) = 1$ et $n_s^F(\mathbf{w}) = 1$. Sinon, on fait $\delta_{s,t}^F(\mathbf{w}') = 0$ et l'algorithme est terminé. En effet :
- si $\delta_{s,t}^F(\mathbf{w}) = 0$ alors le fait d'augmenter la métrique entre s et t n'aura aucune conséquence car cette interface n'est pas sur un plus court chemin vers F, et
- si $n_s^F(\mathbf{w}) > 1$ alors il existe au moins un autre plus court chemin de s vers F ne passant pas par t donc seul l'utilisation de l'interface de s vers t est modifiée par le changement de métrique.

A2. On initialise une liste Q avec le nœud source de l'interface modifiée : $Q = \{s\}$.

A3. Pour chaque nœud $v \in Q$ et pour chaque interface (u,v) utilisée, i.e. $\delta_{u,v}^F(\mathbf{w}) = 1$, on fait $D_u^F(\mathbf{w}') = D_u^F(\mathbf{w}) + \Delta$. Si $n_u^F(\mathbf{w}) = 1$ alors, u est ajouté dans Q et on fixe $\delta_{u,v}^F(\mathbf{w}') = 0$ (interface non utilisée).

Sur l'exemple, on a bien une évolution des distances vers F puisque $\delta_{D,G}^F(\mathbf{w}) = 1$ et $n_D^F(\mathbf{w}) = 1$. Au départ, $Q = \{D\}$. En remontant les PCC utilisés, on obtient $Q = \{D, C, A\}$ comme illustré sur la figure 3.4(b).

Deuxième étape : mise à jour des distances

Si $\Delta = 1$, vu que $n_s^F(\mathbf{w}) = 1$ (cf. A1), on a $\delta_{s,t}^F(\mathbf{w}') = 1$ même après changement et les distances $D_u^F(\mathbf{w}')$, $u \in Q$, calculées à l'étape 1 sont exactes. En effet, les distances étant entières, le second (ou les seconds) plus court chemin entre s et F est plus long au minimum d'une unité par rapport au plus court passant par t. Dès lors, le fait d'augmenter de 1 la métrique de l'interface entre s et t crée au mieux un partage de charge entre le chemin passant par t et le(s) second(s) plus court(s) chemin(s).

Donc, si $\Delta = 1$, la route entre s et F passant par t reste un plus court chemin entre s et F. Sinon, il est possible que l'interface (s,t) ne soit plus utilisée, auquel cas ces distances peuvent avoir été surévaluées. Si $\Delta > 1$, on réalise les opérations suivantes pour les ajuster :

B1. On traite le routeur source s. On va regarder s'il existe un chemin au départ de s tel que sa distance soit plus petite que $D_s^F(\mathbf{w}) + \Delta$. On calcule :

$$d = \min_{(s,v)} \left[{w'}_{s,v} \; + \; D_v^F(\mathbf{w}) \right]$$

Si $d < D_s^F(\mathbf{w}')$, on pose $\Delta_1 = d - D_s^F(\mathbf{w}')$. Δ_1 représente l'erreur sur l'augmentation de distance qui a été propagée à tous les nœuds de Q. On met à jour la distance de s à F : $D_s^F(\mathbf{w}') = d$.

B2. Pour tout nœud $u \in Q$, on corrige les distances en faisant : $D_u^F(\mathbf{w}') = D_u^F(\mathbf{w}') - \Delta_1$. On calcule alors :

$$d = \min_{(u,v)} \left[{w'}_{u,v} \; + \; D_v^F(\mathbf{w}) \right]$$

Si $d < D_u^F(\mathbf{w}')$, on corrige la distance de u à F en faisant $D_u^F(\mathbf{w}') = d$, puis on insère u dans un tableau associatif H en l'associant à la distance d.

B3. Cette dernière étape permet de réajuster les distances faussées du fait de l'ordre dans lequel les routeurs ont été traités à l'étape B2. Si le traitement précédent était fait dans l'ordre des distances à la destination, cette sous-partie serait inutile.
Tant que $H \neq \emptyset$, on retire u de H où $u = \underset{u \in H}{\text{argmin}}(D_u^F(\mathbf{w}'))$.

Pour toutes les interfaces (v,u) entrantes dans u, on compare : $d_1 = D_v^F(\mathbf{w}')$ à $d_2 = D_u^F(\mathbf{w}') + {w'}_{v,u}$. Si v vérifie $d_1 > d_2$ alors, on fixe $D_v^F(\mathbf{w}') = d_2$ et on ajoute le couple $(v, D_v^F(\mathbf{w}'))$ dans H.

Pour notre exemple, le résultat de cette partie est illustré sur la figure 3.4(c). On obtient $\Delta_1 = 0$ à l'étape B1. Puis on insère $(C,4)$ puis $(A,5)$ à l'étape B2.

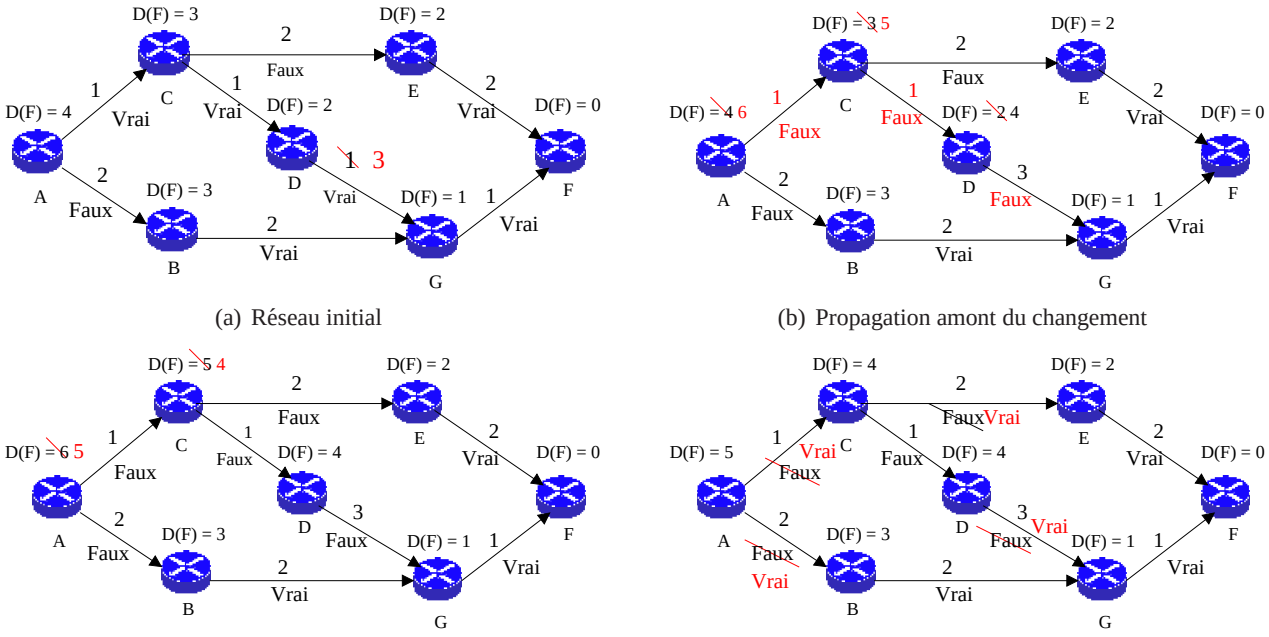
Troisième étape : mise en place des routes à l'aide des nouvelles distances.

Cette étape permet de corriger les flags d'utilisation des interfaces $\delta_{u,v}^F(\mathbf{w}')$, ainsi que le nombre de PCC vers F pour chaque nœud u, $n_u^F(\mathbf{w}')$.

Pour chaque nœud $u \in Q$, on va tester les distances sur les différentes interfaces sortantes. Pour chaque interface (u,v) on compare : $D_u^F(\mathbf{w}')$ à $D_v^F(\mathbf{w}') + {w'}_{u,v}$. Si l'égalité est vérifiée, on fait :

$$\delta_{u,v}^F(\mathbf{w}') = 1 \quad \text{et} \quad n_u^F(\mathbf{w}') = n_u^F(\mathbf{w}') + 1$$

Pour notre exemple, on peut remarquer que nous n'avons modifié que quelques nœuds : D, C et A. Le résultat final est sur la figure 3.4(d). L'algorithme de Dijkstra demanderait un coût calculatoire bien plus élevé pour obtenir le même résultat.



(a) Réseau initial

(b) Propagation amont du changement

(c) Modification si $\Delta > 1$

(d) Résultat final

FIG. 3.4 – Déroulement de l'algorithme de Ramalingam et Reps

3.3.5 La propagation dynamique

La dernière étape coûteuse qui doit être optimisée est la propagation des trafics qui est nécessaire pour pouvoir calculer le coût des solutions explorées. Nous utilisons la même technique que Fortz et Thorup dans [51] connue sous le nom de propagation dynamique des trafics.

Cette technique de propagation permet de propager les trafics une fois les PCC connus. On considère ici la propagation vers un nœud destination x. Il faudra donc itérer cette technique pour toutes les destinations pour calculer la charge $Y_{i,j}(\mathbf{w})$ des interfaces. La relation de base utilisée, qui exprime la conservation des flots, est la suivante :

$$Y^x_{i,j}(\mathbf{w}) = \frac{\lambda^x_i + \sum_{k \neq x} Y^x_{k,i}(\mathbf{w})}{n^x_i} \tag{3.9}$$

Nous distinguons deux cas : le cas statique et le cas dynamique. Le cas statique propage tous les trafics. Le cas dynamique est utilisé suite à un changement de métrique sur le réseau. Il s'appuie sur les charges des interfaces calculées lors du cas statique et ne propage que des variations de trafic pour les nœuds impactés par la modification de métrique.

Le cas statique

Pour chaque nœud i, du plus éloigné au plus proche de la destination x traitée, on calcule le trafic $\gamma_i^x(\mathbf{w})$ reçu pour la destination x : somme du trafic direct $\lambda_i^x(\mathbf{w})$ et des trafics $Y_{k,i}^x(\mathbf{w})$ reçu sur chaque interface entrante (k, i). Ce trafic est ensuite réparti sur les interfaces sortantes en tenant compte du partage de charge. L'algorithme détaillé est l'algorithme 4.

algorithm 4 Propagation Statique

```
1:  procedure StaticPropagation(x)
2:      Q = {1, ..., N} − {x}                          ▷ Q contient tous les nœuds sauf x
3:      while Q ≠ ∅ do
4:          i = argmax_{u∈Q} ( D_u^x(w) )              ▷ i est le nœud le plus éloigné de x
5:          γ_i^x = λ_i^x                              ▷ Trafic direct en i vers x
6:          for k tel que δ_{k,i}^x(w) = 1 do
7:              γ_i^x(w) = γ_i^x(w) + Y_{k,i}^x(w)     ▷ Ajout du trafic des interfaces entrantes
8:          end for
9:          for j tel que δ_{i,j}^x(w) = 1 do
10:             Y_{i,j}^x(w) = γ_i^x(w)/ n_i^x         ▷ Propagation sur les interfaces sortantes
11:         end for
12:         Q = Q − {i}
13:     end while
14: end procedure
```

Le cas dynamique

Suite à une modification de métrique, nous supposons avoir exécuté la mise à jour des PCC à l'aide de l'algorithme des plus courts chemins dynamique. Lors de cette exécution, nous construisons une liste M contenant les nœuds affectés par le changement de métrique. C'est à dire que M contient l'ensemble des nœuds dont le routage vers x a été modifié lors de la mise à jour. Plus précisément, nous dirons qu'un nœud est affecté s'il est source ou destination d'une interface dont l'utilisation a été modifiée suite au changement de métrique. L'algorithme détaillé est alors l'algorithme 5. Il est quasiment identique au premier sauf que le nombre d'actions est minimisé. On ne s'occupe que des nœuds impactés pour la destination x, et de ceux qui le deviennent en aval des premiers dans la direction de x.

3.3.6 Intégration de la résilience

Il est important d'intégrer la notion de panne dans tout algorithme cherchant à améliorer l'utilisation des ressources sur un réseau. Sans cette prise en compte, aucune solution ne peut réellement être évaluée car elle peut s'avérer être bien pire que la situation nominale en cas de panne.

Prendre en compte toute panne éventuelle dans le cadre de l'optimisation de métriques risquerait de rendre l'algorithme inefficace si cela n'est pas fait de manière réfléchie. Dans leur travaux [51] et [62], Fortz et Thorup propose d'intégrer dans l'évaluation des voisins les pires pannes repérées au préalable. Pour se faire, un ensemble de T pannes considérées comme les pires sont prises en compte au moment

algorithm 5 Propagation Dynamique

```
1: procedure DynamicPropagation(x)
2:     while M ≠ ∅ do
3:         i = argmax_{u∈M} ( D_u^x(w) )        ▷ i est le nœud le plus éloigné de x
4:         M = M − {i}
5:         Y = λ_i^x(w)                          ▷ Ajout du trafic direct en i vers x
6:         for k tel que δ_{k,i}^x(w) = 1 do
7:             Y = Y + Y_{k,i}^x(w)              ▷ Ajout du trafic des interfaces entrantes
8:         end for
9:         Y = Y/n_i^x                           ▷ Partage de charge
10:        for j tel que δ_{i,j}^x = 1 do
11:            if Y_{i,j}^x ≠ Y then
12:                Y_{i,j}^x = Y                 ▷ Propagation sur les interfaces sortantes
13:                M = M ∪ {j}                   ▷ Le trafic vers j a changé
14:            end if
15:        end for
16:    end while
17: end procedure
```

de l'évaluation des voisins. Cet ensemble étant mis à jour régulièrement (toutes les 20 itérations par exemple).

Nous proposons tout d'abord de caractériser les occurrences des pannes possibles et de les relier aux différents états du réseau. Nous caractériserons ensuite l'ensemble des pannes qu'un réseau peut supporter, puis nous intégrerons l'ensemble au problème d'optimisation des métriques.

3.3.7 Probabilités d'états associés aux réseaux IP

Chaque panne possible sur le réseau n'a pas la même probabilité d'occurrence. Il est possible d'associer à chaque équipement e du réseau une probabilité q_e que celui-ci soit en panne. En supposant qu'il existe en tout P équipements, nous pouvons donc considérer un ensemble de variables aléatoires indépendantes $Y_e,\ e = 1..P$ valant 1 si l'équipement est en panne (probabilité q_e) et 0 si celui-ci est actif (probabilité $1 - q_e$). Dès lors, il devient intéressant d'étudier la probabilité qu'aucun élément ne soit en panne, en posant $X = \sum_e Y_e$:

$$p[X = 0] = \prod_{e=1}^{P} (1 - q_e) \tag{3.10}$$

De la même manière, la probabilité qu'un et un seul élément soit en panne peut être calculée :

$$\begin{aligned} p[X=1] &= p[Y_1 = 1 \text{ et } Y_i = 0 \text{ pour } i \neq 1] \\ &+ p[Y_2 = 1 \text{ et } Y_i = 0 \text{ pour } i \neq 2] \\ &+ \dots \\ &+ p[Y_P = 1 \text{ et } Y_i = 0 \text{ pour } i \neq P] \\ p[X=1] &= \sum_{e=1}^{P} p[Y_e = 1 \text{ et } Y_i = 0 \text{ pour } i \neq e] \\ &= \sum_{e=1}^{P} q_e \prod_{i \neq e} (1 - q_i) \end{aligned} \tag{3.11}$$

Nous posons alors l'hypothèse qu'un seul équipement peut être en panne à la fois. Cette hypothèse sera évaluée et discutée un peu plus loin dans cette partie. Dès lors, nous obtenons exactement $P + 1$ états possibles pour le réseau. L'état nominal E_0 qui correspond au réseau où aucun équipement n'est en panne et chaque état du réseau $E_e,\ e \neq 0$ qui correspond à un fonctionnement nominal de chacun des équipements sauf pour l'un d'eux : l'équipement e. De manière évidente, si nous notons p_e la probabilité d'être dans l'état E_e alors :

$$\begin{aligned} p_0 &= \frac{1}{Q} \prod_{i=1}^{P} (1 - q_i) \\ p_e &= \frac{q_e}{Q} \prod_{\substack{i=1 \\ i \neq e}}^{P} (1 - q_i) \end{aligned} \tag{3.12}$$

où Q est une constante de normalisation représentant la somme des probabilités non normalisées sur tous les états considérés, de manière plus formelle en se servant des équations (3.10) et (3.11) :

$$\begin{aligned} Q &= P[X=0] + P[X=1] \\ &= \prod_{i=1}^{P} (1 - q_i) + \sum_{e=1}^{P} q_e \prod_{\substack{i=1 \\ i \neq e}}^{P} (1 - q_i) \end{aligned} \tag{3.13}$$

Pour valider l'hypothèse concernant l'unicité des pannes dans les états considérés, il est nécessaire de chiffrer $\overline{p} = P[X > 1] = 1 - Q$. Nous pouvons trouver dans [75] des valeurs mesurées quand aux probabilités de panne des différents équipements sur le réseau. Se référant à ce livre, nous prendrons donc $q_e = 10^{-3}\%$ si e est un routeur et $q_e = 0.18\%$ si e est une interface. Dès lors, les résultats obtenus et visibles dans le tableau 3.1 montrent que cette hypothèse, souvent utilisée, est en fait assez restrictive. En effet, avec 200 routeurs et 600 interfaces, il y a 30% du temps au moins 2 pannes (avant dernière ligne du tableau 3.1). Cependant, de part l'aspect hiérarchique du routage et du protocole OSPF en particulier, l'optimisation des métriques se fera généralement sur des réseaux ne dépassant pas une centaine de nœuds.

#Routeurs	**#Interfaces**	**P**	p_0	$\sum_{e=0}^{P} p_e(\%)$	$\overline{p}$ (%)
10	50	60	91.4	99.6	0.4
50	250	300	63.7	92.4	7.6
100	400	500	48.6	83.7	16.3
200	600	800	33.9	70.6	29.4
1000	3000	4000	0.4	**2.9**	**97.1**

TAB. 3.1 – Test sur les probabilités d'avoir plusieurs pannes simultanées

3.3.8 Caractéristiques des éléments pouvant être en panne

Nous avons jusqu'alors parlé d'éléments en panne sans jamais préciser le type de ces éléments. Cisco donne des valeurs de disponibilité pour les routeurs et pour les interfaces IP, mais il faut bien noter que la disponibilité d'une interface tient compte de nombreuses causes possibles : panne d'une carte sur laquelle elle est branchée, panne des équipements de couche inférieure. Nous considérerons donc que les valeurs proposées par Cisco intègrent l'ensemble de ces causes.

D'autres types d'éléments peuvent être considérés parmi lesquels

Les cartes (PIC) : la panne d'une carte entraîne la chute de l'ensemble des interfaces qui sont branchées dessus. Le nombre de ports disponibles sur une carte peut varier jusqu'à des valeurs supérieures à la dizaine. La chute d'une carte peut donc avoir des conséquences extrêmement graves. Cette notion d'interfaces associées les unes aux autres par des risques de pannes peut être généralisée.

Les SRLG : « Shared Risk Link Groups » ou groupe de liens à risques partagés, ils représentent un ensemble de liens liés les uns aux autres de telle manière qu'ils peuvent tous être en panne en même temps. Cette liaison peut être due à des équipements de niveau inférieur, au branchement sur une même carte etc. . .. Dans tous les cas, considérer ces éléments peut être d'une importance cruciale.

Les sites : composés d'un ensemble de routeurs, ils représentent par exemple un immeuble ou une zone géographique peu étendue. Si un incident tel qu'une panne d'électricité, une panne de climatisation, ou un incendie se produit, c'est l'ensemble des routeurs composant ce site qui tombent en panne.

Dans l'étude proposée ici, seules les pannes de nœuds et de lien IP (deux interfaces de sens opposé) seront considérées. Cependant, la structure même de l'algorithme développé permet une prise en compte aisée de l'ensemble de ces éléments.

3.3.9 Minimisation de l'espérance du critère sur l'ensemble des états du réseau

Nous proposons alors de changer le calcul du critère considéré lors de l'optimisation des métriques. L'intégration des pannes peut être faite de telle manière à éviter les changements de métriques entraînant

des surcoûts trop importants en cas de panne. Mais il nous semble bien plus intéressant de proposer une optimisation d'un critère tenant compte des pannes directement. Ainsi, le critère utilisé sera l'espérance du coût associé à chaque état possible. En notant $\Phi^e(\mathbf{w}),\ e = 0..P$ la valeur du critère calculé sur le réseau dans l'état e pour un ensemble de métriques $\mathbf{w}$, nous pouvons donc écrire le critère global Φ par :

$$\Phi = \sum_{e=0}^{P} p_e \Phi^e(\mathbf{w}) \tag{3.14}$$

Où l'évaluation du coût $\Phi^e(\mathbf{w})$ se fait de la même manière que dans la position du problème de base (3.5) mais en considérant le réseau dans son état E_e. Nous pouvons ainsi donner une formulation du problème d'optimisation des métriques tenant compte de la résilience :

$$\underset{\mathbf{w}\in\Omega^L}{\text{Minimiser}} \sum_{e=0}^{P} p_e \Phi^e(\mathbf{w})$$

avec $(\lambda)^e$, $(\gamma)^e$, $(\delta)^e$, et $(n)^e$ calculés à partir du réseau dans l'état E_e et :

$$\begin{aligned}
\Phi^e(\mathbf{w}) &= \underset{(i,j)}{Max}\ \frac{Y^e_{i,j}(\mathbf{w})}{C_{i,j}} \\
Y^e_{i,j}(\mathbf{w}) &= \sum_{x=1}^{N} \frac{(\delta^x_{i,j}(\mathbf{w}))^e}{(n^x_i(\mathbf{w}))^e} (\gamma^x_i(\mathbf{w}))^e \quad \forall i,j = 1\ldots N \\
(\gamma^x_i(\mathbf{w}))^e &= (\lambda^x_i)^e + \sum_{j\neq x} \frac{(\delta^x_{j,i}(\mathbf{w}))^e}{(n^x_j(\mathbf{w}))^e} (\gamma^x_j(\mathbf{w}))^e \quad \forall i,x = 1\ldots N
\end{aligned} \tag{3.15}$$

3.3.10 Minimisation du critère en situation nominale sous contraintes de résilience

La formulation précédente cherche donc le vecteur de métriques permettant d'obtenir un comportement moyen optimal au cours du temps tenant compte des différents états possibles du réseau. Il peut cependant s'avérer tout aussi intéressant de chercher le vecteur de métriques menant à une solution nominale optimale sous contrainte que les conséquences des pannes ne soient pas critiques.

Ainsi, si le critère étudié est l'utilisation maximale des interfaces, cela revient à minimiser l'utilisation maximale de l'état nominal tout en se restreignant aux solutions telles que l'utilisation maximale en cas de panne ne dépasse pas une borne U_{max} donnée. Ce nouveau problème se pose aisément à partir de la définition du problème donné par (3.5) à laquelle il faut ajouter la contrainte que nous venons de préciser.

Nous sommes donc en présence de trois formulations bien distinctes du problème d'optimisation des métriques

1. minimisation d'un critère de performance,
2. minimisation d'un critère de performance sous contraintes d'absence de panne critique, et

3. minimisation de l'espérance du critère de performance sur tous les états possibles du réseau.

L'heuristique décrite plus tôt était associée à la résolution du premier problème, cependant son extension pour traiter les deux autres problèmes se fait alors aisément c'est pourquoi nous ne donnerons pas plus de détail quant à la résolution de ces nouveaux problèmes.

3.4 Tests et résultats

Cette partie est dédiée à l'étude de l'efficacité et de la pertinence de nos approches et des algorithmes associés. Nous proposons donc de débuter cette partie avec un réseau simple sur lequel nous effectuons une minimisation de l'utilisation maximale des interfaces sans tenir compte des pannes. Puis, nous donnerons des résultats sur des topologies plus complexes pour les différents problèmes posés.

3.4.1 Minimisation de l'utilisation maximale des interfaces sans contrainte de résilience

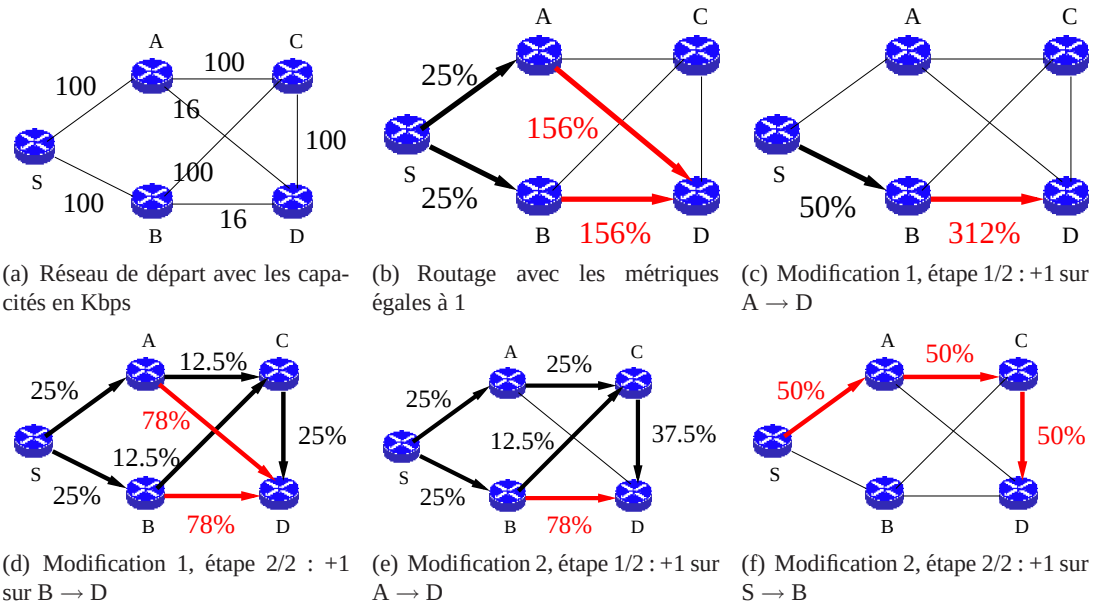
Le premier exemple que nous avons choisi a pour but d'illustrer l'intérêt et l'efficacité des modifications de métriques proposées par l'algorithme. Ainsi, la figure 3.5(a) illustre la topologie de départ où toutes les métriques sont unitaires. Les valeurs des capacités des interfaces sont données en Kbps sur la figure et un seul trafic, de demande 50 Kbps, est émis entre S et D.

Dès lors, un premier routage nous donne deux routes de S vers D (voir figure 3.5(b)). Le partage de flot équitable répartit donc 25 Kbps sur chaque route. Mais ces routes empruntent les interfaces A/D et B/D qui ne peuvent donc pas supporter tout le trafic (leur capacité est de 16 Kbps). La valeur initiale du critère vaut donc 156 %. Nous allons proposer une suite de modifications de métriques permettant de changer cette situation de surcharge.

L'ensemble des figures 3.5(c), 3.5(d), 3.5(e), et 3.5(f) montrent les modifications proposées par l'algorithme ainsi que les utilisations de chacune des interfaces chargées (en %). L'utilisation maximale étant écrite en rouge pour plus de lisibilité. Nous proposons de détailler les propositions de l'algorithme une à une

Première modification (Figures 3.5(c) et 3.5(d)) : cette modification contient en fait deux changements de métriques. En effet, l'algorithme itératif que nous proposons ne donne que des solutions améliorant le critère. Ainsi, la situation initiale correspond justement à un minimum local. Sortir de ce minimum revient à changer les deux métriques proposées c'est pourquoi elles devront être appliquées ensemble pour réellement améliorer la solution initiale. De manière pratique, les deux modifications proposées permettent un partage global sur l'ensemble des chemins possibles diminuant l'utilisation de l'interface entre B et D et entre A et D qui étaient trop chargées.

Seconde modification (Figures 3.5(e) et 3.5(f)) : les deux changements composant cette modification ont pour but de décharger complètement les deux interfaces de capacités plus faibles. Ainsi, le premier changement permet de dévier la totalité du flot de l'interface entre A et D et le second permet la déviation du flot de l'interface entre B et D. Ces deux changements sont proposés dans une même modification étant donné que le coût ne diminue pas strictement suite au



(a) Réseau de départ avec les capacités en Kbps

(b) Routage avec les métriques égales à 1

(c) Modification 1, étape 1/2 : +1 sur A → D

(d) Modification 1, étape 2/2 : +1 sur B → D

(e) Modification 2, étape 1/2 : +1 sur A → D

(f) Modification 2, étape 2/2 : +1 sur S → B

FIG. 3.5 – Déroulement de l'optimisation

premier changement.

L'utilisateur se voit donc conseiller quatre changements de métriques sous forme de deux modifications à effectuer. Pour chacune d'entre elles, l'utilisation maximale des interfaces diminue passant de 156% à 78% tout d'abord puis de 78% à 50% ensuite. L'opérateur peut donc décider d'appliquer le nombre de modifications qu'il souhaite avec une garantie d'amélioration quel que soit ce nombre à partir du moment où il respecte l'ordre dans lequel elles sont proposées. Cette approche itérative semble donc parfaitement adaptée à un réseau opérationnel où changer une métrique entraîne une instabilité temporaire du réseau de part les mises à jour du protocole.

3.4.2 Etude globale des algorithmes proposés pour optimiser l'état nominal sans contrainte de résilience

Pour justifier les performances de l'algorithme proposé, nous donnons ici les résultats obtenus sur un ensemble de 104 tests réalisés sur 13 topologies distinctes chacune associée à 8 matrices de trafics générées aléatoirement. Pour ces tests, le but de notre optimisation était de minimiser le délai moyen de bout en bout des paquets (sa formulation est donnée par l'équation (3.1)).

Le coût de la solution optimale par partage de flot est calculé pour chaque test par un algorithme basé sur des techniques de direction de descente admissible de type « Frank and Wolfe » ([27] et chapitre 2). Ce coût est de manière évidente une borne inférieure pour tous algorithmes d'optimisation des métriques IP. Nous avons également calculé les coûts des solutions proposées pour les algorithmes suivants :

– Les deux heuristiques standard c'est à dire les métriques égales à 1 (UNIT) et les métriques inversement proportionnelle à leur capacité (INVCAPA),
– l'algorithme de Fortz et Thorup (FTA) comme ils l'ont décrit dans [51], mais que nous avons limité à 200 itérations totales et à 20 itérations consécutives sans évolution de coût pour des raisons de temps d'exécution, et
– notre algorithme incrémental d'optimisation des métriques A_{INCR}, et le même algorithme exécuté en multi-start $A^{\min}_{\mathrm{MS}}$ et $A^{\mathrm{stop}}_{\mathrm{MS}}$.

Les solutions initiales utilisées pour FTA et A_{INCR} sont celles où toutes les métriques valent 1. Pour chacun des tests, nous avons calculé l'écart relatif en pourcent entre la borne inférieur donnée par la solution par partage de charge optimal et le coût obtenu par l'algorithme testé. Le tableau 3.2 présente la moyenne (AVG) sur les 8 matrices de trafics testées et l'écart type (STD) de ces écarts relatifs. Le tableau 3.2 donne également le nombre de nœuds et d'interfaces pour chaque topologie.

Topologies (#Routeurs, #Interfaces)		UNIT (%)	INVCAPA (%)	A_{INCR} (%)	$A^{\mathrm{stop}}_{\mathrm{MS}}$ (%)	$A^{\min}_{\mathrm{MS}}$ (%)	FTA (%)
Topologie 1	AVG	33.57	31.16	14.40	**7.75**	**6.39**	17.26
(13, 48)	STD	13.68	21.01	9.33	5.20	3.81	8.41
Topologie 2	AVG	186.96	3.74	4.45	**0.83**	**0.82**	4.30
(30, 122)	STD	47.82	1.46	2.29	0.53	0.54	1.25
Topologie 3	AVG	27.93	2.58	1.71	**1.05**	**1.04**	2.26
(50, 148)	STD	3.29	0.66	0.47	0.37	0.37	1.16
Topologie 4	AVG	20383.58	**0**	131.38	**0**	**0**	1800.96
(8, 22)	STD	4827.45	0	38.76	0	0	2759.31
Topologie 5	AVG	11.23	1.82	1.97	**1.57**	**1.57**	4.44
(7, 16)	STD	3.26	1.11	1.24	1.30	1.30	4.49
Topologie 6	AVG	6346.75	0.94	940.04	**0.38**	**0.38**	1782.46
(15, 42)	STD	992.50	1.35	2016.26	0.57	0.57	2596.26
Topologie 7	AVG	31.57	1.03	3.60	**0.24**	**0.24**	4.28
(12, 46)	STD	9.41	0.80	2.04	0.15	0.15	2.97
Topologie 8	AVG	74.42	1.07	4.35	**0.37**	**0.37**	4.05
(10, 28)	STD	41.00	0.41	4.08	0.30	0.30	3.11
Topologie 9	AVG	9.56	2.07	3.06	1.30	**1.25**	3.67
(12, 38)	STD	1.46	1.22	1.10	1.31	1.23	1.32
Topologie 10	AVG	14.74	0.33	**0.11**	**0.09**	**0.09**	0.29
(21, 46)	STD	1.75	0.49	0.06	0.07	0.07	0.30
Topologie 11	AVG	32.08	2.78	1.30	**0.35**	**0.35**	2.30
(13, 38)	STD	10.27	1.84	0.47	0.19	0.19	1.24
Topologie 12	AVG	4.15	0.86	1.07	**0.11**	**0.10**	1.33
(32, 94)	STD	1.51	0.83	0.45	0.12	0.12	0.44
Topologie 13	AVG	23.49	21.73	3.26	**3.01**	**3.01**	3.96
(8, 24)	STD	49.33	52.36	6.42	6.52	6.52	6.23
Moyenne sur	AVG	2091	5.39	85.44	1.31	**1.2**	279.35
les topologies	STD	461.75	6.42	160.22	1.28	1.17	414.35

TAB. 3.2 – Ecart relatif avec le partage optimal des flots , fonction coût (3.1) .

Le tableau 3.3 présente les augmentations relatives des temps de calcul des algorithmes FTA et $A^{\min}_{\mathrm{MS}}$ par rapport au temps de calcul de l'algorithme $A^{\mathrm{stop}}_{\mathrm{MS}}$. Comme précédemment, les résultats sont

moyennés sur les 8 matrices de trafic définies pour chaque topologie. L'algorithme d'optimisation des métriques incrémental A_{INCR} est bien évidemment beaucoup plus rapide que l'algorithme multi-start.

Topologie	FTA (%)	$A_{\mathrm{MS}}^{\mathrm{min}}$ (%)	Topologie	FTA (%)	$A_{\mathrm{MS}}^{\mathrm{min}}$ (%)
1	1511 ± 501	123 ± 47	8	891 ± 428	78 ± 41
2	961 ± 446	34 ± 22	9	1408 ± 478	103 ± 58
3	2277 ± 1446	42 ± 22	10	4027 ± 1377	45 ± 13
4	1906 ± 1889	86 ± 53	11	766 ± 140	35 ± 9
5	2303 ± 1112	104 ± 35	12	923 ± 304	47 ± 22
6	1404 ± 1174	70 ± 90	13	1924 ± 976	87 ± 27
7	843 ± 168	66 ± 29			

TAB. 3.3 – Moyenne et écart type de l'augmentation de temps de calcul comparé à $A_{\mathrm{MS}}^{\mathrm{stop}}$

L'ensemble de ces résultats montrent que les algorithmes proposés sont toujours plus rapides que FTA et proposent toujours de meilleures solutions. Sur les 104 tests, l'algorithme $A_{\mathrm{MS}}^{\mathrm{min}}$ (respectivement $A_{\mathrm{MS}}^{\mathrm{stop}}$) atteint 17 fois (resp. 17 fois) la borne inférieure et est 67 fois (resp. 66 fois) à moins de 1% de cette borne inférieure. Il peut également être noté que l'algorithme A_{INCR} améliore toujours de manière significative la solution initiale (métriques égales à 1) et qu'il est relativement proche de la borne inférieure pour 11 des 13 topologies.

Nous proposons maintenant la même étude pour un autre critère : l'utilisation maximale des interfaces. Cependant, comme ce coût n'est pas continu, aucune solution ne peut être trouvée par l'algorithme de déviation des flots permettant de résoudre le problème de répartition optimale des flots. Cependant, comme l'algorithme $A_{\mathrm{MS}}^{\mathrm{min}}$ a toujours trouvé une solution égale à la borne inférieure, nous proposons de le prendre comme référence. Les résultats sont proposés dans le tableau 3.4 qui montre une efficacité certaine malgré le changement de critère.

Topologie	UNIT (%)	INVCAPA (%)	$A_{\mathrm{MS}}^{\mathrm{stop}}$ (%)	FTA (%)	A_{INCR} (%)
1	18.45 ± 22.19	19.67 ± 4.19	0 ± 0	2.41±4.73	0±0
2	637.25±310.87	41.59±16.36	0±0	140.47±272.71	158.12±355.28
3	129.69±32.93	36.44±7.43	1.21±3.21	32.90±28.56	25.75±18.15
4	28181±5755	17.03±14.36	0±0	6036±11643	164.27±50.33
5	33.46±9.14	59.45±20.25	0±0	8.58±15.76	0±0
6	14464±19535	40.64±123.61	0±0	8772±7392	7753±9264
7	144.68±50.25	39.55±24.92	0.10±0.25	9.89±13.82	0.92±1.97
8	169.92±98.22	28.65±17.3	11.93±21.23	36.63±35.28	18.95±19.22
9	53.17±27.33	40.18±15.58	2.28±2.84	11.21±6	2.14±2.4
10	18.24±22.7	36.31±33.35	0±0	0±0	0±0
11	75.23±20.36	32.91±18.57	0±0	20.54±10.59	13.79±8.41
12	29.30±18.33	43.89±8.68	1.27±1.48	5.84±4.64	0±0
13	13.44±19.6	14.7±19.3	0±0	0.82±1.7	31.50±77.16

TAB. 3.4 – Ecart relatif par rapport à $A_{\mathrm{MS}}^{\mathrm{min}}$, coût utilisé (3.4).

3.4.3 Intégration de la résilience dans l'optimisation

Les résultats proposés jusqu'ici ne montrent que l'efficacité des algorithmes proposés sans tenir compte des contraintes de résilience. Pour se faire, nous proposons dans cette partie de traiter un cas particulier de réseau d'opérateur. En effet, nous avons pu travailler avec certains opérateurs durant mon doctorat et l'un d'entre eux s'est posé un problème de planification sur une topologie réelle qu'il était en train de mettre en place. Nous utiliserons cette topologie pour tester et valider l'approche intégrant la résilience dans l'optimisation des métriques.

La topologie considérée est illustrée par la figure 3.6. C'est donc un réseau à l'échelle européenne contenant 26 PE (« Provider Equipment Node »), 8 P (« Provider Node »), 4 concentrateurs VPN, 2 switchs et 5 serveurs. De plus, la topologie est associée à 20 SAP (« Service Access Point ») représentés par des points bleus et qui seront les sources des trafics considérés. De manière plus générale, cette topologie contient donc 45 nœuds (nous comptons les switchs comme des nœuds IP même si cela n'est pas logique car ces switchs ont été placés pour permettre d'affiner le routage des SAP connectés dessus), et 78 liens (donc 156 interfaces IP). Les serveurs et chaque SAP sont les seuls sources et puits de trafic.

Les métriques proposées initialement par l'opérateur sont les suivantes : 10000 sur les interfaces associées aux liens secondaires (entre PE et concentrateur VPN) et 1000 sur l'ensemble des autres interfaces. Les liens secondaires étant des liaisons de plus faible débit (IPSec loué à un autre opérateur et avec moins de garantie).

Du coup, après un premier routage des trafics sur cette topologie, on trouve une utilisation maximale légèrement supérieure à 49% et une utilisation des seuls liens primaires et du cœur de réseau. La visualisation graphique de l'utilisation nominale peut être vue de manière colorée sur la figure 3.7 la légende des couleurs étant affichée sous le nœud « P Lyon 2 ».

Si nous exécutons l'algorithme $A^{\min}_{\mathrm{MS}}$ sans tenir compte de la résilience, nous obtenons une solution qui tend à répartir les flots sur l'ensemble des liens qu'ils soient primaires ou secondaires. L'utilisation maximale vaut 27.7% soit un écart relatif de plus de 40% avec la valeur précédente. Cette solution est trouvée à l'aide de la solution initiale ou toutes les métriques sont égales à 1.

Remarque 3.2 *Les métriques proposées par A_{INCR} sous les mêmes hypothèse mènent à une utilisation maximale des interfaces valant 30.1%. Ce résultat est donc très proche de la meilleure solution possible.*

La figure 3.8 montre les utilisations des interfaces les plus chargées pour les deux répartitions proposées : solution nominale proposée par l'opérateur (figure 3.8(a)) et celle proposée par notre algorithme (figure 3.8(b)).

L'optimisation proposée permet une diminution nette de l'utilisation maximale des interfaces par une faible augmentation de l'utilisation moyenne qui passe de 5% à 5.4%. Cependant, cette solution est obtenue de part l'utilisation des liens secondaires pour le cas nominal. D'un autre côté, ne pas utiliser ces liens alors qu'ils existent nous semble être une aberration de conception. Pour information, la même

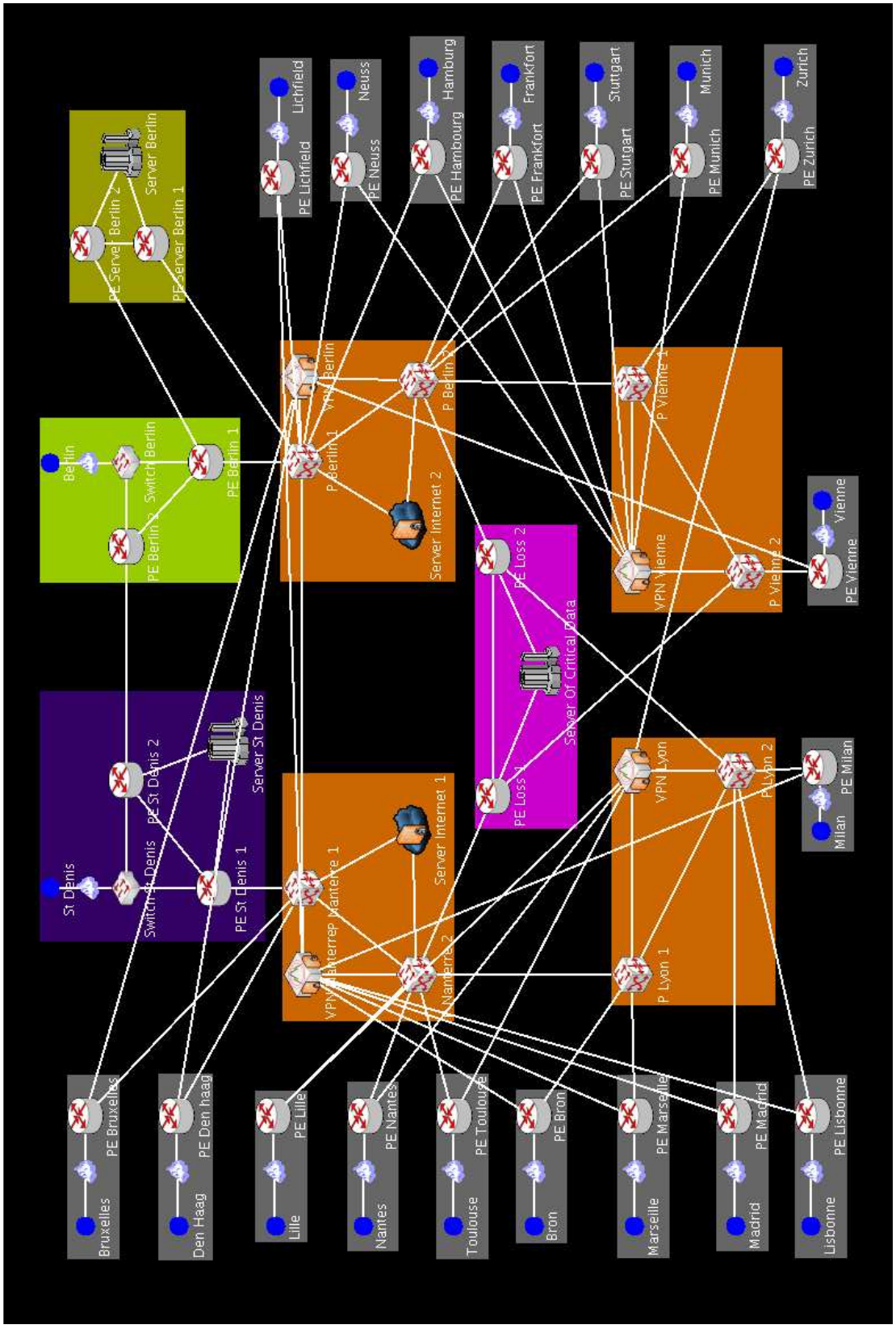


FIG. 3.6 – Topologie européenne de référence

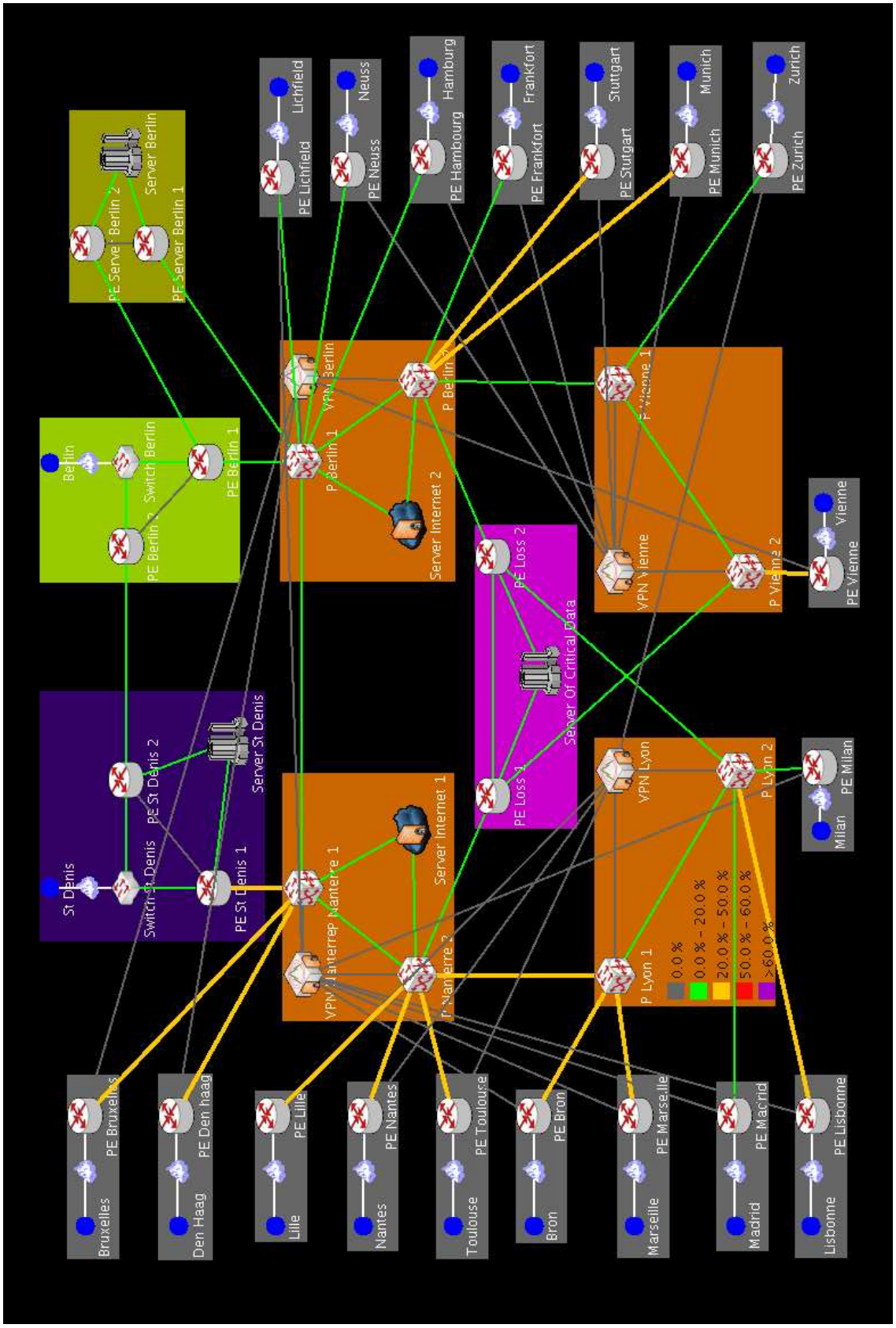
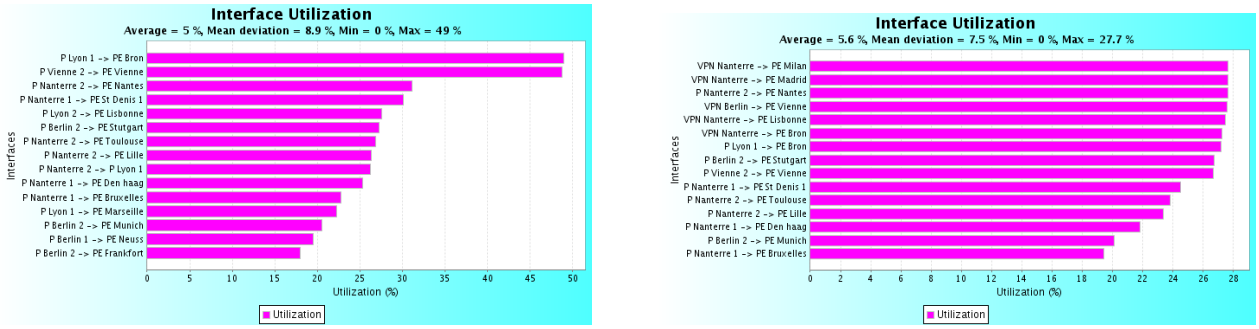


FIG. 3.7 – Routage initial et utilisation des ressources



(a) Solution proposée par l'opérateur

(b) Solution proposée par l'algorithme $A_{\mathrm{MS}}^{\min}$

FIG. 3.8 – Utilisations des 15 interfaces les plus chargées avant et après optimisation par $A_{\mathrm{MS}}^{\min}$

topologie configurée avec des métriques égales à 1 est au maximum chargée à 50.7% et sa configuration par des métriques Cisco inversement proportionnelles à leur capacité entraîne un coût de 44.9%.

Nous proposons alors de tester les résultats obtenus si l'on tient compte des pannes dans le réseau. Un premier résultat relativement attendu est que si l'analyse de la résilience n'intervient que pour valider une meilleure solution nominale, dès lors la solution proposée est identique à celle que nous venons d'exposer. En effet, minimiser l'utilisation maximale des interfaces sous contrainte de ne pas empirer les cas de panne n'a pas vraiment d'intérêt dans un réseau adapté aux situations de panne de part sa biconnexité et son architecture.

Cependant, si nous proposons d'optimiser l'espérance de l'utilisation maximale des interfaces pour tous les cas de panne possibles et ceci en utilisant les valeurs de probabilité de panne proposées par Cisco ([75]) alors la solution proposée est différente. Ainsi, la figure 3.9 propose une visualisation de l'utilisation des interfaces obtenues suite à la mise en place de la solution proposée. Nous pouvons voir que l'utilisation maximale obtenue est proche de 30% donc la solution proposée est moins bonne dans le cas nominal que celle obtenue auparavant.

Il nous faut tout de même noter que les temps de calcul deviennent conséquents étant donné que chaque évaluation de voisin revient à évaluer l'ensemble des pannes possibles. Pour information, sur un ordinateur de bureau classique muni d'un processeur cadencé à 3Ghz, nous avons obtenu les temps de calcul suivants pour notre algorithme en mode multi-start :

- 1 minute pour optimiser le critère sans contrainte de résilience,
- 30 minutes pour optimiser le critère sous contrainte de ne pas l'empirer en cas de panne, et
- 48 heures pour optimiser l'espérance du critère sur tous les états de pannes possibles.

Il est possible de réduire les temps de calcul lors de l'évaluation des pannes possibles à l'aide du calcul dynamique du routage et de la propagation. L'algorithme de Ramalingam et Reps du calcul dynamique des routes peut être adaptée à la suppression d'un nœud ou d'un lien (cela revient à fixer un ensemble de métriques égale à l'∞). Cependant, cette dernière étape n'a pas été réalisée lors de nos

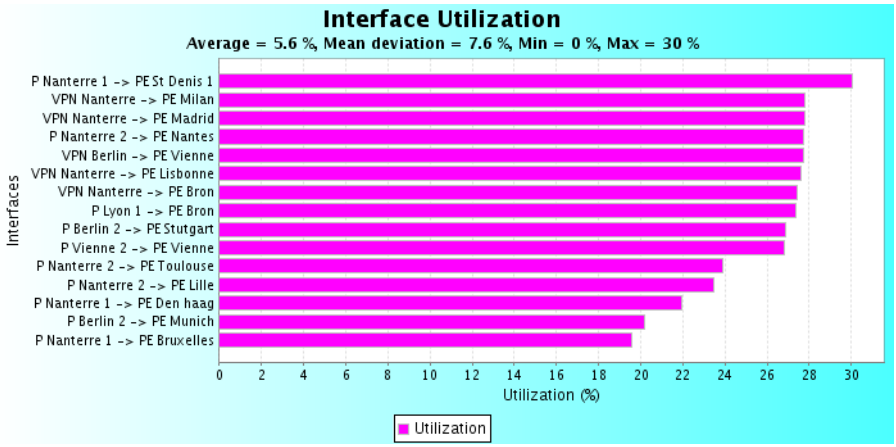


FIG. 3.9 – Utilisations des interfaces avec les métriques optimisant l'espérance du critère

travaux ce qui explique cette différence majeure dans les temps de calcul présentés.

3.5 Conclusion

Nous avons donc développé de nouveaux algorithmes d'optimisation des métriques permettant d'obtenir de bons résultats dans des temps de calculs raisonnables. Les contributions apportées portent sur plusieurs points clés de l'ingénierie des trafics :

Approche incrémentale : nous proposons un algorithme qui fournit une solution de bonne qualité basée sur le configuration initiale du réseau étudié. La suite de modifications proposée peut être appliquée en totalité ou de manière partielle tout en conservant la garantie que le critère sélectionné sera amélioré.

Approche multi-start : nous proposons également un algorithme dédié à la planification des réseaux qui propose une solution très souvent optimale que l'opérateur peut ensuite mettre en place dans son réseau avant de le rendre opérationnel.

L'ensemble des algorithmes proposés ont des résultats proches de l'optimum dans la majorité des cas testés. Leurs temps de calcul sont très courts comparés aux algorithmes existant, cela provient de trois choix majeurs :

1. Une notion de voisinage de taille réduite mais bien adapté au problème
2. L'intégration d'un routage dynamique suite à un changement de métrique
3. L'utilisation de la propagation dynamique des flots

Nous avons également traité la résilience des réseau IP/MPLS en l'associant au problème du choix optimal des métriques IP. Pour cela, nous avons proposé une modélisation des états associés à un réseau IP/MPLS. Partant de cette modélisation, nous avons proposé d'intégrer la résilience dans l'optimisation

des métriques IP et d'adapter nos algorithmes de départ pour les deux nouveaux modèles :

Contraintes : la première modélisation proposée consiste à éviter la dégradation de la résilience du réseau suite aux modifications des métriques.

Espérance : la seconde modélisation proposée porte sur une optimisation de l'espérance du critère sur l'ensemble des états associés au réseau.

A notre avis, le routage par ToS des mécanismes DiffServ intégré au problème d'optimisation du routage IP est une perspective intéréssante à developper. Pour être optimale, une telle approche doit être globale et optimiser en même temps l'ensemble des métriques du réseau. Une extension de l'algorithme proposé semble possible par une multiplication du nombre d'inconnues par le nombre de ToS ainsi que part une distinction des demandes par ToS.

CHAPITRE 4

Multi Protocol Label Switching (MPLS)

Les limitations du routage IP ont déjà été évoquées plus tôt dans les chapitres 1 et 3. Elles portent principalement sur les aspects suivants :

- Garanties de QoS difficiles à fournir.
- Ingénierie des trafics et optmisation de l'utilisation des ressources limitées.
- Tables de routage conséquentes pouvant conduire à un temps de consultation parfois trop long.

Ainsi, en 1996, Ipsilon proposait IP Switching qui avait pour but de réduire le temps d'interrogation des tables de routage. Plus tard, dans la même année, Cisco annonça sa propre solution Tag Switching pour commuter l'IP. Ces deux propositions furent à la base de l'établissement du protocole MPLS basé sur la commutation de labels, à opposer à la classique commutation de paquets ayant lieu dans IP.

MPLS est un protocole se situant entre les couches 2 et 3 du modèle OSI (Open Systems Interconnection) qui interagit complètement avec IP, c'est pourquoi nous parlerons principalement de réseau IP/MPLS. Il permet ainsi un routage sur des chemins préétablis, les Label Switching Path (LSP), composés de commutateurs. L'action ayant lieu en chaque point intermédiaire le long du chemin n'est donc plus le maniement parfois complexe d'une table de routage mais une commutation de labels se révélant être bien plus rapide.

L'avantage majeur de MPLS dont les RFC principales ([55]) et ([56]) ont été publiées en janvier 2001 portait donc sur la rapidité associée à la commutation de labels et sur ses possibilités d'ingénierie bien plus développées qu'avec IP. Aujourd'hui, de part l'apparition de routeurs aux capacités de traitement de plus en plus performantes, ce sont surtout ses possibilités d'ingénierie qui justifient l'adoption massive de MPLS.

Nous proposons une description succinte de MPLS au début de ce chapitre, puis nous donnerons les principaux algorithmes utilisés pour le placement des LSP sous certaines contraintes. Nous présenterons ensuite les solutions proposées par notre équipe de recherche pour le placement des LSP.

MPLS est la solution de routage utilisée par la majorité des opérateurs pour améliorer les performances de leur réseau et qui leur permet une analyse plus aisée de ces performances par la notion de LSP associé à une bande passante reservée. Cependant, le routage IP optimisé peut s'avérer plus performant que la solution MPLS si l'ingénierie des LSP est trop simpliste. De plus, la modification du routage IP sous-jacent entraîne le plus souvent des évolutions sur les performances MPLS par une évolution des routes MPLS basées sur les plus courts chemins IP. Ainsi, le but de ce chapitre est de montrer

- que le routage IP optimisé est compétitif par rapport à la solution MPLS dans de nombreux cas de configuration,
- que l'impact de l'optimisation des métriques sur les performances MPLS est majoritairement positif, et
- qu'une solution très proche du routage optimal peut être obtenue par MPLS à l'aide d'un routage explicite des LSP.

4.1 Multi Protocol Label Switching (MPLS)

Nous proposons dans cette partie un descriptif rapide des caractéristiques du protocole MPLS. Le routage à commutation de labels, l'ingénierie des trafics, les techniques dédiées à la résilience, le placement des LSP sont autant de différences majeures avec le protocole IP que nous détaillerons ici.

4.1.1 Le routage MPLS : commutation de labels

La figure 4.1 illustre un réseau IP/MPLS constitué d'un ensemble de routeurs IP (en rouge) et d'un cœur de réseau IP/MPLS (entouré en bleu) composé lui de LER (Label Edge Router) à la frontière (en bleu) et de LSR (Label Switched Router en noir) qui sont tous des routeurs IP/MPLS. La figure montre également un LSP (en vert) permettant d'aller d'un LER vers un autre. Les équipements IP permettant la connexion au cœur de réseau MPLS sont communément appelés des « Customer Equipments »(CE).

A l'entrée du cœur de réseau, les LER (ingress) vont marquer les paquets avec un label avant de les propager. Le choix de ce label provient directement des informations de routage contenues dans ce routeur. Pour cela, MPLS se sert de FEC (« Forwarding Equivalence Class ») qui sont des tables de correspondance dont les clés sont des éléments du paquet : adresses IP ou adresses MAC, classe de service ou application (port), etc. . .

Une fois le label récupéré, l'ingress encapsule le paquet IP dans un paquet MPLS avec un nouvel en-tête composé principalement de ce label. La propagation se fait alors par commutation de labels. Les nœud MPLS possèdent une table de commutation mise en place lors de la sélection des chemins pour les LSP. Ces tables de commutations ont pour clés : l'interface d'entrée et le label, et pour valeurs : l'interface de sortie et le nouveau label. L'en-tête MPLS est alors remplacé par un nouvel en-tête composé du nouveau label, puis le paquet est propagé sur l'interface de sortie donnée par la table de commutation. Le dernier nœud le long du LSP (egress) va retirer l'en-tête MPLS du paquet et le propager ensuite en IP.

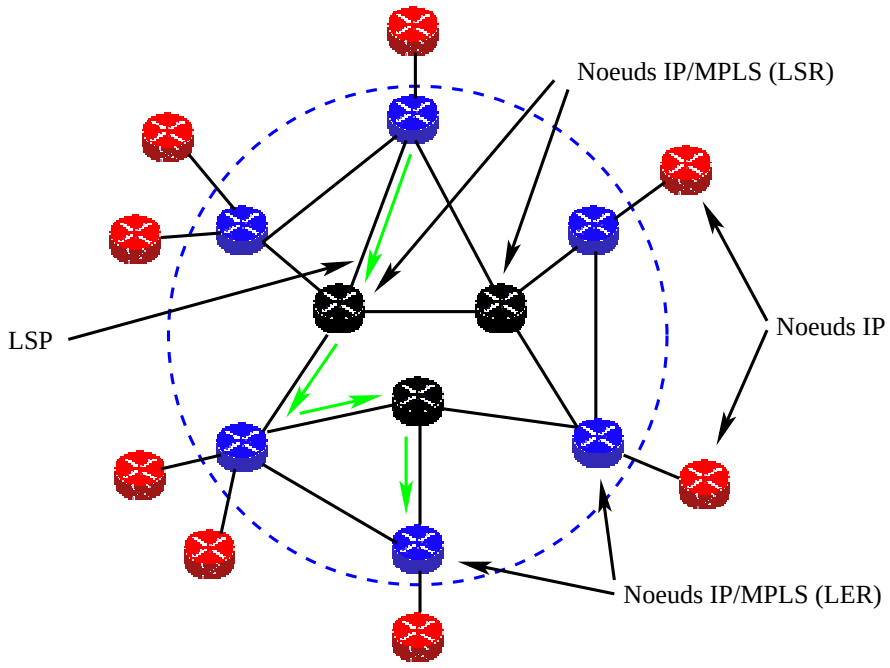


FIG. 4.1 – Exemple de réseau IP/MPLS

4.1.2 Les atouts de MPLS

L'avantage majeur du protocole MPLS et de permettre un routage particulier pour chacun des LSP et donc d'associer un chemin (éventuellement différent du plus court chemin utilisé par les IGP) à chaque groupe de flots considéré. La granularité du choix des routes est donc améliorée permettant ainsi une meilleure gestion de la QoS et surtout une ingénierie de trafic plus aisée.

Mais les avantages de MPLS ne s'arrête pas à ces améliorations. Il permet aussi un grand nombre de fonctionnalités très poussées :

Une intégration IP/ATM : en effet, si la couche 2 est gérée par le protocole ATM (ou Frame Relay), MPLS permet l'utilisation de l'en-tête de couche 2. Cela évite l'ajout d'un en-tête supplémentaire et permet donc de diminuer la taille des données de signalisation.

La flexibilité : MPLS permet l'utilisation de n'importe quelle couche 2 car ce protocole vient se placer entre la couche réseau et la couche de transmission de données. L'ajout d'une couche supplémentaire permet de coexister avec l'ensemble des protocoles existants (surtout pour la couche 2). De plus, il est possible de déployer un réseau IP/MPLS sur des infrastructures sous-jacentes hétérogènes (Ethernet, ATM, SDH, WDM, etc...).

La création de VPN : la notion de réseau privé virtuel s'appuie complètement sur le paradigme de routage intermédiaire proposé par MPLS. Il devient ainsi facile d'intégrer des VPN sur le cœur

de réseau IP/MPLS de part l'affectation de labels directement associés aux VPN.

Approche DiffServ : l'architecture d'un cœur de réseau IP/MPLS formé de LER aux frontières et de LSR dans le cœur est identique à celle retenue dans DiffServ nécessitant des fonctionnalités différentes aux frontières et dans le cœur. De plus, l'association des trafics aux FECs permet et facilite l'affectation des différentes classes de trafics aux files d'attente dédiées. On retrouve ainsi dans MPLS la notion de « Per Hop Behaviour (PHH) » de l'approche DiffServ.

L'ingénierie de trafic : MPLS est souvent associé à son ingénierie de trafic (Traffic Engineering : TE) car la notion de routage explicite des LSP permet une simplification de la mise en place d'un routage calculé au préalable. Ce point sera développé dans la partie suivante.

4.1.3 Ingénierie de trafic

Dans un cœur de réseau IP/MPLS, le placement des LSP définit donc complètement l'utilisation des ressources de la même manière que le choix des métriques IP dans le cas de réseaux IP. L'affectation d'un chemin à un LSP peut être faite de différentes manières. Elle est

- **automatique** et gérée par un algorithme exécuté sur l'ensemble des routeurs IP/MPLS,
- **partielle** en précisant un ou plusieurs nœuds IP/MPLS par lequel (lesquels) le LSP doit passer, le reste de la route étant calculé par un algorithme exécuté sur les routeurs, ou
- **explicite**, c'est à dire que l'ensemble du chemin est configuré par l'opérateur.

Le routage explicite des LSP, est l'atout majeur de MPLS qui en fait un outil d'ingénierie de trafic très puissant. Le protocole IP souffre de difficultés importantes quant à l'ingénierie des trafics : la relation entre métriques et utilisation des ressources n'est pas évidente (voir chapitre 3), et le routage par destination impose des routes communes à partir d'un routeur si la destination est la même. Avec MPLS, des chemins différents des plus courts chemins peuvent être choisis et associés à un ensemble de flots dès lors qu'ils font partis de la même FEC.

De plus, il devient possible de calculer hors ligne des chemins en tenant compte de critères plus ou moins complexes. En effet, MPLS permet une gestion de la QoS en intégrant de manière simple les approches IntServ et DiffServ. Pour DiffServ par exemple, la gestion de FEC différenciées par classe de service permet de choisir des chemins différents pour chacune des classes amenant ainsi une gestion plus soutenue de la QoS qu'avec le routage IP.

4.1.4 Fonctionnalités dédiées à la résilience

Tous les LSP sont affectés à des chemins primaires par une des techniques exposées plus tôt. Si aucune précaution n'est prise, lorsqu'une panne rend ce chemin inutilisable, le routage IP est utilisé pour joindre l'egress du LSP. Il existe cependant une caractéristique importante de MPLS qui concerne ses fonctionnalités dédiées à la résilience. Permettant une réaction rapide aux pannes, ces deux fonctionnalités sont

- « **Backup Path** » **:** un LSP peut être configuré avec un chemin de secours disjoint du chemin primaire. Ce chemin de « backup » est affecté par l'une des trois techniques citées plus tôt. En cas de panne sur le chemin primaire, le chemin de secours est
  - calculé puis utilisé pour des configurations automatiques ou partielles,
  - directement utilisé dans le cas d'une configuration explicite.
- « **Fast Reroute** » **:** pour chaque équipement protégé (lien ou routeur) le long du chemin primaire du LSP, un chemin de contournement est établit au préalable. Si un élément du chemin primaire tombe en panne, il est contourné à l'aide du chemin prédéfini. L'utilisation des « Fast Reroute » est le plus souvent temporaire. Elle permet d'éviter une interruption de service lors de la mise en place du chemin de secours.

Un cœur de réseau IP/MPLS ou chaque LSP est protégé (configuration d'un chemin de secours) et où chaque élément du réseau est associé à un (ou plusieurs) « Fast Reroute » permet donc de garantir un service sans interruption quel que soit l'état du réseau sauf dans les cas de pannes multiples.

4.1.5 Affectation des chemins aux LSP

Le placement automatique des LSP s'effectue par un échange de labels entre les routeurs du réseau IP/MPLS. Différents protocoles existent et leurs différences principales portent sur les aspects de QoS et de routage explicite qu'ils offrent. Cependant, il faut bien noter que tous ces protocoles s'appuient sur le routage IGP existant. Ainsi le choix des routes IP peut grandement influencer le placement des LSP dans certains cas. Nous détaillons quelques protocoles et algorithmes de placement dans la suite de cette partie.

Protocoles de placement des LSP

Label Protocol Distribution (LDP [57]) est le plus simple des protocoles effectuant la mise en place des labels MPLS. Il permet un établissement des chemins sans tenir compte d'aucune contrainte. Son principe est le suivant : il sélectionne un des plus courts chemins au sens de l'IGP présent.

Le choix des chemins associés aux LSP définissant complètement l'utilisation des ressources du cœur de réseau, il est indispensable d'intégrer des contraintes en terme de bande passante lors de la sélection des chemins si des garanties de QoS doivent être assurées. Dans cet optique, une extension de LDP : Constraint-based Routing Label Distribution Protocol (CR-LDP [63]), permet une intégration des bandes passantes associées aux LSP et leurs réservations le long du chemin du LSP.

Resource ReSerVation Protocol Traffic Engineering (RSVP-TE [58]) est également un protocole qui permet un routage des LSP en tenant compte des bandes passantes sur les chemins choisis.

La reservation de bande passante lors de la mise en place d'un nouveau LSP se fait à l'aide d'une négociation de bout en bout en tenant compte des reservations existantes. Les LSP sont donc placés dans l'ordre de leur apparition (configuration par l'opérateur) sans jamais remettre en cause un placement

déjà établi. En conséquence, certaines demandes peuvent être refusées à cause de placement antérieurs des autres LSP.

PCALC (Path CALCulation) de Cisco

Cisco propose une implémentation du protocole OSPF-TE pour la distribution des labels et l'établissement automatique des LSP sur le réseau. Ce protocole, comme OSPF, est basé sur la notion de plus courts chemins au sens des métriques associées au réseau. Ces métriques sont dynamiques car elles dépendent de la charge courante du réseau. Cet algorithme se base sur la notion de bande passante résiduelle associée à un chemin qui correspond au minimum des bandes passantes résiduelles des liens qui le composent.

L'algorithme de sélection d'un chemin proposé par Cisco est le suivant

1. Sélectionner l'ensemble des plus courts chemins compte tenu de l'IGP présent.
2. Sélectionner parmi ceux-ci les chemins ayant la bande passante résiduelle maximale.
3. Sélectionner parmi ceux-ci ceux ayant le nombre de sauts minimum.
4. S'il reste plus d'un chemin candidat, choisir un chemin au hasard parmi ceux restants.

L'avantage principal de cet algorithme est de pouvoir fonctionner en ligne. Il est exécuté sur les routeurs MPLS Cisco et permet de trouver un chemin tenant compte de l'état du réseau à chaque nouvelle demande de LSP qu'un LER enregistre. Cependant, cette vision limitée aux chemins partant du LER initiant la demande ne permet pas une gestion optimale des ressources.

Etant donné la confiance des opérateurs envers Cisco ainsi que le nombre de routeurs de ce constructeur qui sont actifs, nous pouvons dire que c'est un algorithme très répandu actuellement dans les réseaux IP/MPLS.

Remarque 4.1 *Tous les protocoles et algorithmes de routage des LSP que nous venons de présenter ont une particularité commune : ils se basent sur les plus courts chemins calculés par l'IGP présent dans le cœur de réseau. Sachant que LDP et PCalc sont les deux protocoles les plus utilisés pour le routage des LSP, la question de l'impact des métriques sur MPLS semble justifiée.*

ILSP-OLS-ACO

L'équipe de recherche dans laquelle j'ai effectué mon doctorat a proposé un algorithme novateur et performant permettant le calcul hors ligne des chemins des LSP et ceci en tenant compte d'un nombre important de contraintes. Les lecteurs intéressés par ce sujet pourront se référer à la thèse de Mohammed Anouar Rachdi [88] ainsi qu'à [72] dans laquelle il formule le problème de routage des LSP, et propose les solutions que je vais résumer dans cette partie.

L'algorithme proposé permet de tenir compte d'un ensemble de contraintes de QoS plus ou moins simples. Ainsi, trois critères sont évalués :

1. Une somme quadratique des charges des interfaces (avec pénalité si une charge dépasse la capacité de l'interface)
2. Une somme pondérée des délais et pertes sur les interfaces (délais et pertes calculées à partir du modèle M/M/1/N)
3. S'appuyant sur un modèle plus complexe de file d'attente (le modèle LLQ proposé par Cisco [91]), l'optimisation cherche à minimiser le premier critère exposé mais sous des contraintes de délais et de pertes bornés sur chaque file (il y en a 4 en tout dans le modèle LLQ, une prioritaire et 3 gérées par une politique « Weighted Fair Queuing »). Ainsi le critère proposé tient compte de la charge mais aussi des écarts aux contraintes et ceci de manière quadratique et pénalisée.

Le problème posé est donc de trouver un et un seul chemin pour chaque LSP (associé à une demande) tout en minimisant un des critères proposés plus tôt. La résolution de ce problème non linéaire et connu pour être NP difficile peut se faire à l'aide de l'algorithme ILSP-OLS-ACO composé de trois étapes :

1. **ILSP :** La première partie de cet algorithme (Iterative Loading Shortest Path ILSP) génère un sous ensemble de chemins possibles pour chaque LSP. ILSP prend en compte les contraintes du problème pour garantir qu'une solution où chaque LSP est routé sur l'un des chemins du sous-ensemble calculé au préalable est admissible.

   Cette propriété fait d'ILSP la partie la plus importante de cet algorithme. En effet, il devient par exemple aisé
   – de savoir s'il existe au moins un chemin admissible pour chaque LSP,
   – de trouver des couples de chemins disjoints admissibles,
   – d'éliminer certains chemins pour respecter des contraintes de couleur (routage limité à un ensemble de liens).

2. **OLS :** La seconde étape (Optimal Load Sharing OLS) de l'algorithme consiste à trouver la (ou les) route(s) minimisant la fonction coût par une répartition optimale des demandes des LSP sur les chemins associés (calculés au préalable par ILSP). Pour se faire, l'utilisation d'un gradient projeté est adaptée et efficace. La projection se fait sur les contraintes de conservation des demandes. La solution trouvée peut associer un chemin unique à chaque LSP (dans ce cas, le problème est résolu), mais il peut y avoir encore non unicité des chemins optimaux trouvés. Une dernière étape est alors nécessaire.

3. **ACO :** Si OLS ne mène pas à une solution de mono routage, la troisième et dernière étape (Ant Colony Optimization ACO) permet de converger vers la solution de mono routage cherchée. C'est une méthode itérative basé sur les techniques d'optimisation par colonies de fourmis. Ainsi, à chaque itération, les fourmis construisent une solution consistant à associer à chaque LSP un et un seul chemin. Le choix du chemin est aléatoire et dépend des choix précédents et de la quantité de phéromones associée à cette décision. Nous ne donnerons pas plus de détail ici mais la convergence est assurée et les résultats fournis par cette heuristique se sont montrés très bons dans le

cadre des problèmes combinatoires sur le routage des paquets dans un réseau.

A. Rachdi propose également dans son mémoire [88] une évolution de son algorithme permettant de trouver des chemins de secours et des Fast Reroute tout en assurant le respect des contraintes sur ces nouveaux LSP proposés. Ainsi, la garantie de QoS que proposent ses algorithmes de placement est assurée même en cas de panne.

Remarque 4.2 *ILSP traite tous les chemins possibles entre chaque couple ingress/egress. Les plus courts chemins calculés par l'IGP sont donc étudiés par ILSP mais ils ne sont pas forcement retenus. Ainsi, le routage final composé de chemins retenus par ILSP uniquement ne dépend pas des métriques IP.*

Nous avons donc montré dans cette partie qu'il existe des algorithmes performants permettant le calcul (en ligne ou hors ligne) de chemins optimaux pour le placement des LSP. Nous avons aussi insisté sur le fait que la majorité des algorithmes de placement ne choisissent les chemins des LSP que parmi les plus courts chemins au sens des métriques IP. Seul l'algorithme proposé par A. Rachdi permet une abstraction de l'IGP présent. Ainsi, MPLS se révèle être un protocole extrêmement efficace pour l'ingénierie de trafic, la gestion de la QoS et/ou la gestion du routage. Cependant, son efficacité dépend grandement de la configuration du protocole IP sous-jacent. Cette dépendance peut certainement être utilisée dans le but d'améliorer le routage IP et le routage MPLS en même temps.

4.2 Optimisation du routage IP

L'utilisation massive aujourd'hui du protocole MPLS nous pousse à remettre en question l'utilité de l'algorithme d'optimisation des métriques IP présenté dans le chapitre précédent. Nous proposons donc de comparer, sur différents réseaux de test, la solution de l'optimisation des métriques avec une solution MPLS simplement configurée.

L'ingénierie MPLS qui peut être associée aux LSP possède de nombreuses fonctionnalités et se traduit principalement par la définition de FEC. Néanmoins, seules quelque-unes sont réellement utilisées par la majorité des opérateurs parmi lesquelles :

- la definition d'un ensemble de destinations
- la définition d'un ensemble de classes de service (CoS)

Pour mémoire, une fois qu'une FEC est définie et associée à un LSP, les trafics propagés le long de ce chemin (le LSP) seront ceux dont les caractéristiques correpondent à la FEC. Définir des FEC revient donc à aggréger les flots sur des chemins identiques (les LSP) entre les couples ingress / egress, et cela peut être fait d'une manière plus ou moins fine.

Dans tous les exemples que nous traitons, nous supposons qu'il existe des LSP entre chaque couple ingress / egress du réseau, nous parlons alors de configuration « full mesh » de MPLS. Deux configurations MPLS sont étudiées dans cette partie : une FEC unique et des FEC regroupant des classes de

service. Le placement des LSP est effectué de manière automatique par PCalc. Ces deux exemples de configuration représentent la majorité des configurations MPLS utilisées de nos jours.

Les bandes passantes associées à chaque LSP représentent les demandes réelles circulant sur ces LSP. Un algorithme d'ingénierie de trafic utilisable dans NEST nous permet en effet de prédéterminer la charge de chaque LSP en fonction des FEC et du trafic ecoulé sur le réseau.

Le routage MPLS ne peut être défini que dans le cœur de réseau IP/MPLS. Pour cette raison, l'optimisation des métriques que nous réalisons n'est pas autorisée à changer des métriques en dehors de ce cœur de réseau. De plus, l'évaluation des critères ne portera que sur les utilisations des interfaces contenues dans ce même cœur.

Le tableau de la figure 4.2(b) présente les résultats obtenus (utilisations moyennes et maximales des interfaces) pour les différents routages

- IP avec métriques initiales (noté IP),
- IP optimisé par A_{INCR} (noté A_{INCR}),
- MPLS configuré avec une FEC unique (noté FEC-1),
- MPLS configuré avec plusieurs FEC regroupant des classes de services (noté FEC-COS),

et sur différentes topologies :

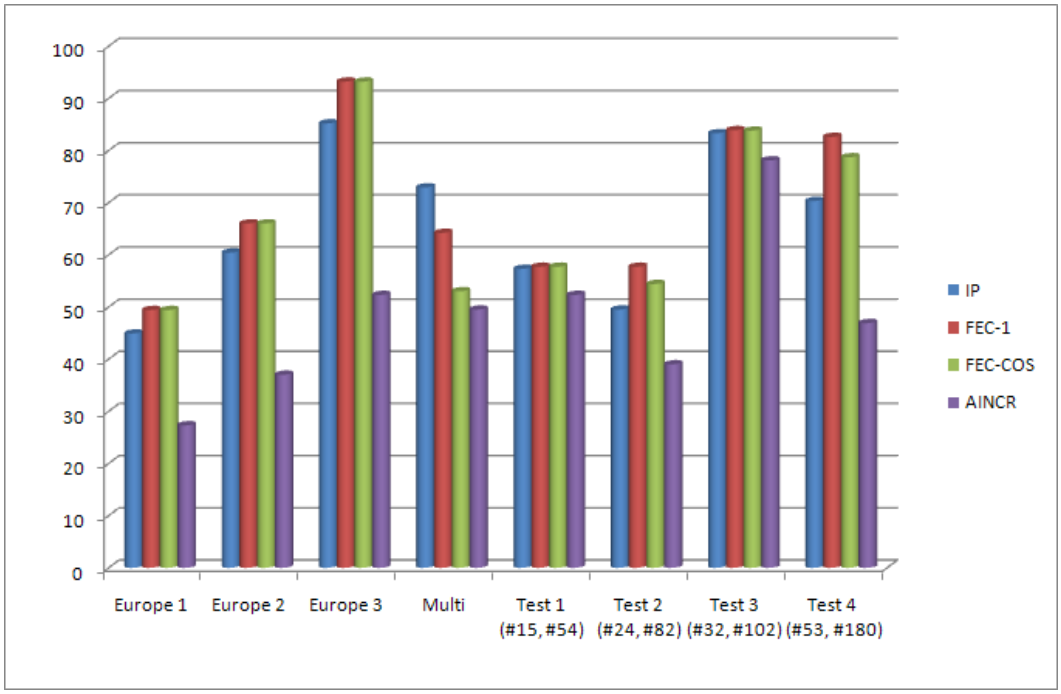
- **Europe 1** : la tolopologie européenne illustrée dans la figure 3.6 dans le chapitre précédent avec une utilisation maximale proche de 50% et des métriques IP de type Cisco,
- **Europe 2** : la même topologie mais avec plus de trafic menant à une utilisation maximale initiale supérieur à 60%
- **Europe 3** : toujours la même topologie mais avec une utilisation maximale proche des 85%
- **Multi** : une topologie réelle composée de multiliens et illustrée par la figure 3.1(a) dans l'introduction du chapitre précédent (les métriques sont égales à 1), et
- **Test** i **(#**n**, #**i**)** : des topologies de tests aléatoires où n est le nombre de nœuds et i le nombre d'interfaces et où toutes les métriques sont égales à 1.

La figure 4.2(a) illustre graphiquement les différences obtenues sur les utilisations maximales des interfaces pour les tests effectués.

On remarque alors que le routage optimisé est toujours meilleur qu'une solution MPLS. Cependant, dans un test (Multi) le routage MPLS avec des FEC spécifiées par classe de service donne des résultats proches du routage optimisé. Néanmoins, MPLS dégrade les performances dans les autres tests par rapport au routage IP original et une configuration des FEC unique donne toujours de moins bons résultats qu'une configuration de FEC plus nombreuses.

Ces faibles performances de MPLS peuvent s'expliquer facilement :

1. Tout d'abord, PCalc cherche à placer les LSP sur les plus courts chemins. L'ensemble des possibilités est ensuite filtré. Cependant, aucune route autre qu'un plus court chemin ne pourra être choisie par PCalc.



(a) Utilisations maximales (en %)

Topologies	IP		A_{INCR}		FEC-1		FEC-COS	
	moy	max	moy	max	moy	max	moy	max
Europe 1	5%	44.9%	5.5%	**27.3%**	5.4%	49.4%	5.4%	49.4%
Europe 2	6.8%	60.4%	7%	**37%**	6.9%	66%	6.9%	66%
Europe 3	9.6%	85.2%	9.8%	**52.3%**	9.7%	93.2%	9.7%	93.2%
Multi	9.6%	72.9%	10.2%	**49.5%**	9.5%	**64.2%**	9.6%	**53%**
Test 1 (#15, #54)	13.7%	57.3%	14.5%	**52.3%**	14.9%	57.7%	14.9%	57.7%
Test 2 (#24, #82)	11.8%	49.5%	12.2%	**39%**	13.7%	57.7%	13.7%	54.4%
Test 3 (#32, #102)	27%	83.3%	27%	**78.1%**	27.2%	83.9%	27.2%	83.8%
Test 4 (#53, #180)	14.6%	70.3%	14.4%	**46.9%**	15.1%	82.6%	15.1%	78.7%

(b) Utilisations moyennes (moy) et maximales (max)

FIG. 4.2 – Utilisations des interfaces pour différents routages

2. La présence d'un en-tête supplémentaire lors de la propagation MPLS est la cause principale des différences lorsque celles-ci sont faibles.
3. L'optimisation du routage cherche à créer des situations de partage de charge alors que l'utilisation de MPLS impose un routage le long de ses LSP. Le cas FEC-1 correspond donc à une situation de mono routage d'où ses performances plus faibles qu'en IP pour l'ensemble des situations. Le cas FEC-COS permet l'utilisation de plus de chemins mais son nombre est limité et dépend des CoS des trafics entre chaque couple ingress / egress. C'est donc ce routage imposé sur un faible nombre de chemins qui entraîne des performances plus faibles que le routage IP.

De part les tests effectués et une connaissance des problèmatiques liées au routage, on se rend bien compte que MPLS peut être une solution redoutable si le nombre de chemins est augmenté. Cela impose donc une définition de plus de FEC ce qui complique tout de même la configuration de MPLS par rapport à une optimisation du routage. Cela pose également un nouveau problème : déterminer le nombre et la composition des FEC pour améliorer les critères de performance.

L'optimisation du routage apparaît donc comme une solution permettant de concilier de meilleures performances du routage et une configuration simple de ce routage. Il est alors légitime de se demander si une configuration simple de MPLS peut être améliorée par une optimisation des métriques IP.

4.3 Impact des métriques IP sur le routage MPLS

Pour étudier l'impact de l'optimisation des métriques IP sur les performances MPLS, nous utiliserons les mêmes topologies que précédemment. Pour chacune d'entre elles, nous étudions les performances obtenues suite à un routage sur les LSP placés par PCalc avec des métriques IP optimisées par A_{INCR}.

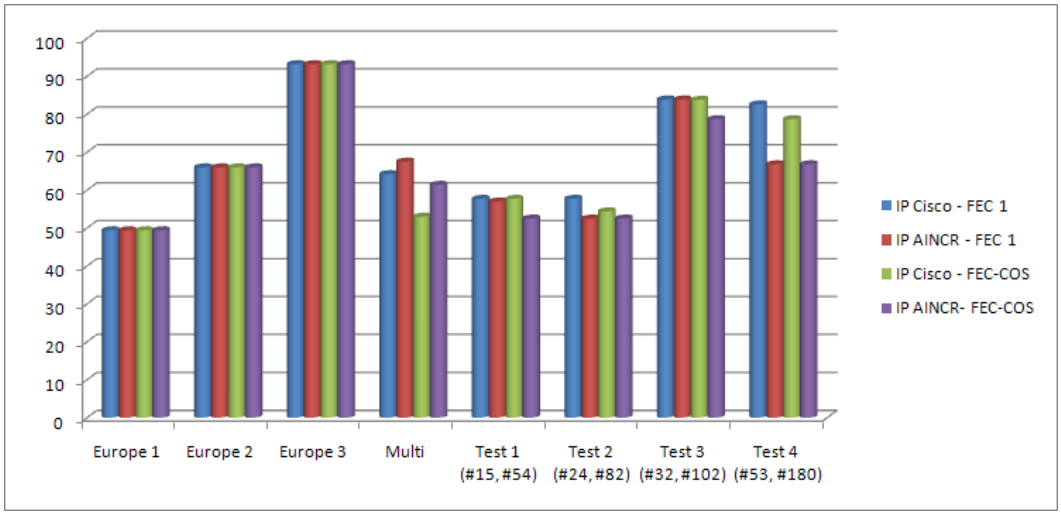
Le tableau de la figure 4.3(b) présente une comparaison entre les résultats obtenus précédemment avec MPLS (dans le tableau de la figure 4.2(b)) et ceux que l'on obtient avec le routage MPLS mais à partir du routage IP optimisé. La même comparaison est illustrée par la figure 4.3(a) pour les utilisations maximales des interfaces. On s'aperçoit alors que pour les 4 derniers tests, nous obtenons une amélioration des performances MPLS grâce à l'optimisation des métriques. Dans les autres cas, les résultats sont plus mitigés mais on ne dégrade jamais les performances d'un facteur trop important.

L'optimisation des métriques semble donc être bénéfique à MPLS dans le cas du routage automatique. Pour aller plus loin, nous proposons d'étudier plus en détail les possibilités de routage sur la topologie européenne (Europe 1).

Les routages IP utilisés sont les suivants :
- **Initial** : routage initial où des métriques valent 1000 (liens principaux) ou 10000 (liens de backup)
- **Unitaire** : routage où toutes les métriques sont égales à 1
- **Cisco** : routage où les métriques sont inversement proportionnelles à leur capacité
- **OptIncr** : routage optimisé par A_{INCR}.
- **OptMS** : routage optimisé par $A_{\text{MS}}^{\min}$.

Pour les routages optimisés par les algorithmes proposés dans le chapitre précédent, nous avons interdit des modifications de métrique sur les interfaces hors du cœur de réseau IP/MPLS. C'est pourquoi les résultats proposés sont différents de ceux présentés dans le chapitre précédent.

Pour chaque routage étudié, le tableau de la figure 4.4(b) présente les utilisations des interfaces moyennes et maximales obtenues pour les différents placements étudiés (LDP, PCalc, ILSP) dans le cas d'une configuration avec une seule FEC. Le tableau de la figure 4.5(b) présente de son côté les résultats obtenus avec la définition de 4 FEC. Les figures 4.4(a) et 4.5(a) illustrent les utilisations maximales des



(a) Utilisations maximales (en %)

Topologies	Métriques IP Cisco				Métriques optimisées par A_{INCR}			
	FEC-1		FEC-COS		FEC-1		FEC-COS	
	moy	max	moy	max	moy	max	moy	max
Europe 1	5.4%	49.4%	5.4%	49.4%	5.5%	49.4%	5.5%	49.4%
Europe 2	6.9%	66%	6.9%	66%	7%	66%	7%	66%
Europe 3	9.7%	93.2%	9.7%	93.2%	9.8%	93.2%	9.8%	93.2%
Multi	9.5%	64.2%	9.6%	53%	10.2%	67.5%	10.1%	61.4%
Test 1 (#15, #54)	14.9%	57.7%	14.9%	57.7%	15.7%	57%	15.7%	**52.5** %
Test 2 (#24, #82)	13.7%	57.7%	13.7%	54.4%	14.1%	52.5%	14.1%	**52.5%**
Test 3 (#32, #102)	27.2%	83.9%	27.2%	83.8%	28%	83.9%	28%	**78.7%**
Test 4 (#53, #180)	15.1%	82.6%	15.1%	78.7%	15.3%	66.8%	15.3%	**66.8%**

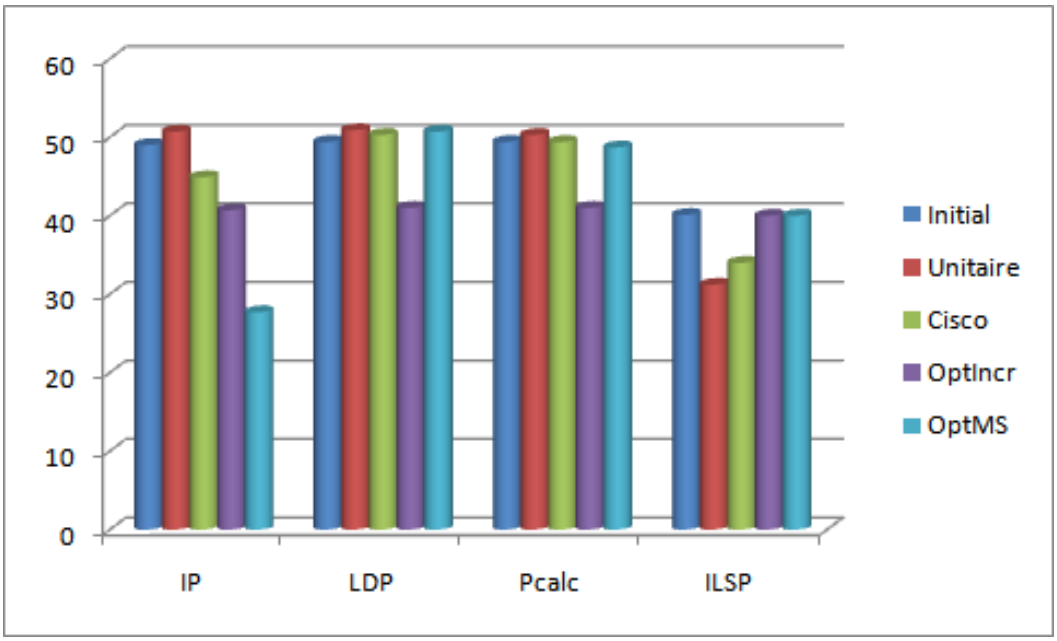
(b) Utilisations moyennes (moy) et maximales (max)

FIG. 4.3 – Utilisations des interfaces obtenues par MPLS pour deux configurations de métriques IP

interfaces respectivement pour une FEC et pour 4 FEC.

Remarque 4.3 *On remarque que les résultats proposés par ILSP dépendent du routage IP étudié ce qui est contradictoire avec la définition d'ILSP faite dans la première partie. En réalité, ce sont les valeurs des trafics entre chaque couple ingress / egress qui évoluent à cause des modifications de métriques IP faites dans le cœur de réseau.*
Il pourrait être intéressant de proposer une optimisation des métriques IP telle que la matrice de cœur de réseau ne change pas. Ainsi, les modifications du routage proposées permettraient une amélioration des performances dans le cœur de réseau sans impacter la bande passante des LSP.

L'ensemble de ces tests, auxquels il faut ajouter de nombreux tests non évoqués ici, permettent de tirer les conlusions suivantes :



(a) Utilisations maximales (en %)

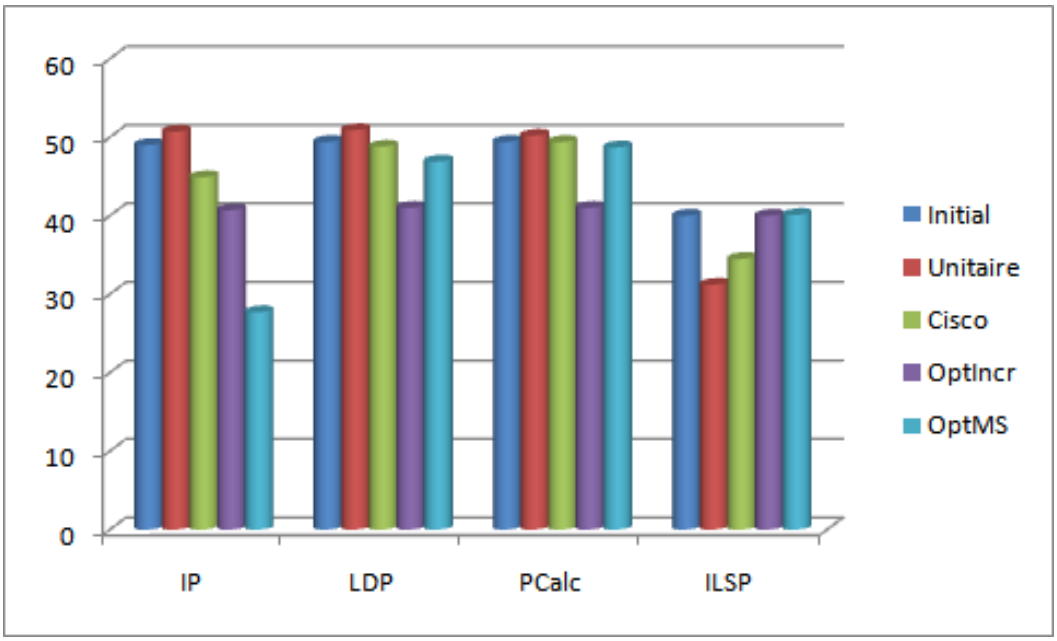
	IP		LDP		PCalc		ILSP	
Métriques IP	Moy	Max	Moy	Max	Moy	Max	Moy	Max
Initial	5.4%	49%	5.4%	49.4%	5.4%	49.4%	6.8%	**40.1%**
Unitaire	5.9%	50.7%	5.9%	50.9%	5.8%	50.3%	6.9%	**31.2%**
Cisco	5.4%	44.9%	5.4%	50.3%	5.4%	49.4%	7%	**34%**
OptIncr	5.4%	40.7%	5.4%	**41%**	5.4%	**41%**	6.9%	**40%**
OptMS	6%	27.7%	6%	50.7%	5.9%	48.7%	7%	**40%**

(b) Utilisations moyennes (moy) et maximales (max)

FIG. 4.4 – Utilisations des interfaces pour différentes configurations du routage IP/MPLS avec une seule FEC

– L'optimisation des métriques proposés dans le chapitre précédent apporte de nettes améliorations au routage MPLS dans certains cas mais ne le détériore pas dans les autres. Il est donc intéréssant de l'utiliser pour des cœurs de réseaux IP/MPLS.
– L'utilisation du routage MPLS n'améliore pas l'utilisation des ressources si le placement des LSP est automatique.
– L'utilisation de LSP de manière optimale (ILSP) permet une amélioration nette des performances.

Cependant, certains avantages majeurs non visibles dans nos exemples doivent être détaillés :
– Même si les routeurs deviennent de plus en plus performants, les routeurs présents dans la plupart des réseaux ne peuvent être changés pour des raisons économiques. Les temps de commutation de label restent donc très comptétitifs par rapport aux temps de routage IP.



(a) Utilisations maximales (en %)

	IP		LDP		PCalc		ILSP	
Métriques IP	Moy	Max	Moy	Max	Moy	Max	Moy	Max
Initial	5.4%	49%	5.4%	49.4%	5.4%	49.4%	6.9%	**40%**
Unitaire	5.9%	50.7%	5.9%	50.9%	5.8%	50.2%	7.1%	**31.2%**
Cisco	5.4%	44.9%	5.4%	48.8%	5.4%	49.4%	7.1%	**34.5%**
OptIncr	5.4%	40.7%	5.4%	**41%**	5.4%	**41%**	7.6%	**40%**
OptMS	6%	27.7%	6%	**46.9%**	5.9%	**48.7%**	6.9%	**40.1%**

(b) Utilisations moyennes (moy) et maximales (max)

FIG. 4.5 – Utilisations des interfaces pour différentes configurations du routage IP/MPLS avec 4 FEC

– La vision sous forme de LSP est beaucoup plus pratique à traiter pour un opérateur. Elle lui permet de distinguer ses différents clients par exemple. Elle permet surtout une simplification lors des études faites sur le réseau.
– Nous avons vu que le routage explicite des LSP donne des résultats presque aussi performant que l'optimisation des métriques. Il peut donc être intéréssant d'optimiser le routage de ses LSP plutôt que le routage IP.

4.4 Ingénierie de trafic MPLS

Nous avons utilisé MPLS bien en dessous de ses capacités et ceci de manière volontaire car peu d'opérateurs utilisent une configuration plus complexe de MPLS. Les atouts principaux de MPLS permettant la mise en place d'un routage quasi-optimal et robuste sont :

– La couleur des liens : chaque interface MPLS du réseau peut être associée à une couleur (sur 32 bits) permettant de la caracteriser. Lors de la configuration des LSP, l'opérateur peut spécifier une affinité et un masque définissant la(les) couleur(s) des liens que le LSP pourra utiliser pour être routé. Pour chaque lien testé, un « et » binaire est réalisé entre la couleur du lien et le masque et si le resultat est égal à l'affinité alors le lien peut être utilisé.

  Il est ainsi possible de choisir les routes de manière plus fine sans passer par le routage explicite car les algorithmes automatiques de placement des LSP tels que CR-LDP ou PCalc permettent l'utilisation de ces couleurs. Cette fonctionnalité de MPLS est plus simple à configurer mais elle demande plus de travail pour trouver quelles valeurs de couleurs, d'affinités et de masques il faut utiliser pour obtenir le routage souhaité.

– Le routage explicite des LSP en tenant compte de contraintes de QoS : il permet donc de faire de l'ingénierie de trafic avec MPLS dans des conditions optimales. Il permet surtout de traiter l'ensemble des demandes en même temps à l'inverse du routage dynamique qui optimise le placement d'un nouveau LSP par rapport aux LSP déjà placés sans remettre en question leurs placements.

  L'utilisation d'un outil comme NEST associé à un éditeur de configuration peut également faciliter la configuration des routes calculées hors ligne par un algorithme comme ILSP-OLS-ACO. L'ingénierie de trafic proposée par MPLS peut donc s'avérer particulièrement performante de part un choix très précis des routes pour des ensembles de trafics choisis.

– Les chemin de secours et les « Fast Reroute » : l'utilisation du « Fast Reroute » est de courte durée jusqu'à l'établissement du chemin de secours si celui-ci est calculée de manière dynamique. Mais de même que pour les chemins primaires, le calcul des chemins de secours et des « Fast Reroute » peut être fait hors ligne pour permettre leurs configurations explicites tout en respectant les contraintes données.

  Si le routage des chemins de secours et des « Fast Reroute » est explicite, il est alors possible de tenir compte des groupes de liens à risques partagés (SRLG). Toutes les routes de secours seront alors disjointes des routes primaires quelle que soit la panne d'un SRLG. Cela permet de s'abstraire de chemin de secours d'apparence disjoint (lien IP) mais en réalité lié au primaire (circuit virtuel ATM commun par exemple).

4.5 Conclusion

Ce chapitre a présenté MPLS qui a connu un succès important ces dernières années grâce à des temps de commutation plus rapides que les temps de routage IP. Cependant, cette différence, significative il y a quelques années, tend à se resorber avec l'apparition des « giga » routeurs.

Nous avons montré dans ce chapitre que le routage au travers de LSP dans le cœur de réseau MPLS n'était pas plus efficace que le routage IP si les métriques sont optimisées par l'algorithme proposé

dans le chapitre précédent. De plus, l'impact de l'optimisation du routage sur les performances MPLS apparait positif. Mais cela n'est pas suffisant.

L'utilisation du routage explicite des LSP est la solution qui nous parait la plus efficace. Un routage optimal se doit d'être construit à partir de MPLS et

- d'une définition précise de FEC,
- d'une définition de couleurs et d'affinité,
- d'un calcul hors ligne
  - des routes optimales pour chaque LSP,
  - des chemins de secours et des « Fast Reroute » ,
- et de la prise en compte de contraintes multiples.

Cependant, une telles configuration du routage nécéssite des outils performants permettant la modélisation du réseau, la connaissance des trafics, le calcul optimal des LSP et l'édition des configurations des routeurs correspondantes. Ces outils commencent à se développer et le développement du logiciel NEST est justement fait dans cette direction.

En attendant de pouvoir utiliser MPLS au maximum de ses capacités, l'optimisation des métriques reste la solution qui nous semble la plus efficace. Sa simplicité de configuration et ses résultats prometteurs poussent à l'utilisation de l'algorithme développé dans le chapitre précédent même si le protocole MPLS est présent. De plus, son extension au routage par ToS permet également une aggrégation fine des flots à l'image de la définition des FEC.

CHAPITRE 5

Dimensionnement de Topologie

Les solutions que nous avons proposées jusqu'ici dans les chapitres 3 et 4 avaient comme particularité principale de permettre une amélioration des ressources existantes sans surcoût supplémentaire. En effet, l'attrait des techniques d'optimisation du routage est de permettre une amélioration de la QoS sur le réseau sans avoir à modifier les équipements même de ce réseau.

Cependant, du fait d'une constante évolution de la demande en trafic, il est souvent nécessaire de faire évoluer le réseau lorsque l'optimisation du routage ne permet plus d'atteindre le niveau de performance cible. Si les ressources ne sont pas suffisantes, aucune ingénierie ne pourra permettre de garantir une QoS acceptable.

Le dimensionnement de topologie connu sous le nom de « capacity planning » est un des problèmes auxquels les opérateurs ont à faire face dans ce cas là. L'objectif est d'affecter à chaque équipement du réseau une capacité optimale. La littérature scientifique contient de nombreux travaux portant sur le dimensionnement des réseaux et le plus souvent, le problème posé consiste à trouver l'allocation de capacités aux liens permettant de minimiser le coût du réseau tout en respectant certaines contraintes de performances.

Dans [9], Kleinrock développe le problème en proposant de minimiser le délai moyen de bout en bout des paquets sur le réseau tout en ne dépassant pas un budget donné. Cependant, minimiser le délai moyen ne permet pas d'assurer un délai acceptable pour chaque flot. Dans les années 70, Meister et Al. proposent dans [5], et [6] la minimisation d'une somme pondérée des délais élevés à une certaine puissance. Cette minimisation est toujours faite sous contrainte de budget constant. Pour permettre la distinction de classes de service dans le réseau, Maruyama and Tang dans [10] propose en 1976 de minimiser une somme pondérée des délais avec des poids dépendant de la classe des paquets.

L'ensemble de ces approches considère un coût total relativement simple formé par la somme des coûts des liens. Ce dernier dépendant simplement de la capacité choisie pour le lien. Mais l'avantage principal de ces méthodes est la linéarité du coût utilisé. Aussi des techniques basées sur les multiplica-

teurs de Lagrange par exemple permettent une résolution efficace de ces problèmes.

Aborder le problème avec des fonctions coûts plus réalistes et avec des notions de priorités sur les paquets le rend plus complexe de part la non linéarité ajoutée. Dans ce cadre là, Maruyama et Al. proposent d'ajouter au problème des bornes sur le délai dépendant de la classe des paquets. Ainsi une nouvelle formulation du problème est proposée dans [12]. Les auteurs cherchent à résoudre simultanément le choix des capacités et l'affectation des priorités aux différentes classes. Plus récemment, en 1999, Runggeratigul et Al. proposent dans [49] une fonction coût permettant de mieux prendre en compte les relations entre la capacité et les technologies en définissant une fonction coût continue par morceau ou chaque domaine de continuité correspond à une technologie précise (Fibre optique, cuivre, etc...).

Le problème d'allocation des capacités a également été traité en parallèle du problème d'allocation des flots. Ainsi, le problème CFA (Capacity and Flow Allocation) traite en même temps la répartition des flots sur le réseau (Multicommodity Flow Problem) et l'affectation des capacités en conséquence. Duhamel et Mahey propose justement dans [59] une résolution de ce problème basée sur une décomposition de Benders.

L'ensemble des travaux antérieurs supposent un coût total du réseau égal au seul coût des liens. L'approximation consistant à négliger le coût des autres équipements était jusqu'alors justifiée. En effet, l'installation d'un lien était une étape extrêmement coûteuse comparée à l'achat d'un routeur ou d'une carte pour brancher le lien.

Cependant, dans le contexte actuel où la fibre optique est largement répandue et où la majorité des liens sont loués à des fournisseurs de bande passante, changer la capacité d'un lien peut se faire plus simplement et consiste en

- une révision du contrat de location passé avec le fournisseur en bande passante, et
- un changement des cartes (« Physical Interface Card (PIC) ») sur les routeurs où le lien est connecté.

De plus, les solutions proposées jusqu'alors ne tiennent pas compte des contraintes matérielles portant sur les équipements utilisés comme le nombre de slots sur un routeur. Il n'existe donc aucune garantie quant à la mise en place effective d'une telle solution.

Partant de ce constat, notre contribution au problème de dimensionnement de réseaux est double :

Coût des équipements : nous proposons une nouvelle définition du problème de dimensionnement qui tient compte des coût réels des équipements qui seront choisis. Ainsi, de part une définition fine des coûts, nous proposons un dimensionnement optimisé avec un coût réaliste (chiffré en K€) très proche de ce que l'opérateur aura à investir pour mettre son réseau à niveau.

Contraintes matérielles : comme notre approche tient compte des PICs permettant la connexion des liens sur les routeurs, elle permet d'intégrer complètement les contraintes des équipements : nombre de slots sur un routeur, nombre de ports sur une carte, capacité de traitement d'un routeur. De part l'intégration de ces contraintes, les solutions que nous proposons sont directement configurables et ceci est un aspect essentiel pour les opérateurs de réseaux.

5.1 Une nouvelle approche du dimensionnement

Nous proposons donc dans ce chapitre de poser le problème de dimensionnement d'une manière nouvelle. Ce nouveau problème nommé par nos soins « Capacity Assignement Problem with Hardware Considerations (CAH) » (ou le problème d'affectation de capacités sous contraintes matérielles) sera complètement défini dans cette partie.

5.1.1 Topologie

Nous noterons N le nombre de routeurs de la topologie, et L le nombre de liens bidirectionnels qu'elle contient. On suppose avoir E routeurs frontières donc $c = E(E-1)$ couples Origine Destination (OD). Il sera utile de pouvoir considérer l'ensemble des liens $U(n)$ connectés au routeur n.

Pour chaque lien l, nous noterons $e_1(l)$ et $e_2(l)$ ses nœuds extrémités. De plus, deux interfaces physiques sont configurées sur chaque lien l (une de $e_1(l)$ vers $e_2(l)$, et une autre en sens inverse), elles composent alors l'ensemble $i(l) = \{i_{e_1(l)\to e_2(l)}, i_{e_1(l)\to e_2(l)}\}$. Le nombre total d'interfaces sera noté I ($I = 2L$). Pour chacune des interfaces i, soient C_i sa capacité et Y_i sa charge.

Nous supposons le routage connu et résumé dans la matrice $A = (a_{i,f})$ de telle manière que $a_{i,f}$ représente la proportion du trafic associé au couple OD f qui est propagée sur l'interface i.

5.1.2 Variables du problème

Nous notons T le nombre de modèles de lien et pour chacun d'entre eux (noté t), C_t sera sa capacité. Le problème que nous cherchons à résoudre peut être vu comme un problème de décision. Nous noterons $\delta_{l,t}$ l'ensemble des variables de décision du problème :

$$\delta_{l,t} = \begin{cases} 1 & \text{Si le modèle } t \text{ est affecté au lien } l \\ 0 & \text{Sinon.} \end{cases} \tag{5.1}$$

La figure 5.1 illustre la manière dont les liens sont branchés sur les routeurs à l'aide des cartes adaptées. Chaque lien du réseau est connecté à ses deux routeurs extrémités par une carte (PIC) offrant le même débit par port que celui du modèle choisi pour le lien. Soit Z le nombre de cartes utilisables, nous affirmons que $Z \geq T$ car il existe plusieurs cartes pour un modèle de lien donné. Chaque carte z sera associée à

- son nombre de ports $p(z)$,
- son modèle $t(z)$, et
- son débit total maximal $r(z)$.

Nous avons donc de manière évidente $r(z) = p(z)C_{t(z)}$ (le débit de la carte est celui du modèle multiplié par le nombre de ports sur la carte).

La configuration de cartes sur un routeur, c'est à dire le nombre de cartes de chaque modèle, est une variable importante qui dépend de l'affectation δ choisie. Nous noterons donc $K(\delta, n) = (k_1, k_2, \ldots, k_Z)$ le vecteur de dimension Z tel que k_z représente le nombre de cartes de modèle z

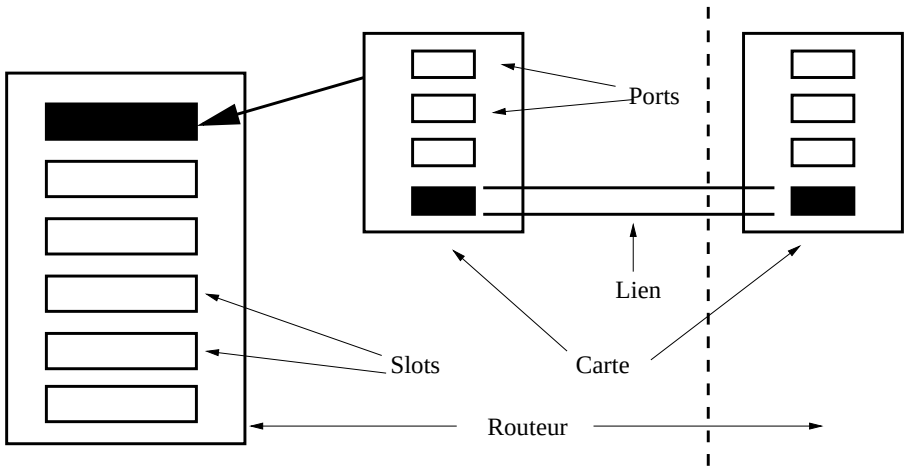


FIG. 5.1 – Branchement des liens sur les routeurs à l'aide des cartes

branchées sur le routeur n. De plus, pour pouvoir tenir compte des cartes déjà présentes sur les routeurs, nous noterons $K^{init}(n)$ la configuration de cartes initiale du routeur n.

Nous supposons qu'aucun modèle de routeur ne sera changé. Chaque routeur est initialement associé à un modèle qui ne changera pas, en conséquence, aucun coût ne sera défini pour les routeurs et nous connaissons dès le départ pour un routeur n

- son nombre de slots que nous notons $S(n)$
- ainsi que son débit maximal autorisé que nous notons $\lambda(n)$

5.1.3 Coût des cartes

Comme nous avons pu l'introduire rapidement dans le début de ce chapitre, une de nos contributions principales porte sur l'aspect économique du modèle que nous proposons. En effet, grâce à nos collaborations avec différents opérateurs, nous proposons une formulation réaliste.

Le coût d'une carte (PIC) sera celui de son achat : nous supposons simplement qu'un opérateur voulant connecter une nouvelle carte doit tout d'abord se l'approprier. Aussi nous noterons Γ_z^C le coût d'une carte de type z.

Ainsi, nous pouvons donner le coût $M(K,n)$ d'une configuration de cartes $K = (k_1, k_2, \ldots, k_Z)$ sur le routeur n en tenant compte de la configuration de cartes initiale sur ce routeur ($K^{init}(n)$) car une carte déjà présente ne coûte rien :

$$M(K,n) = \sum_{z=1}^{Z} \left(\max(0, k_z - k^{init}(n)_z) \ \Gamma_z^C \right) \tag{5.2}$$

Le choix du vecteur d'affectation δ défini complètement les besoins matériels sur chaque routeur. Ainsi, il est possible de trouver la configuration de cartes optimale $K^*(\delta, n)$ permettant de brancher les liens sur le routeur n à un coût $M^*(\delta, n)$ optimal. Nous revenons sur ce point dans la partie 5.2.

5.1.4 Coût des liens

Les opérateurs se tournent aujourd'hui pour la plupart vers des fournisseurs de bande passante qui leur permettent de mettre en place des liens IP d'une capacité donnée. Nous proposons une formulation du coût d'un lien qui respecte pour beaucoup la réalité (aux ristournes et offres promotionnelles près). Comme le coût que nous proposons est le pire coût que devra payer l'opérateur à son fournisseur, nous garantissons par contre que notre minimisation donnera une borne supérieure du coût réel.

Ainsi, le coût d'un lien de modèle t sera défini à partir des éléments suivants :

- Γ_t^L qui est le coût d'installation du lien. Ce coût ne doit être pris en compte que si le modèle du lien a changé.
- Γ_t^F qui est un coût fixe mensuel (dépendant de la longueur).
- Γ_t^D qui est un coût kilométrique mensuel (dépendant de la longueur).
- $w_l(\delta)$ qui vaut 1 si le modèle du lien l est changé et 0 sinon. Dans le cas où aucun modèle n'était choisi au départ, cette variable vaudra 1.

Notons $\Gamma(\delta, l)$ le coût d'un lien l dans la solution δ :

$$\Gamma(\delta, l) = \sum_{t=1}^{T} \delta_{l,t} \left(w_l(\delta)\Gamma_t^L + \Omega \left(\Gamma_t^F + \Gamma_t^D D(l) \right) \right) \tag{5.3}$$

Où $D(l)$ est la longueur du lien en kilomètre et Ω la durée de la période d'étude. Une fois Ω donné, cette fonction coût représente donc le coût réel en K€ qu'un opérateur a à payer à la fin de la période de temps Ω. On remarque donc que notre fonction coût est non nulle même si le lien l ne change pas de modèle.

Les coûts dépendant de la longueur du lien sont souvent donnés par tranche de distance : le coût sera le même pour un lien de 10 ou 20 Km par exemple alors qu'il sera moins cher pour un lien de 100Km. Nous ne nous étendrons pas plus sur ce point mais nous préciserons les seuils utilisés lors de nos tests.

5.1.5 Coût du réseau

Le coût total du réseau $\Gamma(\delta)$ peut donc être calculé à partir de la somme des coûts des cartes et des liens :

$$\Gamma(\delta) = \sum_{n=1}^{N} M^*(\delta, n) + \sum_{l=1}^{L} \Gamma(\delta, l) \tag{5.4}$$

5.1.6 Délai des flots et contraintes associées

Pour chaque couple OD f, en notant P_f l'ensemble de ses chemins, d_p^{path} le délai le long d'un chemin p, et β_p la proportion du trafic associé au couple OD f qui est propagée le long du chemin p, nous pouvons écrire :

$$a_{i,f} = \sum_{p \in P_f} \gamma_i^p \beta_p$$

Où $\gamma_i^p = 1$ si l'interface i appartient au chemin p, 0 sinon. Nous posons l'hypothèse que le délai sur l'interface i est $\frac{1}{C_i - Y_i}$ (nous nous référons à [8] pour appuyer cette hypothèse). Nous utiliserons alors le délai moyen de bout en bout pour un couple OD. Pour cela, nous pondérons les délais moyens de chaque chemin par leurs coefficients de partage. Nous pouvons alors calculer le délai pour un couple OD f :

$$d_f^{flow} = \sum_{p \in P_f} \beta_p d_p^{path} = \sum_{p \in P_f} \sum_{i \in p} \frac{\beta_p}{C_i - Y_i} = \sum_{i=1}^{I} \frac{a_{i,f}}{C_i - Y_i} \tag{5.5}$$

Nous proposons de contraindre les délais des flots, pour cela nous considérerons Q classes de services pour l'ensemble des flots sur le réseau. Nous supposons que chacune des classes possède une borne propre sur le délai moyen de bout en bout de ses paquets : les délais des paquets de classe q ne devront pas être supérieurs à d^q.

Nous notons $X^q = (X_f^q)_{0 \leq f \leq c}$ la matrice contenant les valeurs des demandes en Kbps circulant entre chaque couple OD pour la classe de service q. Dès lors, pour chaque couple OD f, nous posons une contrainte de délai qui peut s'écrire :

$$\forall \text{ OD } f \quad \sum_{i=1}^{I} \frac{a_{i,f}}{C_i - Y_i} \leq d^{max}(f) \tag{5.6}$$

Où :

$$d^{max}(f) = \min_{q / X_f^q \neq 0} d^q$$

est le délai maximum autorisé pour le couple OD f. Cette contrainte est plus lache qu'une contrainte portant sur un délai maximum pour chaque chemin associé à un couple. Cependant, elle permet de contraindre le délai moyen tout en minimisant le nombre de contraintes et d'inconnus associées. Dans notre contexte, cela permet d'éviter d'alourdir encore plus les notations déjà complexes.

5.1.7 Contraintes matérielles

Un et un seul modèle doit être affecté à chaque lien. Par conséquent les variables de décision sont reliées par la contrainte suivante :

$$\forall l = 1..L \quad \sum_{t=1}^{T} \delta_{l,t} = 1 \tag{5.7}$$

Le nombre de liens connectés sur une carte de type z est limité au nombre de ports $p(z)$ de ce modèle de carte comme nous l'avions illustré dans la figure 5.1. De plus, le nombre de cartes qui peuvent être installées sur un routeur n est limité par le nombre de slots de ce routeur $S(n)$. En se souvenant que $K^*(\delta, n) = (k_1^*, k_2^*, \ldots, k_Z^*)$ est la configuration de cartes optimale présente sur le routeur n pour la solution δ, nous avons pour tout modèle de lien t et pour tout routeur n :

$$\sum_{l \in U(n)} \delta_{l,t} \leq \sum_{z/t(z)=t} K_z^* p(z) \tag{5.8}$$

Le terme de gauche représente le nombre de ports qu'il faudrait pour brancher les liens de modèle t sur le routeur n et le terme de droite représente lui le nombre de ports disponibles du fait des cartes installées.

Les débits sur les routeurs sont également des contraintes à prendre en compte. Ainsi, le débit total des cartes branchées sur un routeur (terme de gauche) ne doit pas dépasser le débit maximal que supporte ce routeur (terme de droite) :

$$\sum_{z=1}^{Z} K_z^* r(z) \leq \lambda(n) \quad \forall n = 1..N \tag{5.9}$$

5.1.8 Allocation de capacité sous contraintes matérielles : problème CAH

Le problème que nous proposons de résoudre est le suivant : trouver la meilleure association δ entre les liens et les modèles disponibles dans le but de minimiser le coût réel $\Gamma(\delta)$ du réseau sur une période de temps Ω. Le tableau 5.1 expose le problème CAH dans sa totalité.

Ce problème de minimisation qui est un problème combinatoire peut être résolu de manière exacte avec un algorithme basé sur les techniques de « branch and bound » . Cette solution qui a l'avantage de donner le minimum global mais qui implique des temps de calcul relativement longs sera développée plus loin dans la partie 5.3.

5.2 Configuration optimale de cartes

Le calcul de la configuration de cartes optimale $K^*(\delta, n)$ et de son coût $M^*(\delta, n)$ sur chaque nœud n et pour tout vecteur d'affectation δ peut être fait de manière indépendante. Nous proposons donc de détailler ce calcul dans cette partie.

Soit $G = (g_1, g_2, \ldots, g_T)$ un vecteur de dimension T tel que g_t représente un nombre de liens de modèle t. Le vecteur G est une configuration de liens qui pourra être extraite du vecteur δ pour chaque nœud n :

$$g_t = \sum_{l \in U(n)} \delta_{l,t} \quad \forall t = 1..T \tag{5.10}$$

$$\underset{\delta}{\text{Minimiser}} \quad \Gamma(\delta)$$

Avec :

$$\Gamma(\delta) = \sum_{n=1}^{N} M^*(\delta, n) + \sum_{l=1}^{L} \Gamma(\delta, l)$$

Où $M^*(\delta, n)$ est le coût de la configuration de cartes optimales $K^*(\delta, n) = (k_1^*, k_2^*, \ldots, k_Z^*)$ sur le routeur n, et $\Gamma(\delta, l)$ est le coût du lien l.

Sous les contraintes :

$$\sum_{i=1}^{I} \frac{a_{i,f}}{C_i - Y_i} \leq d^{max}(f) \qquad \forall f = 1..c$$

$$\sum_{t=1}^{T} \delta_{l,t} = 1 \qquad \forall l = 1..L$$

$$\sum_{l \in U(n)} \delta_{l,t} \leq \sum_{z/t(z)=t} k_z^* p(z) \quad \forall n = 1..N$$

$$\sum_{z=1}^{Z} k_z^* r(z) \leq \lambda(n) \qquad \forall n = 1..N$$

TAB. 5.1 – Allocation de capacité sous contraintes matérielles (CAH)

Pour toute configuration de liens G et pour tout routeur n, nous cherchons à résoudre le problème défini dans le tableau 5.2 avec $\lambda = \lambda(n)$, $S = S(n)$, et $K^{init} = K^{init}(n)$ pour trouver la configuration optimale de cartes $K^*(G, n)$ permettant de brancher les liens de G à moindre coût sur le routeur n.

$$K^*(G, \lambda, S, K^{init}) = \underset{K}{\text{Argmin}} \quad M(K, K^{init}) \tag{5.11}$$

Où $M(K, K^{init})$ est le coût de la configuration de cartes $K = (k_1, k_2, \ldots, k_Z)$ sur un routeur dont la configuration de cartes initiale est K^{init} :

$$M(K, K^{init}) = \sum_{z=1}^{Z} \max(0, k_z - k_z^{init})\Gamma_z^C$$

Sous les contraintes matérielles :

$$\sum_{z=1}^{Z} k_z \leq S \tag{5.12}$$

$$g_t \leq \sum_{z/t(z)=t} k_z p(z) \quad \forall t = 1..T \tag{5.13}$$

$$\sum_{z=1}^{Z} k_z r(z) \leq \lambda \tag{5.14}$$

TAB. 5.2 – Configuration de cartes optimale pour une configuration de liens G sur un routeur n

La contrainte (5.12) relate le nombre de slots limité et la contrainte (5.13) traite du nombre nécessaire de ports pour chaque modèle de lien t. Enfin, (5.14) permet de tenir compte de la limitation quand au débit du routeur.

Ce problème peut ne pas avoir de solution. Dans le cas contraire, nous notons $K^*(G, \lambda, S, K^{init})$ la configuration de cartes optimale solution du problème et $M^*(G, \lambda, S, K^{init})$ le coût de cette configuration.

Le principe d'optimalité de Bellman permet de déterminer efficacement la solution optimale de ce problème. En effet, supposons que nous ayons une carte d'un modèle z déjà choisie. Sur cette carte, un maximum de $p(z)$ liens pourront être branchés. Une fois cela effectué, nous devons alors trouver la configuration de cartes optimale associée aux liens de G non branchés.

Nous pouvons donc écrire :

$$M^*(G, \lambda, S, K^{init}) = \min_z \left(\epsilon\Gamma_z^C + M^*(G - z, \lambda - r(z), S - 1, K^{init}) \right) \tag{5.15}$$

algorithm 6 Algorithme donnant la configuration de cartes optimale

procedure BESTCARDCONFIGURATION(Nœud n, Configuration de liens G)
  $min = -1$
  for $z = 1..Z$ **do**
    if $g_{t(z)} = 0$ **then** ▷ Si aucun lien de modèle $t(z)$
      continuer la boucle « for » principale
    end if
    Construire le vecteur $G - z$
    Chercher en mémoire $K^*(G-z) = K^*(G-z, \lambda(n) - r(z), S(n) - 1, K^{init}(n))$
    et $M^*(G-z) = M^*(G-z, \lambda(n) - r(z), S(n) - 1, K^{init}(n))$
    if rien n'est trouvé **then**
      continuer la boucle « for » principale
    else
      if $M^*(G-z) + \epsilon\Gamma_z^C < min$ **then**
        $min = M^*(G-z) + \epsilon\Gamma_z^C$
        $K^*(G, \lambda(n), S(n), K^{init}(n)) = K^*(G-z)$
      end if
    end if
  end for
  if $min > 0$ **then**
    $M^*(G, \lambda(n), S(n), K^{init}(n)) = min$
    Mémoriser K^* et M^* associés à G, $\lambda(n)$, $S(n)$, et $K^{init}(n)$
  else
    La configuration de liens G n'est pas admissible sur le routeur n
  end if
end procedure

Où $G - z = ((g-z)_1, (g-z)_2, \ldots, (g-z)_Z)$ est la configuration de liens G où les liens ayant pu être affectés sur la carte z sont supprimés :

$$(g-z)_i = \begin{cases} \max(0, g_i - p(z)) & \text{Si } i = z \\ g_i & \text{Sinon} \end{cases} \tag{5.16}$$

L'algorithme 6 résume tout ce qui vient d'être dit. Nous supposons que toutes les configurations de cartes optimales associées aux configurations de liens $G - z$ ont déjà été calculées pour le routeur n traité. De plus, si le vecteur $G-z$ est le vecteur nul, alors la configuration de cartes optimale et son coût sont également nuls. Nous notons alors plus simplement $K^*(\delta, n)$ la configuration de cartes optimale pour le routeur n pour le vecteur d'affectation δ et $M^*(\delta, n)$ son coût.

La variable ϵ permet de savoir si le coût de la carte ajoutée doit être pris en compte. Elle dépend de la configuration optimale de cartes $K^*(G - z, \lambda - r(z), S - 1, K^{init}) = (k_1^*, k_2^*, \ldots, k_Z^*)$ et de la

configuration de cartes initiale K^{init} :

$$\epsilon = \begin{cases} 0 & \text{si } k_z^* < k_z^{init} \\ 1 & \text{sinon} \end{cases} \tag{5.17}$$

Pour chaque nœud n, en traitant les configurations de liens G possibles pour toute solution δ dans un ordre croissant du nombre total de liens, il est alors possible de calculer les configurations de cartes optimales associées. Nous construisons alors l'ensemble $\Delta(n)$ qui ne contient que les configurations de liens G admissibles sur le routeur n étant donné que certains des sous problèmes traités plus tôt n'auront pas de solution.

Prenons un exemple avec 2 modèles possibles de bandes passantes respectives 10Mbps et 20 Mbps. Pour chaque modèle, 2 cartes sont utilisables, leurs paramètres sont donnés dans le tableau 5.3. Nous cherchons les configurations de cartes optimales sur un routeur possédant 2 slots ($S(n) = 2$)et une bande passante maximale de 60Mbps ($\lambda(n) = 60$). Nous supposerons de plus qu'aucune carte n'est présente initialement ($K^{init}(n) = (0, 0, 0, 0)$).

Carte (z)	Modèle ($t(z)$)	#ports ($p(z)$)	#débits ($r(z)$)	Coût (Γ_z^C)
1	1	1	10	10
2	1	2	20	15
3	2	1	20	20
4	2	2	40	25

TAB. 5.3 – Exemples de cartes utilisables

Les configurations de liens G sont des vecteurs à deux composantes (une pour le modèle 1 et une autre pour le modèle 2) et les configurations de cartes K ont 4 composantes (une par type de carte). Le tableau 5.4 illustre, dans un des ordres possibles, le calculs des configurations de cartes pour différentes configurations de liens. Le coût des configurations de cartes K proposées est noté M.

5.3 Résolution exacte du problème CAH par des techniques de « branch and bound »

Le fait de trouver le vecteur optimal δ peut être vu comme une suite de choix où l'ordre n'a pas d'impact sur la qualité de la solution trouvée. Le problème CAH peut donc être formulé sous forme de programmation dynamique où L choix doivent être faits parmi T valeurs possibles.

5.3.1 Définition des états

La programmation dynamique construit une suite d'états δ^k convergant vers l'état δ^L solution au problème. A l'étape k, l'état δ^k est défini ainsi :

1. Les liens $l = 1..k$ ont un modèle affecté, les autres non.

G	K	M	Algorithme 6 exécuté pour G	
$(1,0)$	$(1,0,0,0)$	10	Test $z=1 : G-z=(0,0)$ donc $M=\Gamma_1^C+M(0,0)=10$	*
			Test $z=2 : G-z=(0,0)$ donc $M=\Gamma_2^C+M(0,0)=15$	
$(0,1)$	$(0,0,1,0)$	20	Test $z=3 : G-z=(0,0)$ donc $M=\Gamma_3^C+M(0,0)=20$	*
			Test $z=4 : G-z=(0,0)$ donc $M=\Gamma_4^C+M(0,0)=25$	
$(1,1)$	$(1,0,1,0)$	30	Test $z=1 : G-z=(0,1)$ donc $M=\Gamma_1^C+M(0,1)=30$	*
			Test $z=2 : G-z=(0,1)$ donc $M=\Gamma_2^C+M(0,1)=35$	
			Test $z=3 : G-z=(1,0)$ donc $M=\Gamma_3^C+M(1,0)=30$	*
			Test $z=4 : G-z=(1,0)$ donc $M=\Gamma_4^C+M(1,0)=35$	
$(2,0)$	$(0,1,0,0)$	15	Test $z=1 : G-z=(1,0)$ donc $M=\Gamma_1^C+M(1,0)=20$	
			Test $z=2 : G-z=(0,0)$ donc $M=\Gamma_2^C+M(0,0)=15$	*
$(2,1)$	$(0,0,0,0)$	35	Test $z=1 : G-z=(1,1)$ d'où la contrainte (5.12) violée	
			Test $z=2 : G-z=(0,1)$ donc $M=\Gamma_2^C+M(0,1)=35$	*
			Test $z=3 : G-z=(2,0)$ donc $M=\Gamma_3^C+M(2,0)=35$	*
			Test $z=4 : G-z=(2,0)$ donc $M=\Gamma_4^C+M(2,0)=40$	

TAB. 5.4 – Exemple de recherche basée sur le principe d'optimalité

2. Les délais courants pour chaque couple OD f sont connus et notés $d^k(f)$. Le délai d'un lien n'ayant pas de modèle affecté est nul.
3. Les configurations de cartes optimales sont notées $K^*(\delta^k, n)$ pour tout routeur n.

Pour évoluer vers l'état $k+1$, une décision u_{k+1} doit être prise. Cette décision consiste à choisir l'affectation du modèle u_{k+1} au lien $k+1$. La figure 5.2 permet d'illustrer l'évolution dynamique de la solution.

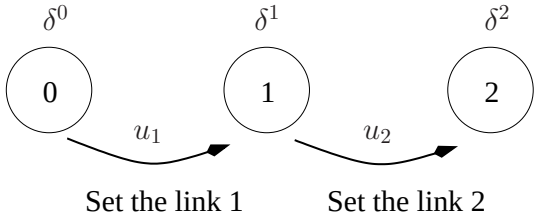


FIG. 5.2 – Evolution dynamique de δ

5.3.2 Admissibilité des décisions

Chaque état δ^k doit être admissible. Or l'affectation d'un modèle à un lien k n'affecte
– que les nœuds extrémités de ce lien (configurations de cartes)
– que les flots passant par ce lien (délais). Nous les notons $F(k) = \{f = 1..c / a_{k,f} \neq 0\}$

Sous l'hypothèse que δ^k est admissible, notons

– $\Delta^k(\delta^k) \subseteq \{1, 2, \ldots, T\}$ l'ensemble des décisions menant à une solution δ^{k+1} admissible par rapport aux contraintes (5.8) et (5.9) :

$$u_{k+1} \in \Delta^k(\delta^k) \quad \Leftrightarrow \quad \exists K^*(\delta^{k+1}, e_i(k+1)) \in \Delta(e_i(k+1)) \text{ pour } i \in \{1, 2\} \tag{5.18}$$

– $\Pi^k(\delta^k) \subseteq \{1, 2, \ldots, T\}$ l'ensemble des décisions menant à une solution admissible pour les contraintes (5.6) :

$$u_{k+1} \in \Pi^k(\delta^k) \quad \Leftrightarrow \quad \forall f \in F(k+1) \quad d^k(f) + \frac{a_{k+1,f}}{C_{u_{k+1}} - Y_{k+1}} \leq d^{max}(f) \tag{5.19}$$

Remarque 5.1 *Pour éviter de surcharger les notations, nous avons fait l'amalgame entre le lien et ses interfaces. Plus précisément, les couples OD qui sont traités lors de l'affectation du lien $k+1$ sont ceux passant sur l'une des deux interfaces de l'ensemble $i(k+1)$ (interfaces portées par le lien $k+1$). La notation utilisée plus tôt $a_{k,f}$ n'a effectivement aucun sens pour un indice k représentant un lien, cet indice doit être celui d'une interface. Nous conserverons cet abus de notation par la suite pour éviter d'alourdir encore plus les formules que nous donnons.*

5.3.3 Dynamique des états

En conclusion, si δ^k est admissible et si la commande u_{k+1} est choisie parmi l'ensemble des commandes admissibles $A^k(\delta^k) = \Delta^k(\delta^k) \cap \Pi^k(\delta^k)$, le nouvel état est admissible et défini par $\delta^{k+1} = f(\delta^k, u_k)$ avec

1. Les variables de décision :

$$\delta^{k+1}_{l,t} = \begin{cases} \delta^k_{l,t} & \forall l = 1..k \quad \forall t = 1..T \\ 1 & \text{si } l = k+1 \text{ et } t = u_{k+1} \\ 0 & \text{sinon} \end{cases} \tag{5.20}$$

2. Le délai de bout en bout :

$$\forall f = 1..c \quad d^{k+1}(f) = d^k(f) + \frac{a_{k+1,f}}{C_{u_{k+1}} - Y_{k+1}} \tag{5.21}$$

3. Les configurations optimales de cartes $K^*(\delta^{k+1}, n)$:
- Elles sont égales à $K^*(n, k)$ si $n \neq e_1(k+1)$ et $n \neq e_2(k+1)$
- Elles doivent être trouvées dans $\Delta(n)$ sinon.

5.3.4 Coûts des décisions

Soit $g^k(\delta^k, u_{k+1})$ le coût de la décision u_{k+1} dans l'état δ^k alors nous pouvons écrire :

$$g^k(\delta^k, u_{k+1}) = \Gamma(k+1, \delta^{k+1}) + \sum_{i=1}^{2} \left[M^*(\delta^{k+1}, e_i(k+1)) - M^*(\delta^k, e_i(k+1)) \right] \tag{5.22}$$

où le premier terme de la somme est le coût d'affectation dû au nouveau lien $k+1$, et le second donne les différences de coût sur les deux nœuds extrémités du lien $k+1$.

5.3.5 Résolution dynamique du problème CAH

A partir de l'ensemble des définitions et équations définies plus tôt dans cette partie, le problème CAH exposé dans la partie 5.1.8 peut donc être reformulé ainsi : trouver la suite optimale de décisions $(u_k)_{1\leq k\leq L}$ minimisant le coût Γ^* où :

$$\Gamma^* = \underset{(u_1,u_2,\ldots,u_L)}{\text{Minimum}} \sum_{k=0}^{L-1} g^k(\delta^k, u_{k+1}) \tag{5.23}$$

avec δ^0 qui est le vecteur nul et en suivant la dynamique $\delta^{k+1} = f(\delta^k, u_{k+1})$.

algorithm 7 Algorithme résolvant le problème CAH

```
1:  procedure B&B(k, δ^k, cost, upper_bound)
2:      if k = L + 1 then
3:          if cost < upper_bound then
4:              upper_bound = cost
5:          end if
6:          return 0
7:      end if
8:      Calculer Λ^k(δ^k) et Π^k(δ^k)
9:      BestCost = 0
10:     for u_{k+1} ∈ Λ^k(δ^k) ∩ Π^k(δ^k) do
11:         costToGo = g_k(δ^k, u_{k+1})
12:         if cost + costToGo < upper_bound then
13:             if cost + costToGo + lower_bound(k, δ^k, u_{k+1}) < upper_bound then
14:                 costToGo = costToGo + B&B(k+1, f(δ^k, u_{k+1}), cost + costToGo, upper_bound)
15:                 if costToGo < bestCost then
16:                     bestCost = costToGo
17:                 end if
18:             end if
19:         end if
20:     end for
21:     Retourner bestCost
22: end procedure
```

L'algorithme 7 basé sur des techniques de « branch and bound » permet une résolution exacte du problème CAH reformulé plus haut. D. Bertsekas propose dans son livre [43] une explication approfondie des résolutions basées sur des graphes dont certaines branches sont explorées et dont d'autres sont coupées. Cela permet en fait de n'explorer qu'une partie de la combinatoire de telle manière que les branches laissées de côté mènent obligatoirement à des solutions plus coûteuses que celles déjà rencontrées.

Ainsi, notre algorithme 7 est récursif et travaille sur les paramètres suivants :

– le lien $k+1$ dont on choisit le modèle,
– l'état δ^k courant et son coût,
– une borne supérieure du coût optimal, et
– une borne inférieure du coût pour l'affectation des liens encore non choisis.

Il explore alors l'ensemble des décisions admissibles dans l'état δ^k, c'est à dire l'ensemble des $u_{k+1} \in A^k(\delta^k)$ (boucle entre les lignes 10 et 20). La valeur retournée par cet algorithme est le coût optimal de la série de décisions (u_{k+1}, u_{k+2}, …, u_L) à partir de l'état courant δ^k.

5.3.6 Bornes supérieures et inférieures

La borne supérieure est égale au coût minimum d'une solution complète δ^L déjà testée (ligne 3 et 4). De ce fait, si le coût de la solution courante additionné au coût de la décision u_{k+1} est supérieur à la borne supérieure, la décision u_{k+1} ne peut pas être optimale (ligne 12).

Il est possible d'améliorer les temps de calculs en chiffrant le coût minimum des décisions restantes (u_{k+2}, u_{k+3}, …, u_L). Chacun de ces liens encore non décidés auront un modèle de capacité supérieure à leur charge. En supposant que les coûts (liens et cartes) augmentent en fonction de la capacité, nous formons la borne inférieure des décisions en associant les liens au modèle de capacité directement supérieure sans tenir compte des contraintes éventuellement violées.

Ainsi si le résultat de l'addition du coût courant, du coût de la décision u_{k+1}, et de la borne inférieure calculées pour (u_{k+2}, u_{k+3}, …, u_L) est supérieur au meilleur coût déjà rencontré, la décision u_{k+1} ne peut pas être optimale et donc elle n'est pas explorée (ligne 13).

5.3.7 Ordre des valeurs testées

La rapidité de notre algorithme dépend grandement de l'ordre dans lequel les valeurs sont testées. Nous affectons les liens dans l'ordre de leurs indices et nous testons les différents modèles également dans l'ordre avec lequel ils sont indicés. Aussi, bien choisir ces indices apparaît comme nécessaire pour permettre une efficacité optimale de notre algorithme.

Les décisions les plus coûteuses doivent être prises en premier pour permettre le maximum de coupes. Aussi, partant du principe que plus la capacité d'un modèle de lien est élevée, plus son coût l'est aussi, les liens les plus chargés devront être traités en premier.

Pour l'ordre dans lequel les décisions sont testées, nous faisons le contraire. En effet, nous voulons obtenir le plus rapidement possible une solution complète de coût raisonnable de manière à couper un maximum ensuite. Donc les décisions les moins coûteuses doivent être testées en premier.

Ainsi, les indices proposés sont les suivants :

$$\forall\, 1 \leq l_1 < l_2 \leq L\,, \quad Y_{l_1} \geq Y_{l_2} \quad \text{et} \quad \forall\, 1 \leq t_1 < t_2 \leq T\,, \quad C_{t_1} \leq C_{t_2}$$

Nous avons testé cet algorithme dans notre outil de simulation NEST ([94]) dans le but principal d'en estimer les temps de calcul. Nous savons que les temps de calcul de ces techniques ne supportent pas la mise à l'échelle. Sur un réseau composé de 50 liens et avec 6 modèles de liens proposés, nous avons stoppé l'algorithme avant sa terminaison après une semaine de calcul. Nous gardons donc cet algorithme comme un outil nous donnant la solution exacte. Mais nous sommes obligés de proposer une heuristique adaptée pour permettre des temps de calcul raisonnables.

5.4 Heuristique basée sur la méthode LDS

Les techniques de résolutions exactes ont le défaut principal d'avoir des temps de calcul trop importants dès que la taille du problème augmente. Pour résoudre ce problème de passage à l'échelle, il est possible d'utiliser des méthodes de résolution non exactes basées principalement sur des heuristiques.

Les techniques de recherche locale étant des techniques relativement efficaces dès lors que le choix du voisinage est bien conçu, nous proposons dans cette partie des les utiliser dans le but de résoudre le problème CAH précédemment développé. Cependant, l'ensemble des contraintes que nous avons à prendre en compte lors de la résolution du problème a tendance à limiter l'efficacité d'une recherche locale basée sur un voisinage de taille faible.

Nous utiliserons alors la recherche à écarts limités (LDS « Limited Discrepancy Search »), qui est proche du fonctionnement du « branch and bound » , pour nous aider à étendre le voisinage quand celui-ci ne propose pas de meilleure solution. Nous proposons dans cette partie une description complète de l'heuristique implémentée.

5.4.1 Définition de la recherche locale

La recherche locale est une technique d'optimisation que nous avons déjà utilisée et présentée dans les chapitres précédents de ce mémoire. Les deux aspects importants à définir sont le voisinage d'une solution et la prise de décision faite suite à l'exploration de celui-ci.

Cette méthode itérative permet une évolution de la solution convergeant toujours vers un optimum local. Ainsi, à l'itération k, à partir du vecteur solution δ^k associant les liens aux modèles, nous construisons un ensemble de $L \times (T-1)$ voisins tels qu'un seul modèle de lien diffère de la solution courante. En notant $V(\delta^k)$ le voisinage ainsi construit, nous pouvons écrire que :

$$\delta(l^{'}, t^{'}) \in V(\delta^k) \quad \Leftrightarrow \quad \exists (l^{'}, t^{'}) \text{ tel que } \begin{cases} \delta^k_{l^{'},t^{'}} = 0 & \\ \delta_{l^{'},t^{'}} = 1 & \\ \delta_{l,t} = \delta^k_{l,t} & \forall\, l \neq l^{'},\ \forall\, t \neq t^{'} \\ \delta_{l^{'},t} = 0 & \forall\, t \neq t^{'} \end{cases} \tag{5.24}$$

L'ensemble des solutions traitées lors des itérations de l'heuristique doivent être admissibles, ainsi les contraintes (5.6), (5.8), et (5.9) doivent être respectées par toutes les solutions δ^k , $k \geq 0$.

Soit $\delta(l^{'}, t^{'})$ le voisin de δ^k où le lien $l^{'}$ est associé au modèle $t^{'}$. Nous notons t^{old} le modèle associé au même lien dans la solution δ^k. En se souvenant que $\Delta(n)$ contient l'ensemble des configurations de liens admissibles pour le nœud n au sens des contraintes (5.8) et (5.9), nous pouvons affirmer que $\delta(l^{'}, t^{'})$ respecte ces mêmes contraintes si et seulement si $\delta(l^{'}, t^{'}) \in (\Delta(e_1(l^{'})) \cap \Delta(e_2(l^{'})))$.

Pour la dernière contrainte (5.6) portant sur les délais, nous devons tout d'abord mettre à jour les délais des flots circulant sur le lien $l^{'}$ par :

$$\forall f \in F(l^{'}) \quad d^{k+1}(f) = d^k(f) - \frac{a_{l^{'},f}}{C_{t^{old}} - Y_{l^{'}}} + \frac{a_{l^{'},f}}{C_{t^{'}} - Y_{l^{'}}} \tag{5.25}$$

La mise à jour des flots est simple : il faut diminuer le délai en retirant l'ancien délai du lien (celui associé au modèle t^{old}) puis l'augmenter en ajoutant le nouveau délai (celui associé au modèle $t^{'}$).

En supposant que la solution δ^0 est admissible au sens des trois contraintes que nous traitons depuis le début, alors la suite de solutions $(\delta^k)_{k\geq 0}$ sera admissible si elle respecte les conditions suivantes

1. $\delta^{k+1} \in V(\delta^k)$. Soient alors $l^{'}$ et $t^{'}$ les paramètres de ce voisin.
2. $\delta^{k+1} \in (\Delta(e_1(l^{'})) \cap \Delta(e_2(l^{'})))$
3. $\forall f \in F(l^{'}), \quad d^{k+1}(f) < d^{max}(f)$ avec $d^{k+1}(f)$ calculé à partir de l'équation (5.25).

La variation de coût $\Delta(\Gamma^k, l^{'}, t^{'})$ entre la solution voisine $\delta(l^{'}, t^{'})$ et la solution courante δ^k peut être calculée de la manière suivante :

$$\Delta(\Gamma^k, l^{'}, t^{'}) = \Delta^{link}(\delta(l^{'}, t^{'}), \delta^k) + \Delta^{node}(\delta(l^{'}, t^{'}), \delta^k) \tag{5.26}$$

$$\Delta^{link}(\delta(l^{'}, t^{'}), \delta^k) = \Gamma(\delta(l^{'}, t^{'}), l^{'}) - \Gamma(\delta^k, l^{'}) \tag{5.27}$$

$$\Delta^{node}(\delta(l^{'}, t^{'}), \delta^k) = \sum_{i=1}^{2} \Big[M^*(\delta(l^{'}, t^{'}), e_i(l^{'})) - M^*(\delta^k, e_i(l^{'})) \Big] \tag{5.28}$$

Avec l'équation (5.27) qui représente la variation de coût due au changement du modèle du lien $l^{'}$, et l'équation (5.28) qui est la variation de coût sur les deux nœuds extrémités.

La définition proposée permet d'obtenir des voisinages de taille faible qui s'explorent relativement vite compte tenu de l'évolution dynamique de coût que nous avons mis en place. Cependant, les contraintes associées au nombre de ports et au nombre de slots ont tendance à limiter énormément le voisinage et à faire converger trop souvent l'algorithme vers des minima locaux. Dès lors deux solutions s'offrent à nous : sortir de ces minima locaux à l'aide de liste tabou par exemple ou par des techniques statistiques d'acceptation ou de rejet des solutions plus coûteuses, ou alors permettre un voisinage plus grand lorsque celui-ci ne permet plus de continuer.

Nous avons opté pour la seconde solution en intégrant des techniques de recherche plus sophistiquées lorsque l'algorithme semble bloqué. Ainsi, nous utiliserons une recherche à écarts limités pour permettre un élargissement de notre voisinage sous certaines conditions.

5.4.2 Recherche à écarts limités

Il est largement reconnu que les méthodes de recherche locale souffrent d'un manque cruel d'efficacité dans certains cas. La rapidité d'exécution de ces techniques suppose principalement des voisinages de taille réduite entraînant parfois la non exploration de solutions plus intéressantes.

Autoriser de temps en temps des voisinages plus larges dans le but de sortir de minima locaux est un bon compromis que beaucoup d'algorithmes d'optimisation cherchent à mettre en place. Harvey et Ginsberg proposent en 1995 dans [35] une solution permettant justement un élargissement contrôlé du voisinage par une technique dite de « recherche à écarts limités » (ou « Limited Discrepancy Search (LDS) » en anglais). L'idée principale est de construire un voisinage tel que chaque voisin possède un nombre limité de différences avec la solution courante.

Ainsi, le voisinage proposé au départ revient à l'utilisation d'un LDS borné à une seule différence. Vu les limitations de cette approche, il est intéressant de limiter le nombre de différences avec une borne B strictement supérieure à 1 lorsqu'un minimum local est trouvé.

La construction de ce voisinage ainsi que son exploration simultanée peut être faite à moindre coût par l'utilisation de l'algorithme 7 défini dans la partie 5.3. Une modification doit tout de même être apportée par l'ajout d'une contrainte supplémentaire lors de la prise de décision : si une décision mène à une solution partielle trop éloignée d'une solution de référence donnée, alors la décision n'est pas explorée et la branche est coupée.

L'heuristique que nous proposons est donc bien une recherche locale mais elle utilise un second voisinage construit et exploré par l'algorithme 7 modifié si aucune meilleure solution n'est trouvée dans le premier voisinage. L'algorithme se termine si aucune meilleure solution n'est trouvée dans le second voisinage également.

En procédant ainsi, nous autorisons l'exploration de plus de solutions dès qu'un minimum local est trouvé cependant, une question reste en suspend : comment fixer la borne B sur le nombre de différences autorisées ? Pour trouver un bon compromis pour cette valeur, il est nécessaire d'étudier l'évolution de la taille du voisinage en fonction de B.

5.4.3 Choix de la valeur limitant les écarts autorisés

Soit $X = (x_1, x_2, \ldots, x_L)$ un vecteur d'entiers compris entre 1 et T. Soit Y un autre vecteur construit selon le même principe. En notant z_i la variable binaire qui vaut 1 si $x_i \neq y_i$ et 0 sinon, nous

cherchons alors le nombre de vecteurs Y tels que $\sum_i z_i \leq B$ pour une valeur B donnée.

Soit $N(k)$ le nombre de vecteurs Y tels que $\sum_i z_i = k$, c'est à dire les vecteurs Y ayant exactement k différences avec X. Si l'on suppose que les positions des différences sont connues, alors on peut écrire simplement que $N(k) = (T-1)^k$ étant donné que chaque différence peut prendre $T-1$ valeurs.

Soit $p(k)$ le nombre de choix possibles pour les k positions des différences parmi les L possibilités. On reconnaît alors la définition d'une combinaison donc :

$$p(k) = C_L^k = \frac{L!}{k!(L-k)!} \tag{5.29}$$

Dès lors, le nombre $Z_{L,T}(B)$ de vecteurs Y avec un maximum de B différences avec X peut être obtenu comme suit :

$$Z_{L,T}(B) = \sum_{k=0}^{B} N(k)p(k) = \sum_{k=0}^{B} \frac{L!}{k!(L-k)!}(T-1)^k \tag{5.30}$$

Il est évident que le nombre de solutions au problème CAH est $Z_{L,T}(L)$. Nous pouvons donc nous servir de ces résultats de combinatoire pour évaluer la quantité de solutions explorées suite au choix de la valeur de B. Pratiquement, si l'on suppose $T = 10$ modèles de liens et $L = 50$ liens alors on obtient les valeurs suivantes :

- Si $B = 1$: $Z_{50,10}(1) = 451$ soit $4.5e^{-46}$% de l'ensemble des solutions
- Si $B = 2$: $Z_{50,10}(2) = 99676$ soit $9.9e^{-44}$% de l'ensemble des solutions
- Si $B = 5$: $Z_{50,10}(5) = 1.2e^{11}$ soit $1.2e^{-37}$% de l'ensemble des solutions
- Si $B = 10$: $Z_{50,10}(10) = 3.6e^{19}$ soit $3.6e^{-29}$% de l'ensemble des solutions
- Si $B = 20$: $Z_{50,10}(20) = 6.1e^{32}$ soit $6.1e^{-16}$% de l'ensemble des solutions

Nous proposons alors de choisir B de telle manière

- qu'au moins 1% du nombre total de solutions soit exploré par l'algorithme, et
- qu'au plus 500000 voisins soient explorés.

5.4.4 Prise en compte des pannes

Le problème posé peut être étendu à l'ensemble des cas de panne possibles sur le réseau. Nous pouvons ainsi formuler le problème de dimensionnement sous des contraintes

- matérielles (cf. équation 5.9 et 5.8),
- de délai dans le cas nominal (cf. équation 5.6), et
- de performances garanties en cas de panne.

Pour traiter la nouvelle contrainte, étant donné que le routage est connu est qu'aucun lien ne sera ajouté ou supprimé, nous savons que la pire charge sur chaque lien est le maximum sur tous les états possibles du réseau. Nous considérerons donc que la charge Y_l associée au lien l est la pire charge que ce lien pourrait supporter. En divisant cette valeur par un facteur U (par exemple 70%), nous pourrons

garantir que chaque lien est dimensionné de telle manière que son utilisation ne dépasse jamais U quel que soit l'état du réseau en ajoutant une contrainte sur la capacité des liens :

$$\sum_{t=1}^{T} \delta_{l,t} C_t < \frac{Y_l}{U} \quad \forall l = 1..L \tag{5.31}$$

Le respect de la contrainte 5.31 où Y_l est la charge maximale que le lien peut supporter permet de proposer des solutions robustes où l'utilisation des liens ne dépasse pas U%.

5.5 Tests et résultats

Nous avons intégré l'algorithme défini dans les parties précédentes dans l'outil NEST. Nous avons également développé la résolution exacte du problème à l'aide de l'algorithme 7. Nous proposons donc dans cette partie de décrire le protocole de test utilisé ainsi que les résultats obtenus. Nous donnerons également un certain nombre de commentaires sur ces résultats.

Pour tous les tests que nous avons effectués, nous nous sommes limités à un ensemble de modèles de lien restreint. Chaque lien peut être branché sur 3 cartes qui diffèrent par leur coût et leur nombre de ports. Cette restriction est visible dans le tableau 5.5 (pour les liens) et 5.6 (pour les cartes). De plus, les routeurs utilisés ont pour caractéristiques : un nombre de slots limité à 8 et une capacité de traitement maximale égale à 1e+08 Kbps.

Modèle de lien	**Capacité (Kbps)**	**Coût d' installation** Γ_t^L	**Distance (Km)**	**Coût fixe mensuel** Γ_t^F	**Coût kilométrique mensuel** Γ_t^D
OC-1	50725	4500€	1-10	2100€/mois	0€/Km/mois
			>10	2000€/mois	25€/Km/mois
OC-3	152174	4500€	1-10	2600€/mois	0€/Km/mois
			>10	2500€/mois	25€/Km/mois
OC-12	608698	4500€	1-10	5000€/mois	0€/Km/mois
			>10	4700€/mois	25€/Km/mois

TAB. 5.5 – Modèles de lien, capacités, et coûts utilisés pour nos tests

Le réseau décrit par la figure 5.3 est relativement simple. Avec seulement 7 nœuds et 7 liens, il est facile et rapide de trouver la solution optimale à partir de réflexions simples. La figure 5.4 montre les capacités et les modèles associés à la topologie initiale. La même figure donne également les valeurs des trafics entre les sources et les destinations des liens, ainsi que les utilisations obtenues (le routage est ici extrêmement simple).

Sur cet exemple, les délais autorisés sont largement dépassés étant données les interfaces chargées à plus de 100%. De manière générale, nous ne parlerons pas des contraintes de délais mais des utilisations

Cartes	Modèle	#Ports	Coût (K€)
1	OC-1	1	30
2	OC-1	2	40
3	OC-1	4	60
4	OC-3	1	40
5	OC-3	2	60
6	OC-3	4	80
7	OC-12	1	60
8	OC-12	2	80
9	OC-12	4	120

TAB. 5.6 – Cartes utilisées lors des tests

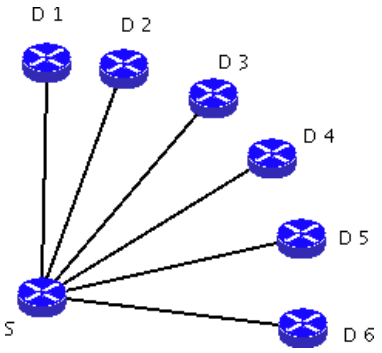


FIG. 5.3 – Test 1 : Topologie

	Modèle de lien	Demandes (Mbps)	Utilisation (%)
S → D 1	OC-1	20	39.4
S → D 2	OC-1	100	197.1
S → D 3	OC-1	100	197.1
S → D 4	OC-1	100	197.1
S → D 5	OC-1	100	197.1
S → D 6	OC-1	100	197.1

FIG. 5.4 – Test 1 : Valeurs initiales

des interfaces. Nous supposerons que dans les exemples traités, les délais sont respectés si les interfaces sont chargées à moins de 80%.

La solution optimale de ce problème est classique car les contraintes ne sont pas violées par la solution consistant à choisir un modèle de capacité directement supérieure à la charge. Ainsi, il faut changer le modèle des liens chargés à plus de 197% en leur affectant un modèle OC-3 permettant de ramener l'utilisation à 65.7%. En termes de carte, cela revient à ajouter une carte de modèle OC-3 possédant 1 port sur l'ensemble des routeurs destinations sauf D 1. La carte initialement présente sur ces routeurs ne sera plus utilisée, elle est donc enlevée. De plus, une carte OC-3 à 4 ports et une carte OC-3 à 1 port sont ajoutées sur le nœud S, la carte OC-1 à 4 ports initialement présente étant retirée.

L'heuristique proposée trouve rapidement cette solution (moins de 20 secondes). La figure 5.6 montrent la solution proposée telle qu'elle est présentée dans NEST. Les résultats généraux de l'étude sont illustrés sur la figure 5.5 avec entre autre le coût réel de la topologie modifiée pour une durée de 3 mois. Les figures 5.6(a) et 5.6(b) montrent respectivement les modifications proposées sur les liens et sur les cartes.

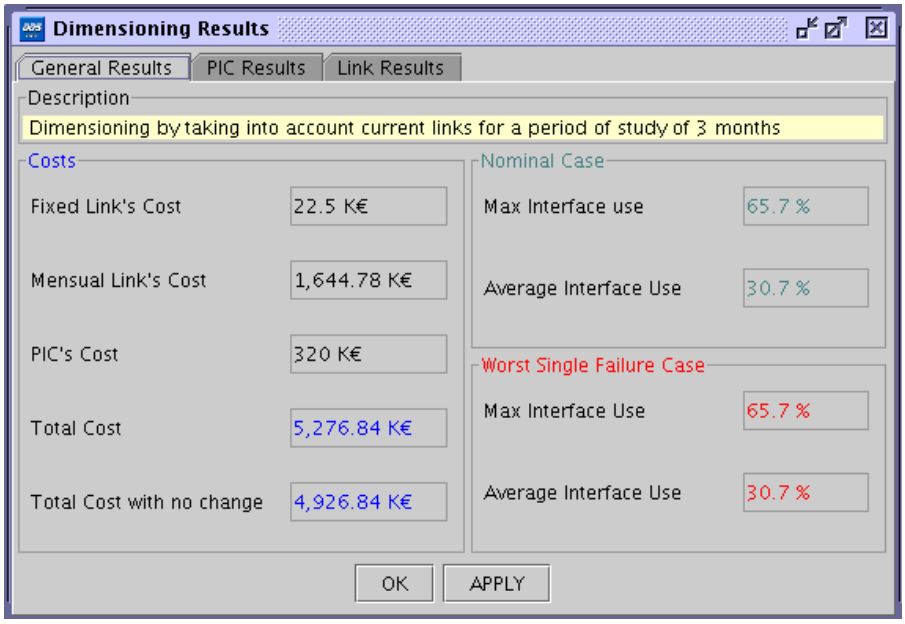
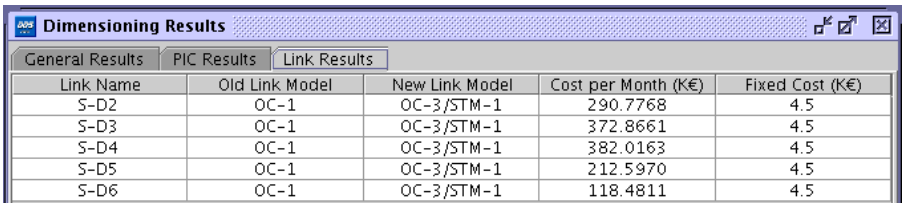
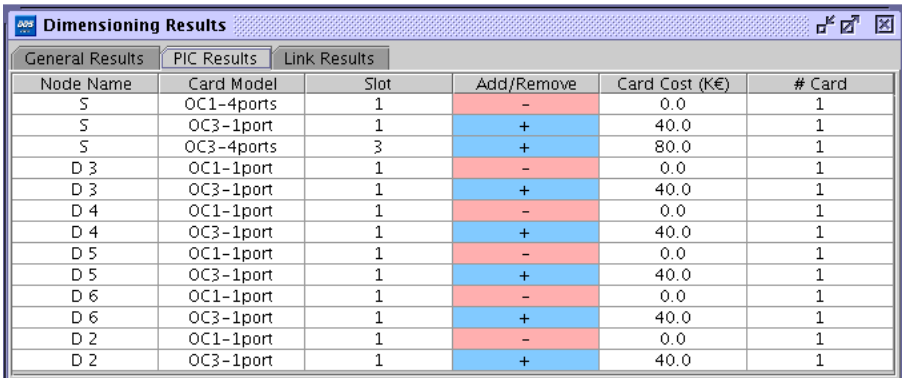


FIG. 5.5 – Résultats généraux sur le premier test



Link Name	Old Link Model	New Link Model	Cost per Month (K€)	Fixed Cost (K€)
S-D2	OC-1	OC-3/STM-1	290.7768	4.5
S-D3	OC-1	OC-3/STM-1	372.8661	4.5
S-D4	OC-1	OC-3/STM-1	382.0163	4.5
S-D5	OC-1	OC-3/STM-1	212.5970	4.5
S-D6	OC-1	OC-3/STM-1	118.4811	4.5

(a) Résultats sur les liens



Node Name	Card Model	Slot	Add/Remove	Card Cost (K€)	# Card
S	OC1-4ports	1	-	0.0	1
S	OC3-1port	1	+	40.0	1
S	OC3-4ports	3	+	80.0	1
D 3	OC1-1port	1	-	0.0	1
D 3	OC3-1port	1	+	40.0	1
D 4	OC1-1port	1	-	0.0	1
D 4	OC3-1port	1	+	40.0	1
D 5	OC1-1port	1	-	0.0	1
D 5	OC3-1port	1	+	40.0	1
D 6	OC1-1port	1	-	0.0	1
D 6	OC3-1port	1	+	40.0	1
D 2	OC1-1port	1	-	0.0	1
D 2	OC3-1port	1	+	40.0	1

(b) Résultats sur les cartes

FIG. 5.6 – Résultat de dimensionnement sur le premier test

La solution optimale du problème précédent est donc classique car elle consiste à associer chaque lien avec le modèle de capacité directement supérieure à la charge du lien. Cependant, les contraintes matérielles dont nous tenons compte permettent

– d'éviter de telles solutions si celles-ci ne peuvent pas être mise en place par l'opérateur, et
– de trouver des solutions moins coûteuses de part les équipements déjà en place.

Le premier point peut être illustrer facilement à partir de la topologie précédente où les caractéristiques des routeurs sont changées : nous limitons le nombre de slots disponibles à 2 au lieu de 8. Dès lors, la solution trouvée précédemment ne convient plus car elle suppose 3 cartes sur S. Dans ce cas là, l'algorithme proposé converge vers une solution ne nécessitant que 2 cartes sur S : tous les liens sont affectés au modèle OC-3. Ainsi, une carte OC-3 à 4 ports et une carte OC-3 à 2 ports suffisent pour brancher les 6 liens sur S.

Pour illustrer le second point, considérons la topologie décrite sur la figure 5.7. Les valeurs des modèles et des charges de chaque lien sont données dans la figure 5.8.

Dans cet exemple, le lien B $\rightarrow$ E est associé à un modèle de trop faible capacité. De part la charge sur ce lien (100 Mbps), le modèle qui semble le plus adapté est OC-3 (capacité égale à 152 Mbps) mais il n'est pas choisi par notre algorithme. En effet, des cartes OC-12 à 4 ports sont présentes sur les nœuds B et E mais seulement 3 liens sont branchés dessus. La présence de ports OC-12 non connectés font que l'association du modèle OC-12 sur le lien B $\rightarrow$ E coûte moins cher.

Si nous reprenons les définitions des coûts donnés plus haut, sachant que le lien B $\rightarrow$ E mesure 120Km, nous pouvons évaluer le coût d'affectation de ce lien pour les deux solutions étudiées ci-dessus et pour une durée Ω de 6 mois

Modèle OC-3 sur le lien B $\rightarrow$ E

1. Coût du lien : $4500 + (2500 + 120 * 25) * 6 = 37500$€
2. Coût des cartes (une sur B et une sur E) : $2 * 40000 = 80000$€
3. Coût total de 117.5K€

Modèle OC-12 sur le lien B $\rightarrow$ E

1. Coût du lien : $4500 + (4700 + 120 * 25) * 6 = 50700$€
2. Coût des cartes (aucune carte nécessaire) : 0€
3. Coût total de 50.7K€

La solution proposée amène donc un gain de 66.8K€ soit une diminution du coût de plus de 56%. Cela montre l'intérêt économique majeur de l'intégration des contraintes matérielles dans le dimensionnement de topologie.

Le tableau 5.7 propose d'autres résultats permettant une comparaison entre le coût C_{sup} obtenu par un dimensionnement fait à l'aide du modèle de capacité directement supérieure permettant de respecter toutes les contraintes et le coût C_{opt} donné par notre algorithme. Ce tableau montre sur trois exemples différents les économies conséquentes qui peuvent être faites si l'on tient compte des cartes existantes.

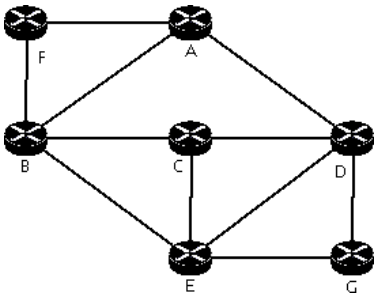


FIG. 5.7 – Test 2 : Topologie

	Modèle de lien	Demandes des trafics (Kbps)	Utilisation (%)
B → E	OC-1	100000	197.1
B → C	OC-12	400000	65.7
B → F	OC-12	400000	65.7
C → D	OC-12	400000	65.7
A → D	OC-12	300000	49.3
B → A	OC-12	300000	49.3
E → C	OC-12	400000	65.7
E → G	OC-12	300000	49.3

FIG. 5.8 – Test 2 : Valeurs initiales

Tests	# Nœuds	# Liens	C_{opt} (K€)	C_{sup} (K€)	Gain (K€)	Gain (%)
1	22	36	1689	2015	326	**16.2**
2	36	56	3223	3404	181	**5.3**
3	84	145	4487	5977	1490	**24.9**

TAB. 5.7 – Economies réalisées sur différentes topologies de test

5.6 Conclusion

Le changement de capacité d'un lien a longtemps été vu comme le remplacement de ce lien de manière physique, entraînant des coûts importants tant au niveau du matériel installé que de son installation (tranchée etc...). En conséquence, de nombreux travaux portant sur le dimensionnement de topologie négligeaient le coût des autres équipements. Ce n'est plus le cas aujourd'hui, les capacités physiques des liaisons installées étant largement suffisantes, et la majorité des liens étant loués, cette approximation ne tient plus. Le coût des autres équipements et principalement celui des cartes permettant le branchement des liens sur les routeurs ne peut plus être négligé.

Nous proposons donc dans ce chapitre une nouvelle approche du dimensionnement basée sur

1. une définition réaliste des coûts menant à la minimisation du coût réel de la topologie proposée,
2. un ensemble de contraintes matérielles permettant de prendre en compte l'ajout de nouvelles cartes quand cela est nécessaire et de garantir la faisabilité matérielle de la solution proposée,
3. des contraintes de QoS sur les flots (délai moyen des paquets de bout en bout) permettant de proposer une solution respectant les SLA associés aux trafics, et
4. une prise en compte des performances en cas de panne.

Le problème ainsi défini peut être résolu par des techniques de « branch and bound » permettant d'explorer la combinatoire. Cependant de tels algorithmes ne passant pas à l'échelle, nous proposons une heuristique efficace pour résoudre ce problème. Les principaux atouts de cette heuristique sont

1. le calcul a priori des configurations optimales de cartes sur les routeurs pour toutes solutions traitées par l'heuristique,

2. l'utilisation d'un premier voisinage simple, de taille réduite et contenant souvent une meilleure solution, et
3. l'utilisation d'un second voisinage plus complexe , de taille plus grande, utilisé dans le cas où le premier voisinage mène à un minimum local.

Les perspectives de recherche associées à la nouvelle modélisation proposée portent principalement sur

- l'amélioration de la convergence et de la rapidité de l'algorithme proposé,
- l'intégration de nouvelles variables portant sur les modèles des routeurs, et
- l'ajout éventuel d'équipements (liens, routeurs) quand la topologie proposée ne peut être améliorée dans l'état.

CHAPITRE 6

Modélisation et simulation dans NEST

L'association nationale de la recherche technique (ANRT) associée à la start-up QoS Design a subventionné les travaux de thèse que j'ai pu mener depuis plus de trois ans au sein du Laboratoire d'Architecture et d'Analyse des Systèmes (LAAS). Les contributions apportées durant mon doctorat sont donc doubles, d'un côté des travaux de recherche effectués en collaboration avec le LAAS qui m'a accueilli, et de l'autre des contributions en terme d'analyse et de développement pour QoS Design.

Les travaux de recherche et les résultats obtenus ont été présentés dans les chapitres précédents. Il faut bien noter que l'ensemble de ces travaux ont été intégrés dans la suite logicielle NEST commercialisée par QoS Design. Cependant, de nombreuses autres contributions ont été apportées à ce logiciel et même si le contenu scientifique est plus pauvre, il me parait important de les préciser. Nous les exposerons donc dans la suite de ce chapitre.

Avant de détailler les principaux aspects auxquels j'ai pu contribuer, revenons sur NEST : Network Engineering and Simulation Tool. Cet outil est une suite logicielle dédiée aux opérateurs et aux fournisseurs d'accès à Internet (FAI). Une interface conviviale permet à l'utilisateur de « dessiner » la topologie de son réseau. Une bibliothèque extrêmement riche de modèles de routeurs et de modèles de liens permet une configuration très précise de la topologie. De plus, la plupart des protocoles de routage existants sont implémentés : OSPF, ISIS, EIGRP, RIP, BGP ainsi que le routage statique. L'utilisateur peut donc facilement configurer l'ensemble des équipements de son réseau (son routage par exemple).

NEST possède également une bibliothèque très riche de modèles de services pour les trafics permettant une définition très poussée des différents aspects statistiques des sources considérées. L'utilisateur peut donc, à partir de ces services, définir une matrice de trafics que le réseau doit supporter.

Enfin, il est possible de configurer un cœur de réseau IP/MPLS avec NEST en définissant des LSP avec leurs FEC associées. Ces LSP ainsi mis en place sur le réseau sont complètement intégrés au routage et à la propagation des trafics. De même, l'utilisateur peut définir des VPN de niveau 2 ou de niveau 3. Le protocole MPBGP vient d'être intégré et il permet donc de mieux préciser la communication des

différents VPN entre eux.

Une fois l'ensemble du réseau configuré, l'utilisateur a accès à un ensemble d'outils d'analyse

1. Routage et propagation des demandes offertes permettant de repérer rapidement les points de congestion éventuels et d'observer l'utilisation des ressources sur le réseau.
2. Etude de la résilience du réseau.
3. Simulation événementielle de la propagation des trafics.
4. Simulation analytique de la propagation des trafics.
5. Simulation hybride : elle permet de préciser les ensembles du réseau qui seront simulés de manière événementielle et ceux qui seront simulés de manière analytique. Ce nouveau procédé de simulation est protégé par un brevet du LAAS et est le produit de recherches intensives effectuées par l'équipe du LAAS avec laquelle j'ai pu travailler.

L'utilisateur peut également exécuter un ensemble d'algorithmes d'ingénierie portant sur

1. l'optimisation des métriques (cf. chapitre 3),
2. le placement optimal des LSP (cf. chapitre 4),
3. l'estimation de matrice de trafics à partir de données d'activité et de population,
4. et le design de topologie (nouveauté en cours d'implémentation).

6.1 Amélioration du routage

La majeure partie de mes travaux dans le cadre du logiciel porte sur les aspects de routage et de propagation. En effet, je suis devenu en quelque sorte le « Mr routage » de l'équipe. De part ce statut, j'ai pu améliorer, maintenir, et ajouter de nombreux aspects du routage des flots dans un réseau IP/MPLS.

6.1.1 Intégration des trafics multicasts

Nous avons tout d'abord travaillé sur l'intégration des trafics multicasts dans le routage et la propagation associé à NEST. En effet, il existe plusieurs algorithmes de routage multicast permettant la mise en place de l'arbre de propagation. De notre côté, suite aux recherches effectuées et de part les besoins exprimés des opérateurs intéressés, nous avons opté pour une intégration du protocole PIM-SSM ou « Protocol Independant Multicast Source Specific Multicast » . Ce protocole a l'avantage d'être totalement indépendant des protocoles de routage IGP utilisés. Il s'appuie sur les routes précalculées en unicast et permet la mise en place d'un arbre de propagation pour chaque source identifiée et associée à un groupe multicast.

Pour permettre une telle intégration, nous avons donc proposé une interface permettant à l'utilisateur d'entrer les spécifications d'un groupe multicast : l'ensemble de ces sources et l'ensemble de ces receveurs. Une fois de tels groupes configurés, l'utilisateur peut spécifier des trafics à destination des

ces groupes, la seule contrainte étant que la source de ces trafics fasse partie des sources associées au groupe destinataire.

Le module de calcul effectuant le routage et la propagation des flots a donc été adapté pour la lecture de tels trafics. Ensuite, un routage multicast est effectué pour l'ensemble des couples sources / groupe multicast identifiés. Ce routage consiste en la mise en place d'un arbre de propagation entre la source et les receveurs du groupe basé sur le principe du « reverse path forwarding » . C'est à dire que le chemin (la partie de l'arbre) emprunté entre la source et chaque receveur est le chemin inverse du plus court chemin entre le receveur et la source (plus court au sens de l'IGP présent).

Le module de calcul du routage permet donc la mise en place de table de routage multicast où un ensemble d'interfaces de sortie est associé à une destination multicast et ce localement sur chaque nœud. Lors de la propagation, le trafic est dupliqué sur chacune des entrées de la table de routage correspondant à la destination. Nous donnons un exemple d'arbre de propagation multicast dans la figure 6.1. On peut voir en flèche verte les différentes interfaces à emprunter pour aller de la source A à l'ensemble des destinations (B, C, D, E, F, et G) sachant que les métriques OSPF de ce réseau sont toutes égales à 1.

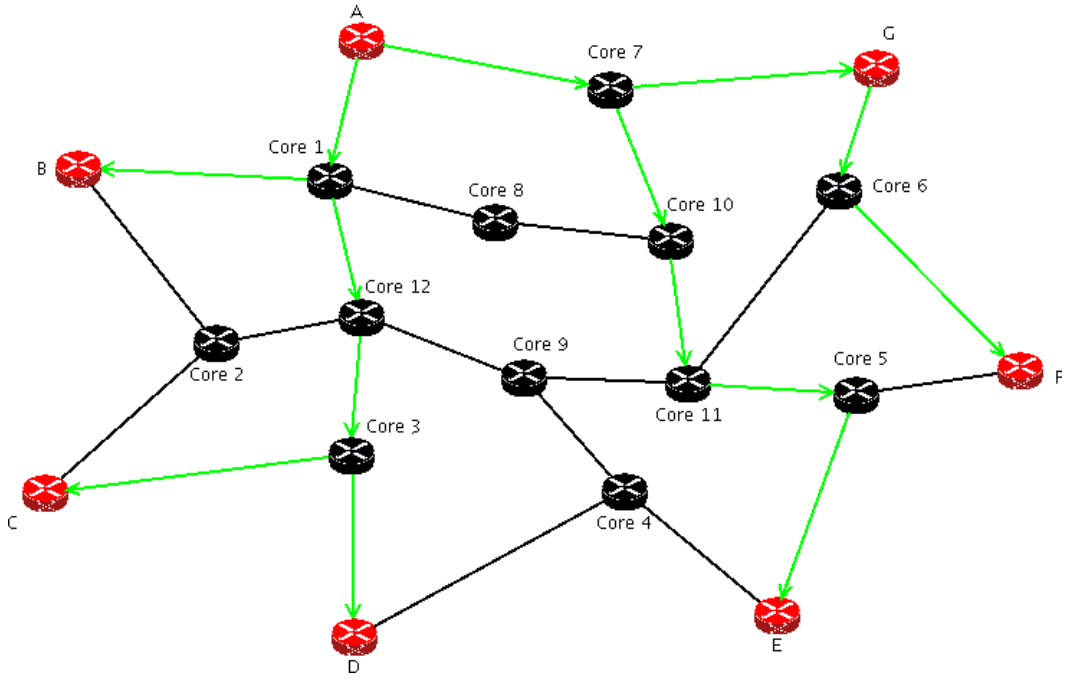


FIG. 6.1 – Exemple d'arbre de propagation multicast issu de A

Enfin, nous avons également fait évoluer MPLS et la gestion des LSP dans NEST pour permettre une utilisation des LSP par les trafics multicasts. Ainsi, chaque branche de l'arbre de propagation peut être associée à un LSP.

6.1.2 Intégration des trafics radios 2.5/3G

Dans le cadre d'un projet de recherche mêlant laboratoires et entreprises, les aspects radios 2.5/3G ont été traités et intégrés dans NEST. De nouvelles sources de trafics ont été ajoutées, de nouveaux équipements ont été créés et la gestion de la propagation des trafics radios dans les cœurs de réseaux IP/MPLS a donc dû être également implémentée.

Les flots radios issus de terminaux 2.5/3G sont transmis depuis leur source vers des points d'entrée du réseau IP/MPLS associé : les SGSN (pour « Serving GPRS Support Node »). Les SGSN se chargent donc de transformer les données venant de la partie radio du réseau en données IP transmissibles sur le cœur de réseau.

Cependant, le routage radio est très particulier. Cela est dû principalement à la nécessité de contrôle d'accès et de facturation qui se cache derrière toute utilisation d'un service 2.5/3G à partir de son téléphone portable. Ainsi, le SGSN doit établir un tunnel GTP avec un GGSN (pour « Gateway GPRS Support Node ») tout en maintenant l'état de cette connexion : le PDP context. Une fois ce tunnel en place, le SGSN émet les données IP vers le GGSN qui se charge de son côté de transmettre les données vers le serveur applicatif recherché.

La propagation des données entre SGSN et GGSN puis entre GGSN et serveur se fait bien sûr de manière classique c'est à dire en IP voire en IP/MPLS le plus souvent. Cependant, les choix d'un GGSN et d'un serveur dépendent de la configuration des DNS (« Domain Name Server »).

Suivant le service qu'un utilisateur cherche à contacter, le SGSN recevra une liste de GGSN suite à l'interrogation du DNS auquel il est associé. De la même manière, le GGSN interroge le DNS pour connaître la liste des serveurs vers lesquels propager les données pour le service demandé.

La configuration d'un trafic radio 2.5/3G consiste en une définition de sa source (un SGSN) et de son service sans oublier bien sûr de lui donner un nombre de connexions ou une bande passante moyenne. Dès lors, il était impératif d'implémenter les mécanismes de routage radio dans NEST pour pouvoir résoudre la question suivante : comment propager un trafic n'ayant pas de destination ?

La figure 6.2 montre le chemin en vert emprunté par un trafic radio issu de « portable » qui est connecté à « SGSN » et à destination du service « Wap 3G ». Le DNS configuré précise l'utilisation des éléments « GGSN 1 » et « GGSN 2 » pour le routage des flots de ce service. De plus, le DNS précise que les serveurs « Server 1 » et « Server 2 » sont à contacter pour pouvoir établir une connexion pour ce service. Nous partageons donc le flot sur les différents éléments précisés par le DNS et ceci avec les poids précisés par le DNS. Les poids étant identiques ici, le partage sera équitable.

Une fois cette nouvelle évolution apportée au module de propagation, il nous a fallu également prendre en compte les mécanismes de redirection souvent utilisés par les opérateurs. Ainsi, les listes retournées par les DNS comportent le plus souvent plus d'un élément. Cette multiplicité doit être associée à un mécanisme de choix (typiquement du round robin) pour permettre la communication avec un seul élément de la liste. Mais un GGSN peut refuser l'établissement d'un nouveau PDP Context. Il

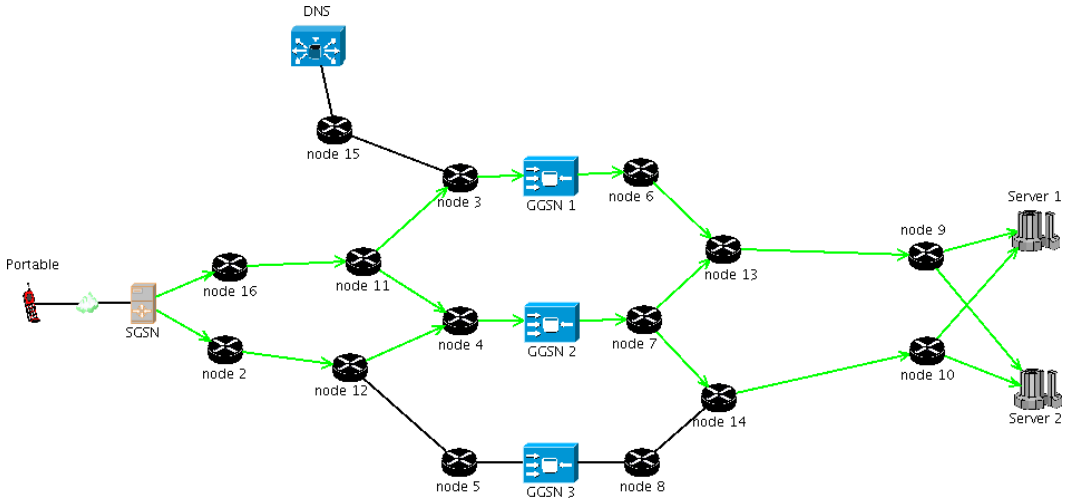


FIG. 6.2 – Exemple de décision de routage radio sur un réseau 2.5/3G

est en effet bridé pour lui assurer certaines performances. Aussi, si un GGSN refuse une connexion, cette connexion est débordée sur le second de la liste et ainsi de suite jusqu'à ce que les possibilités soient épuisées (et la connexion alors perdue) ou qu'un GGSN accepte la connexion.

Nous reviendrons sur cet aspect de la propagation des flots radios dans la partie 6.2 où nous proposons une optimisation de la configuration des DNS tenant compte des aspects précédemment cités.

6.1.3 Intégration des VPN L3 MPBGP

Le dernier point important que nous avons ajouté dans le module de routage du logiciel est une meilleure gestion des réseaux privés virtuels (VPN). En effet, la notion de service offert aux clients au travers de VPN dédiés est de plus en plus fréquente sur les réseaux d'opérateur. Ainsi, tous les sous réseaux d'un client peuvent communiquer entre eux au travers d'un VPN dédié. Cela permet d'abstraire complètement le cœur de réseau traversé en une connexion direct entre les différents LAN (Local Area Network) ou sous-réseaux du client.

Suivant que le client se connecte au cœur de réseau au niveau de la couche 3 (par des routeurs IP) ou au niveau de la couche 2 (par des switch Ethernet ou ATM), nous parlerons de VPN L3 ou de VPN L2. Les VPN L3 ont l'inconvénient majeur suivant : le système d'adressage du client doit être connu de l'opérateur. Une extension au protocole BGP, MPBGP, permet justement de contourner cet inconvénient. L'utilisation plus approfondie du paramètre de communauté de BGP permet en effet une distinction d'adresses identiques suivant le VPN auxquelles elles appartiennent. C'est pourquoi il devient de plus en plus répandu. Pour cette raison, il devenait indispensable de proposer ce protocole dans NEST.

Nous n'avons cependant pas mis de côté les autres types de VPN. Mais leurs configurations sont relativement plus simples car elles sont basées sur un maillage complet des points d'accès au VPN.

Ainsi, les VPN L2 gérés par VPLS par exemple, ou les VPN L3 point à point classiques proposent des tunnels entre chaque couple de « Service Access Point (SAP) » par lesquels le client émet ou reçoit du trafic. Ces VPN ne posent donc pas de problème majeur quant à leurs configurations dans NEST étant donné que le routage entre SAP est direct.

MPBGP est une extension au protocole BGP qui permet le calcul des routes entre les hôtes communicants derrière chaque SAP pour un VPN donné. De plus, MPBGP permet également de définir certaines relations entre des VPN différents permettant ainsi la communication de plusieurs VPN les uns avec les autres. Pour se faire, une nouvelle table de routage est activée sur les routeurs faisant tourner MPBGP : la « Virtual and Forwarding Routing table (VRF) » qui permet d'installer les routes vers des destinations non visibles en IP.

Chaque VRF est associée à un ensemble d'interfaces MPBGP connectant un LAN (ou sous réseau) au routeur. Cela lui permet de connaître les adresses joignables directement qui seront les adresses apprises aux autres routeurs MPBGP. Une VRF est également configurée avec un paramètre d'import et un paramètre d'export. Ainsi, toutes les destinations joignables directement seront émises vers les autres routeurs MPBGP en étant associées au paramètre d'export de la VRF qui les émet. Une VRF apprenant une adresse vérifie tout d'abord que son paramètre d'import correspond au paramètre d'export associé. Si oui, la route est apprise et stockée dans la VRF, sinon la route est rejetée. Les VRF sont ainsi construites et ce mode de fonctionnement permet d'obtenir des routages différents des classiques « Full Mesh » que l'on connaissait.

Nous avons donc intégré dans NEST :

1. une définition d'un client et des VPN associés,
2. une configuration et une gestion des interfaces virtuelles MPBGP,
3. une configuration et une gestion des VRF sur chaque nœud,
4. une exécution du protocole MPBGP dans le module de calcul du routage,
5. et une propagation intégrant le routage des VRF dans le module de propagation.

6.2 Optimisation des poids DNS dans le routage radio

Comme nous l'avons expliqué dans la partie 6.1.2 de ce chapitre, le routage des flots radios 2.5/3G s'appuie sur la configuration du DNS associé au SGSN émetteur. Une fois le routage « radio » établit, le routage IP prend le relais pour permettre une propagation du trafic vers le(s) GGSN(s) et le(s) serveur(s) retournés par le DNS. L'utilisation des ressources associée à un tel paradigme de routage et de propagation est donc complètement liée à la configuration du DNS. Dans le cadre de nos travaux, nous avons proposé d'étudier cette relation de manière à pouvoir mettre en œuvre une configuration optimale des DNS.

6.2.1 Modélisation de la topologie radio

Les charges sur le réseau étant à la base de tout calcul de performance, nous proposons dans cette partie d'établir leurs relations avec la configuration du DNS. Pour se faire, nous avons considéré un seul service (les poids DNS étant dépendant du service, l'extension à plusieurs services se fait de manière triviale), et nous avons supposé que chaque SGSN et chaque GGSN est lié au même DNS (le cas multi-DNS peut être étendu de manière simple). Dès lors, nous utiliserons les notations suivantes :

- le poids DNS associé au GGSN g sera noté α_g,
- le poids DNS associé au serveur s sera noté β_s.
- Un SGSN source s sera caractérisé par :
  - F_s son débit montant,
  - B_s son débit descendant, et
  - N_s son nombre de connexions émises.
- Un GGSN g (point de transit) sera caractérisé par :
  - TF_g son débit montant vers les serveurs,
  - TB_g son débit descendant vers les SGSN, et
  - TN_g son nombre de connexions ouvertes.

Enfin, nous modélisons le réseau IP par un graphe G avec son ensemble de nœuds N, son ensemble de liens L, et sa matrice de routage A. Ces différentes notations sont illustrées sur la figure 6.3. Pour plus de lisibilité, nous noterons s un SGSN, g un GGNS et sv un serveur dans les formulations qui suivent.

Nous considérerons des agrégations de flots ayant la même source (toujours pour le service considéré) et un flot sera donc caractérisé par une source (le SGSN émettant ce flot), un nombre de connexions (nombre de PDP à ouvrir pour ce flot agrégé), un débit montant et un débit descendant. Ce flot peut alors être découpé en quatre parties principales comme illustré sur la figure 6.3, chaque partie correspondant à un certain nombre d'Origine/Destination (OD). Nous pouvons alors utiliser l'équation de propagation pour chaque OD et sommer le résultat pour chaque lien. Cela nous donne pour un lien l :

$$\begin{aligned} Y_l &= \sum_s \sum_g (F_s \alpha_g A_{l,s\to g} + B_s \alpha_g A_{l,g\to s}) \\ &+ \sum_g \sum_{sv} (TF_g \beta_{sv} A_{l,g\to sv} + TB_g \beta_{sv} A_{l,sv\to g}) \end{aligned} \tag{6.1}$$

Le terme $A_{l,i\to j}$ correspond à la proportion du trafic associé au couple (i,j) qui est routée sur le lien l. La première somme contient les trafics entre SGSN et GGSN et la seconde contient les trafics entre GGSN et serveur. A cette première équation doit être ajoutée l'expression des trafics en transit sur les GGSN à partir des paramètres que nous connaissons. Pour un GGSN g :

$$TF_g = \sum_s F_s \alpha_g \quad \text{, et} \quad TB_g = \sum_s B_s \alpha_g \tag{6.2}$$

Une contrainte importante sur les réseaux radios est la limitation des PDP Context ouverts simultanément sur les GGSN. Il existe en réalité deux limitations : $NMax_g^{serv}$ (respectivement $NMax_g$) qui est la valeur maximale de PDP Context ouverts sur le GGSN g pour le service $serv$ (respectivement

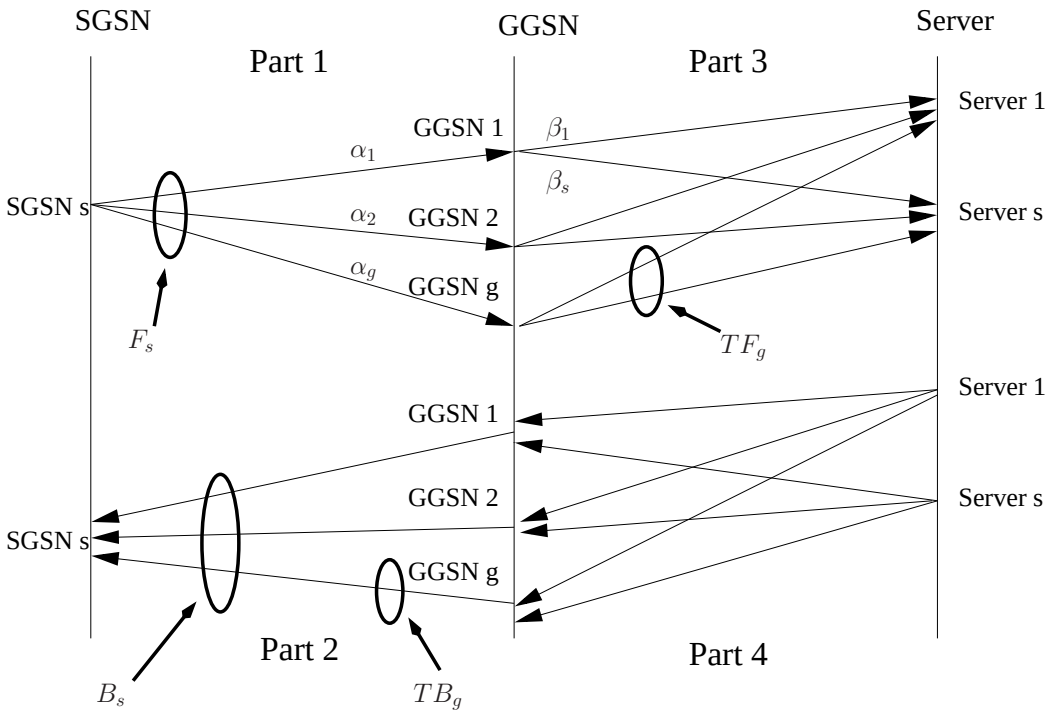


FIG. 6.3 – Exemple de propagation radio et notations associées pour un service donné

pour tous services confondus). Nous pouvons écrire simplement le nombre de connexions simultanées sur le GGSN g :

$$\begin{aligned} N_g^{serv} &= \sum_s N_s \alpha_g \quad \text{pour le service } serv \\ N_g &= \sum_{\text{serv}} N_g^{serv} \quad \text{de manière globale} \end{aligned} \tag{6.3}$$

Nous avons donc deux relations intéressantes : une liant la charge des liens aux poids DNS et au routage IP et une autre liant les contraintes sur les GGSN aux poids DNS. Nous proposons dans la partie suivante une optimisation des poids DNS dans le but de diminuer l'utilisation globale des liens tout en respectant les contraintes sur les GGSN.

6.2.2 Optimisation des ressources par les poids DNS

L'utilisation des interfaces est un premier critère de performance sur un réseau. Diminuer l'utilisation globale des interfaces sur le réseau permet de diminuer la congestion et les délais de manière simple. Pour se faire, nous proposons le modèle de coût Γ suivant (avec C_l, Y_l qui sont respectivement la capacité et la charge du lien l) :

$$\Gamma(\alpha,\beta) = \sum_{l\in E} \Gamma_l(\alpha,\beta) \quad \text{où} \quad \Gamma_l(\alpha,\beta) = \begin{cases} (\frac{Y_l}{C_l})^2 & \text{si } Y_l < \gamma C_l \\ (\frac{Y_l}{C_l})^2 + K(\frac{Y_l - \gamma C_l}{C_l})^2 & \text{sinon} \end{cases} \tag{6.4}$$

Ce modèle permet de tenir compte de la saturation d'un lien en pénalisant (coefficient K) les utilisations supérieurs à γ. On pourra choisir $\gamma = 0.7$ par exemple. Il permet également de répartir au mieux les charges sur le réseau. Le problème d'optimisation P que nous proposons de résoudre est résumé ci-dessous.

Minimiser $\Gamma(\alpha, \beta)$

Sous les contraintes :

$$N_g < NMax_g \quad \text{pour tous les GGSN } g \tag{6.5}$$

$$\sum_g \alpha_g = \sum_{sv} \beta_{sv} = 1 \tag{6.6}$$

La première contrainte (6.5) qui traduit les limitations sur les GGSN (qui peut être reformulée à l'aide de l'équation (6.3)) est ramenée dans la fonction objectif sous forme de pénalisation dépendant d'un paramètre ϵ. Cela nous permet d'écrire le problème $P(\epsilon)$.

Minimiser $\Gamma(\alpha, \beta) + \frac{1}{\epsilon}\Gamma N(\alpha, \beta)$

Où :

$$\Gamma N(\alpha, \beta) = \sum_g \left(\max(0, N_g - NMax_g)\right)^2 \tag{6.7}$$

Sous les contraintes :

$$\sum_g \alpha_g = \sum_{sv} \beta_s = 1$$

La solution du problème P_ϵ (noté $(\alpha_\epsilon, \beta_\epsilon)$) converge vers la solution du problème P (noté (α^*, β^*)) quand ϵ tend vers 0. Nous avons donc résolu le problème P en résolvant de manière itérative les problèmes P_ϵ pour des valeurs de ϵ de plus en plus petites. La résolution du problème P_ϵ est faite à l'aide d'un gradient projeté car la formulation des contraintes est relativement classique et le critère est quadratique. Nous présentons nos résultats dans la partie suivante.

6.2.3 Résultats

L'algorithme proposé ci-dessus a été testé sur plusieurs configurations différentes. Nous rassemblons dans cette partie les principaux résultats obtenus. Les topologies sont présentées dans le tableau 6.1. Les trafics propagés dans chaque topologie sont générés à l'aide du logiciel NEST [94] et le nombre de services traités ainsi que le nombre total de connexions sont également donnés dans le tableau 6.1. Enfin, le tableau 6.2 contient les résultats obtenus sur chaque topologie de test avec des poids DNS initiaux tous égaux.

Nous pouvons voir dans ces quelques résultats que la contrainte sur les limitations des GGSN est bien respectée dans les solutions proposées alors qu'avec des poids DNS identiques, cela n'était pas le cas. Nous pouvons également relever la diminution de l'utilisation maximale du réseau. Cette diminution n'est pas toujours présente. Certains tests ont montré une augmentation de cette valeur ce qui n'est

Topologie	#nœuds	#liens	#SGSN	#GGSN	#serveur	#services	#PDP
Test1	41	132	3	3	3	1	6429
Test2	41	132	3	3	3	2	11976
Test3	45	174	3	5	3	3	11997

TAB. 6.1 – Données de test

Topologie	Test 1	Test 2	Test 3
Utilisation maximale **avant** optimisation	71.6	80.8	69.4
Utilisation moyenne **avant** optimisation	18.2	19.4	14.3
#PDP reroutés **avant** optimisation	143	1592	1886
Utilisation maximale **après** optimisation	**59.6**	**69.3**	68.6
Utilisation moyenne **après** optimisation	17.5	18.4	13.9
#PDP reroutés **après** optimisation	**0**	**0**	**0**

TAB. 6.2 – Résultats

pas un bon résultat en soi. Cependant, cela se justifie par le poids donné à l'optimisation des contraintes par rapport à celle de l'utilisation des interfaces. Même si aucun résultat n'est montré ici, notre optimisation fonctionne également dans le cas de plusieurs DNS. Cela permet un choix plus large de solutions tout en considérant l'affectation entre SGSN (ou GGSN) et DNS comme donnée une fois pour toute.

6.3 Analyse de la résilience

Il est rapidement apparu que le logiciel souffrait d'un manque quant à l'analyse de la résilience. Pourtant, la partie analyse de ce comportement pouvait être simplement faite : il suffisait d'effectuer un routage, une propagation des flots puis une analyse de différents critères et ce pour l'ensemble des pannes possibles.

Nous avons donc décidé d'intégrer un nouveau module de calcul au logiciel, celui-ci permettant initialement l'étude de l'impact des pannes sur le réseau. Totalement lié au module de calcul du routage et de la propagation des flots, l'algorithme développé permet d'un simple clic de souris d'exécuter une analyse des critères principaux sur le réseau suite à chaque panne d'équipement.

Les paramètres utilisés lors de l'exécution de l'algorithme sont relativement simples : les types d'équipement dont les pannes doivent être simulées, le seuil d'utilisation définissant une congestion sur une interface, le(s) système(s) autonome(s) sur lesquels simuler des pannes.

Nous proposons en même temps une analyse de la biconnexité du réseau et une hiérarchisation de l'impact des pannes. En effet, nous proposons de classer les pannes selon 4 grandes catégories qui sont, dans un ordre décroissant de gravité,

1. les pannes rendant le réseau non connexe,
2. les pannes entraînant une congestion sur le réseau,

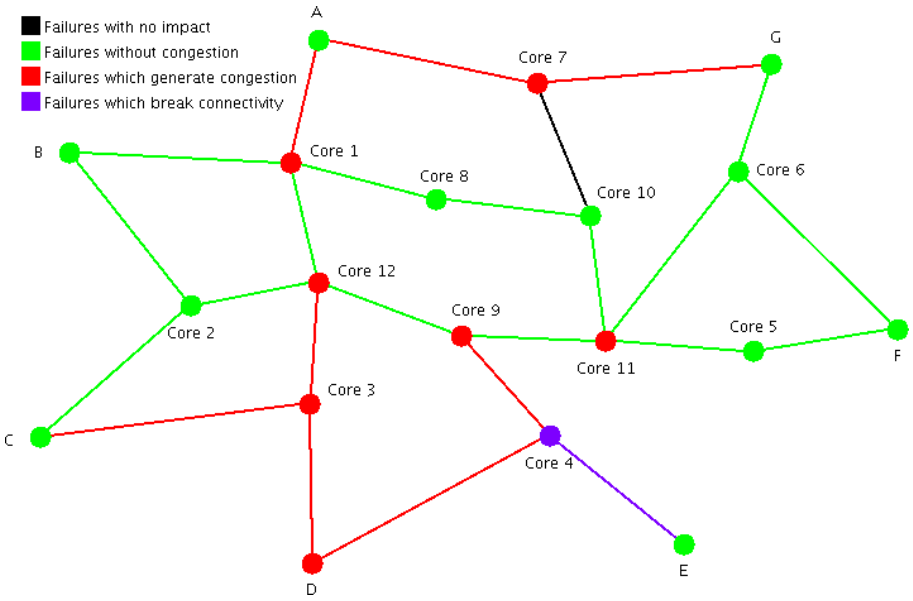


FIG. 6.4 – Coloration en fonction de la gravité

3. les pannes ayant un impact non négligeable sur l'utilisation des ressources mais ne générant aucune congestion,
4. et les pannes sans aucun impact sur l'utilisation des ressources.

Ainsi, comme le montre la figure 6.4, une fois l'algorithme exécuté, l'utilisateur peut d'un simple regard sur sa topologie visualiser la coloration des différents équipements qui dépend de la gravité entraînée en cas de panne de cet équipement. Les résultats proposés portent également sur l'utilisation des ressources et sur l'impact sur les LSP éventuels comme l'illustre la figure 6.5 montrant les résultats généraux de l'étude effectuée ou la figure 6.6 qui montre les résultats pour chaque panne de nœud possible. Le bouton « View Routing Detail For Failure » permet d'un simple clique de souris de lancer le module de calcul de routage et de propagation des flots sur le réseau avec l'équipement sélectionné en panne. Des informations supplémentaires comme les routes peuvent donc être rapidement obtenues.

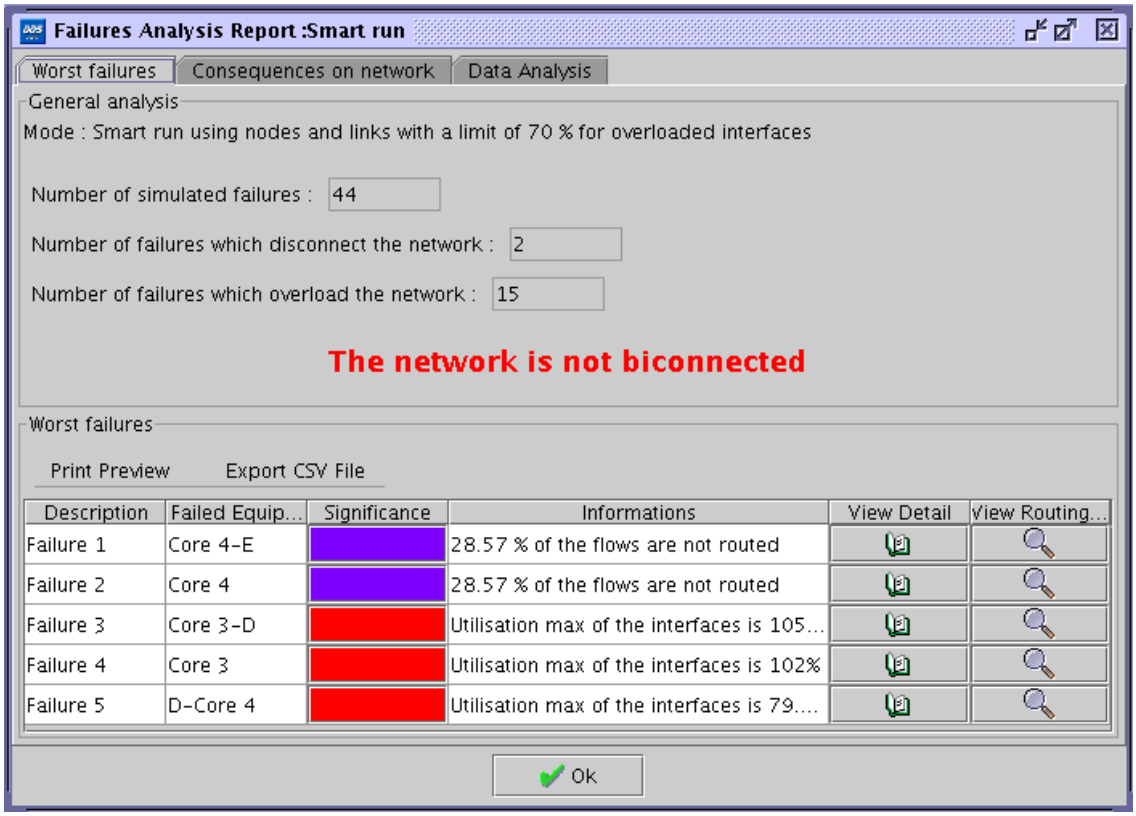


FIG. 6.5 – Détail de l'étude et pires cas rencontrés

6.4 Conclusion

Ce chapitre au contenu scientifique plus pauvre que les premiers contient les travaux majeurs effectués pour la start-up QoS Design dans le cadre de ma thèse Cifre. Il présente donc

- un ensemble d'améliorations apportées au module de routage du logiciel NEST et portant principalement sur :
  - l'intégration des trafics multicasts
  - la prise en compte des trafics radios 2.5/3G
  - une évolution majeure du traitement des VPN
- une optimisation des poids de routage DNS associés aux réseaux 2.5/3G
- l'ajout d'un module d'analyse de la résilience des réseaux.

Il est important de noter que l'ensemble des algorithmes issus des travaux réalisés et décrits dans les premières parties de cette thèse sont exécutables à l'aide du logiciel NEST. De plus, à l'exception de l'estimation des matrices de trafics, ils sont tous utilisables de manière conviviale car totalement intégrés à l'environnement NEST à l'aide d'interfaces graphiques permettant la configuration des paramètres et la visualisation des résultats.

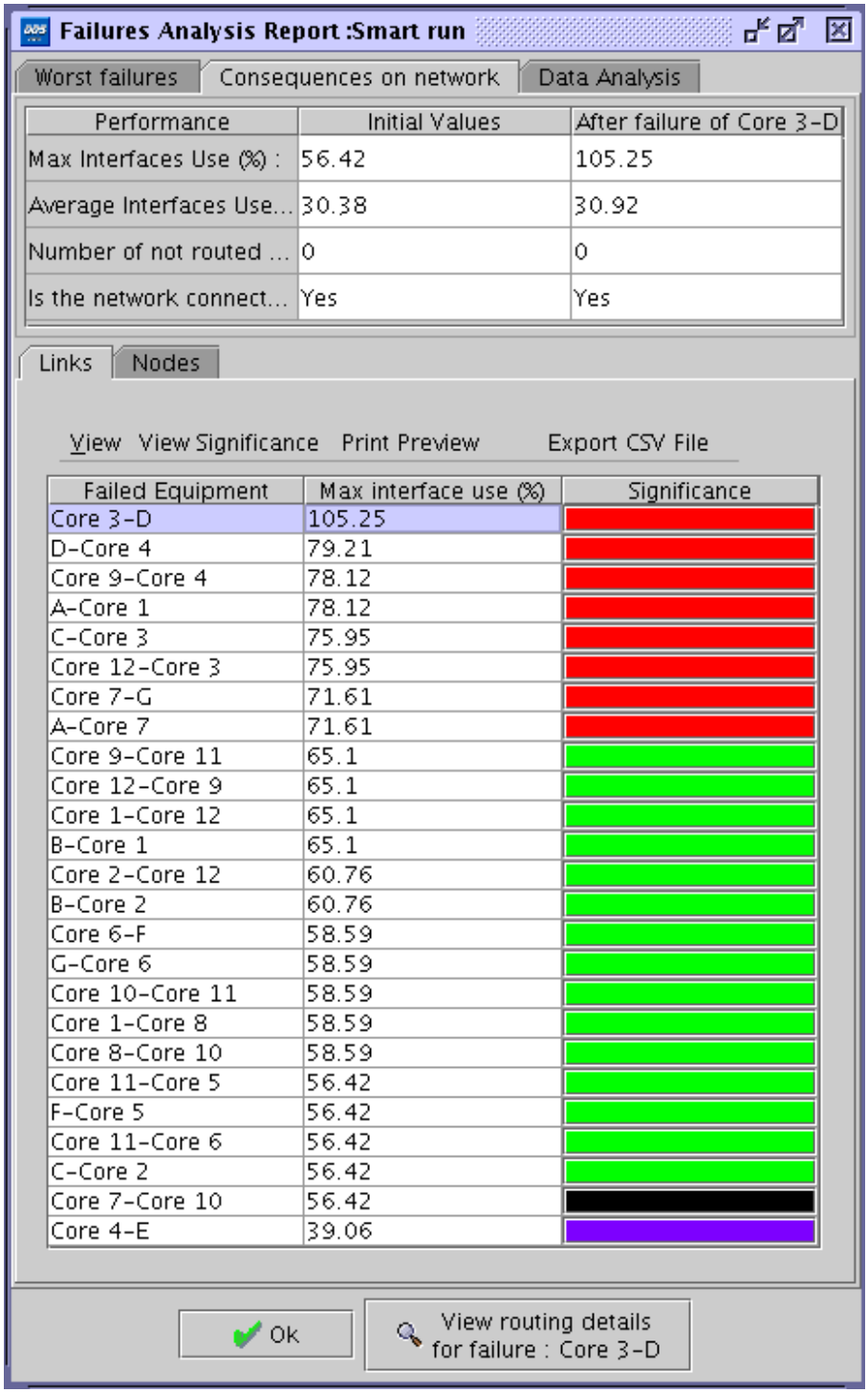


FIG. 6.6 – Détail de l'impact de chaque panne (ici les nœuds)

Conclusion

Internet et les réseaux IP ont des difficultés à faire face à leur paradoxe. La simplicité du service « Best Effort » initialement proposé a entraîné un succès planétaire des réseaux IP caractérisé par une utilisation de plus en plus massive d'un nombre toujours croissant d'utilisateurs. Mais ce succès a, à son tour, entraîné l'apparition de nouveaux services « universels » qui nécessitent plus de bande passante et une meilleure qualité de service et qui ne peuvent fonctionner sur l'architecture IP initiale.
.

Les réseaux IP ont donc subi un ensemble d'évolutions technologiques majeures pour leur permettre d'écouler des volumes de trafics en perpetuelle croissance et aux besoins variés en qualité de service. Cependant, comme nous l'avons montré dans le chapitre 1, une nouvelle approche d'ingénierie des réseaux doit être mise en place pour permettre une planification et une gestion efficace des ressources dans le but de garantir, de manière continue, la qualité de service tout en minimisant les coûts résultants.

Cette approche doit, à notre avis, être basée sur un premier axe important : la supervision du réseau. Nous avons alors proposé, dans le chapitre 2, une contribution au problème d'estimation des matrices de trafic. Ce problème est connu pour être une alternative prometteuse à la métrologie des flots qui est trop lourde à supporter pour les réseaux opérationnels de part un trafic de reporting important et une consommation CPU sur les routeurs ralentissant leur fonctionnalité principale de routage.

Partant d'une hypothèse de faible variation des répartitions des trafics dans le temps, nous avons proposé un algorithme basé sur des techniques de descente admissible et qui donne de bons résultats. Un ensemble de tests sur données simulées ont permis de valider l'approche retenue ainsi que l'algorithme proposé avec des résultats se situant dans des normes acceptables pour un opérateur.

Selon notre point de vue, le deuxième axe important sur lequel doit être basée la nouvelle approche d'ingénierie concerne la planification des réseaux. Pour cette raison, nous avons détaillé nos contributions sur l'optimisation du routage IP dans le chapitre 3 de ce mémoire. La recherche des métriques IP permettant d'obtenir des performances optimales est un problème combinatoire connu pour être NP difficile. Nous avons alors proposé un nouvel algorithme donnant des solutions incrémentales de bonne qualité dans des temps de calculs plus faibles que ceux des algorithmes existants. L'aspect incrémental

des solutions proposées permet leur configuration aisée par les opérateurs. Nous avons étendu l'algorithme incrémental de référence pour proposer des solutions plus performantes mais dans des temps de calcul rallongés d'une part, et d'autre part proposer un routage robuste tenant compte des pannes.

Etant donnée l'utilisation massive du protocole MPLS pour le routage dans les cœurs de réseau, nous avons justifié, dans le chapitre 4, l'intérêt de l'optimisation des métriques IP. Nous avons alors montré sur plusieurs exemples variés que l'optimisation des métriques IP permet une meilleure utilisation des ressources que les LSP lorsque ceux-ci sont configurés de manière usuelle. Une configuration plus fine des LSP et de leurs FEC peut cependant mener à une utilisation efficace des ressources. Néanmoins, même si un algorithme performant de placement des LSP peut être utilisé (ILSP), la difficulté de bien définir les FEC est un problème qui persiste. De part les tests effectués, nous avons également conclu que l'optimisation des métriques IP est un préalable intéréssant au placement des LSP par des techniques automatiques telles que LDP ou PCalc.

Planifier un réseau pour qu'il puisse supporter certains volumes de trafics revient à choisir son routage certes, mais aussi à décider quelles capacités doivent être affectées aux liens de la topologie. Dans le chapitre 5, nous avons donc proposé une nouvelle approche du problème du dimensionnement des réseaux. La formulation de ce nouveau problème est notre contribution principale et se distingue des travaux antérieurs par la prise en compte de contraintes matérielles et par l'utilisation de coûts réalistes. Nous avons proposé un algorithme exact pour résoudre ce problème combinatoire. Cependant, pour contourner les problèmes de passage à l'echelle, nous avons aussi proposé une heuristique aux temps de calcul faibles. Lors de nos différents tests, nous avons montré l'intérêt d'integrer les contraintes matérielles pour pouvoir fournir une solution configurable par la suite. Nous avons aussi obtenu des gains économiques conséquents grâce à la prise en compte du matériel existant.

Enfin, nous avons exposé dans le chapitre 6 différentes contributions concernant le logiciel NEST de la start-up QoS Design avec laquelle j'ai travaillé dans le cadre d'une bourse Cifre. Comme nous l'avons expliqué, ces contributions portent principalement sur des évolutions importantes du moteur de routage ainsi que sur une analyse facilitée de la résilience et sur l'intégration des trafics radio 2.5/3G. De plus, toutes les contributions traitées dans les chapitres précédents ont été intégrées dans ce logiciel.

BIBLIOGRAPHIE

[1] M. Frank and P. Wolfe. An algorithm for quadratic programming. Naval Research Logistics Quarterly, 3 (1956), pp. 95–110.

[2] Richard Bellman. On a routing problem, in quarterly of applied mathematics, 1958.

[3] E. W. Dijkstra. A note on two problems in connexion with graphs, 1959.

[4] L.S. Lasdon. Optimization theory for Large Systems. *Macmillan*, 1970.

[5] B. Meister, H. Muller, and H. Rudin. New optimization criteria for message-switching networks. *Communications, IEEE Transactions on [legacy, pre - 1988]*, Vol 19 Issue : 3 :256–260, Jun 1971.

[6] B. Meister, H. Muller, and H. Rudin. On the optimization of message-switching networks. *Communications, IEEE Transactions on [legacy, pre - 1988]*, Vol 20, no.1 :8–14, Feb 1972.

[7] M. Held and P.Wolfe and H. Crowder. Validation of subgradient optimization. *math. Program. 6*, 1974.

[8] L. Kleinrock. *Queueing Systems*, volume Vol 1 : Theory. Wiley-Interscience, 1976.

[9] L. Kleinrock. *Queueing Systems*, volume Vol 2 : Computer Applications. Wiley-Interscience, 1976.

[10] K. Maruyama and D. T. Tang. Discrete link capacity assignment in communication networks. In *ICCC*, pages 92–97, 1976.

[11] K. Maruyama, L. Fratta, and D. T. Tang. Heuristic design algorithm for computer communication networks with different classes of packets. *j-IBM-JRD*, Vol .21(4) :360–369, jul 1977.

[12] K. Maruyama and D. T. Tang. Discrete link capacity and priority assignments in communication networks. *IBM Journal of Research and Development*, Vol 21 :254–263, May 1977.

[13] D. P. Bertsekas. Algorithm for nonlinear multicommodity flow problems. In *International Symposium on systems Optimization and Analysis*, Springer-Verlag , 1979.

[14] M.R. Garey and D.S. Johnson. Computers and Intractability-A Guide to the Theory of NP-Completeness. Freeman, 1979.

[15] D. P. Bertsekas and E. M. Gafni. Projected newton methods and optimization of multicommodity flows. *LIDS-P-1140*, 1981.

[16] IETF. Rfc 791 - ip internet protocol, 1981.

[17] IETF. Rfc 793 - tcp transmission control protocol, 1981.

[18] Y. Sheffi. Urban transportation networks : Equilibrium analysis with mathematical programming methods, 1985.

[19] Sen. Maximum likelihood estimation of gravity model parameters, 1986.

[20] Cisco. Eigrp - enhanced interior gateway routing protocol, 1986.

[21] IETF. Rfc 1058 - rip routing information protocol, 1988.

[22] Dimitri BERTSEKAS and J. TSITSIKLIS. *Parallel and Distributed Computation : Numerical methods*. Prentice-Hall Inc., 1989.

[23] IETF. Rfc 1142 - is-is intra-domain routing protocol, 1990.

[24] D. P. Bertsekas. *Linear Network Optimization*. MIT Press, 1991.

[25] D.S Johnson and C.R. Aragon and L.A. McGeoch and C. Schevon. Optimization by simulated annealing : an experimental evaluation. *Oper. Res. 39(1)*, May-June 1991.

[26] IETF. Rfc 1267 - bgp border gateway protocol, 1991.

[27] D. P. Bertsekas and R. Gallager. *Data Networks 2nd ed.* Prentice Hall, 1992.

[28] Edsger W. Dijkstra. Bulterman's theorem on shortest tree. http ://www.cs.utexas.edu/users/EWD/ewd11xx/EWD1131.pdf, July 1992.

[29] D. Burton. *the inverse shortest path problem*. PhD thesis, Department of Mathematics, FUNDP, Namur, Belgium, 1993. citeseer.ist.psu.edu/article/burton93inverse.html.

[30] Berka and Boyce. Implementation and solution of a large asymmetric network equilibrium model, 1994.

[31] IETF. Rfc 1633 - integrated services in the internet architecture : an overview, 1994.

[32] M. West. Statistical inference for gravity models in transportation flow forecasting, 1994.

[33] IETF. Rfc 1771 - mpbgp (bgp version 4), 1995.

[34] Lange K. A gradient algorithm locally equivalent to the em algorithm, 1995.

[35] Matthew L. Ginsberg William D. Harvey. Limited discrepancy search. In Chris S. Mellish, editor, *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95) ; Vol. 1*, pages 607–615, Montréal, Québec, Canada, August 20-25 1995. Morgan Kaufmann, 1995.

[36] G. Ramalingam and Thomas W. Reps. An Incremental Algorithm for a Generalization of the Shortest-Path Problem. *J. Algorithms*, 21(2) :267–305, 1996. citeseer.ist.psu.edu/ramalingam92incremental.html.

[37] IETF. Rfc 1902 - snmp simple network management protocol, 1996.

[38] Y. Vardi. Network tomography : Estimating source-destination traffic intensities from link data. *Journal of the American Statistical Association*, 91 :365–377, June 1996.

[39] IETF. Rfc 2205 - rsvp resource reservation protocol, 1997.

[40] M. Pioro and P. Gajoznicwek. solving multicommodity integral flow problems by simulated allocation. *telecommun. Systems 7(1-3)*, pages 17–28, 1997.

[41] A. Ouorou, P. Mahey, and J.-P. Vial. A survey of algorithms for convex multicommodity flow problems. Technical report, Switzerland, 1997.

[42] A. Farago and b. Szviatovszki. Allocation of administrative weight in PNNI. In *Proceedings of the NETWORKS'98*, Sorrento, 1998.

[43] Dimitri P. Bertsekas. *Network Optimization : Continuous and Discrete Models*. Athena Scientific, may 1998.

[44] IETF. Rfc 2475 - an architecture for differentiated services, 1998.

[45] IETF. Rfc 1247 - ospf open shortest path first, 1998.

[46] Claudia Tebaldi and Mike West. Bayesian inference on network traffic using link count data. *Journal of the American Statistical Association*, 93 (442) :557–576, June 1998. With comments by Y. Vardi and Robert McCulloch, and a rejoinder by the authors.

[47] Y. Vardi. Comment : Bayesian inference on network traffic using link count data. *Journal of the American Statistical Association*, 93(442) :573– ?, June 1998.

[48] ILOG S.A. *ILOG CPLEX 6.5 Reference Manual*, 1999.

[49] Suwan Runggeratigul and Sawasd Tantaratana. Link capacity assignment in packet-switched networks : The case of piecewise linear concave cost function, 1999.

[50] P. Mahey A. Ouorou. Minimum mean-cycles cancelling method for nonlinear multicommodity flow problems. *EJOR*, 121 :532–548, 2000.

[51] Bernard Fortz and Mikkel Thorup. Internet Traffic Engineering by Optimizing OSPF Weights. In *INFOCOM (2)*, pages 519–528, 2000.

[52] J. Cao, D. Davis, S. Wiel, and B. Yu. Time-varying network tomography : Router link data, 2000.

[53] B. John Oommen and T. Dale Roberts. Continuous learning automata solutions to the capacity assignment problem. *IEEE Trans. Comput.*, 49(6) :608–620, 2000.

[54] W. Ben Ameur and E. Gourdin and B. Liau. Dimensioning of internet networks. In *Proceedings of the DRCS'2000*, Munich, 2000.

[55] IETF. Rfc 3031 - multiprotocol label switching architecture, 2001.

[56] IETF. Rfc 3032 - mpls label stack encoding, 2001.

[57] IETF. Rfc 3036 - ldp label distribution protocol, 2001.

[58] IETF. Rfc 3209 - rsvp-te, 2001.

[59] A. Benchakroun et F. Boyer P. Mahey. Capacity and flow assignment of data networks by generalized benders decomposition. *Journal of Global Optimization*, 20–2 :173–193, 2001.

[60] W. B. Ameur and N. Michel and B. Liau. Routing Strategies for IP Networks. *Telektronikk Magazine*, 2/3 :145–158, 2001.

[61] W. Ben Ameur and B. Liau. Calcul des métriques de routage pour Internet . *Ann. Telecommun 56*, pages 3–4, 2001.

[62] M. Fortz, B. ; Thorup. Optimizing ospf/is-is weights in a changing world. *Selected Areas in Communications, IEEE Journal on*, 20 (4) :756–767, May 2002.

[63] IETF. Rfc 3212 - cr ldp constraint-based routed ldp, 2002.

[64] M. Pioro and A. Szentesi and J. Harmatos and A. Jüttner and P. Gajowniczek and S.Kozdrowski. On open shortest path first related network optimization problems. *Performance evaluation 48*, pages 201–223, 2002.

[65] A. Medina, N. Taft, S. Battacharya, C. Diot, and K. Salamatian. Traffic matrix estimation : Existing techniques compared and new directions, 2002.

[66] P. B. Luh and S. Song. Modeling and Optimization of Complex Networked Systems, 2002.

[67] B. Fortz and M. Thorup. Robust optimization of ospf/is-is weights, 2003.

[68] IETF. Rfc 3468 - mpls the multiprotocol label switching, 2003.

[69] L.S. Buriol and M. G. C. Resende. speeding up dynamic shortest path algorithms. *AT&T Labs Research Technical Report TD-5RJ8B*, 2003.

[70] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg. Fast accurate computation of large-scale ip traffic matrices from link loads, 2003.

[71] Y. Zhang, M. Roughan, C. Lund, and D. Donoho. An information-theoretic approach to traffic matrix estimation, 2003.

[72] Jean-Marie Garcia, Anoua Rachdi, and Olivier Brun. Optimal lsp placement with qos constraints in diffserv/mpls networks. In *18th International Teletraffic Congress*, september 2003.

[73] Bernard Fortz and Mikkel Thorup. Increasing internet capacity using local search. *Comput. Optim. Appl.*, 29(1) :13–48, 2004.

[74] Anders Gunnar, Mikael Johansson, and Thomas Telkamp. Traffic matrix estimation on a large IP backbone : a comparison on real data. In *Internet Measurement Conference*, pages 149–160, 2004.

[75] Iftekhar Hussain. *Fault-Tolerant IP and MPLS Networks*. Cisco Press, 2004.

[76] A. Lakhina, K. Papagiannaki, M. Crovella, C. Diot, E. Kolaczyk, and N. Taft. Structural analysis of network traffic flows, 2004.

[77] A. Medina, C. Diot, I. Matta, and K. Salamatian. A two-step statistical approach for inferring network. Technical report, 2004.

[78] Antonio Nucci, Rene Cruz, Nina Taft, and Christophe Diot. Design of IGP link weight changes for estimation of traffic matrices. In *IEEE Infocom*, Hong Kong, march 2004.

[79] Konstantina Papagiannaki, Nina Taft, and Anukool Lakhina. A distributed approach to measure IP traffic matrices. In *Internet Measurement Conference*, pages 161–174, 2004.

[80] Suwan Runggeratigul. A memetic algorithm for communication network design taking into consideration an existing network. pages 615–626, Norwell, MA, USA, 2004. Kluwer Academic Publishers.

[81] Augustin Soule, Antonio Nucci, Rene Cruz, Emilio Leonardi, and Nina Taft. How to identify and estimate the largest traffic matrix elements in a dynamic environment. In *SIGMETRICS 2004/PERFORMANCE 2004 : Proceedings of the joint international conference on Measurement and modeling of computer systems*, pages 73–84, New York, NY, USA, 2004. ACM Press.

[82] Augustin Soule, Anukool Lakhina, Nina Taft, Konstantina Papagiannaki, Kave Salamatian, Antonio Nucci, Mark Crovella, and Christophe Diot. Traffic matrices : balancing measurements, inference and modeling. *SIGMETRICS Perform. Eval. Rev.*, 33 (1) :362–373, 2005.

[83] Augustin Soule, Kavé Salamatian, Antonio Nucci, and Nina Taft. Traffic matrix tracking using kalman filters. *SIGMETRICS Perform. Eval. Rev.*, 33 (3) :24–31, 2005.

[84] Sandrine Vaton, Jean sebastien Bedo, and Anni Gravey. Advanced methods for the estimation of the origin destination traffic matrix. *Advanced Methods for the Estimation of the Origin Destination traffic matrix. Performance Evaluation and Planning Methods for the Next Generation Internet (25 th du GERARD)*, XVI, 2005.

[85] Hong Hsu YEN and Frank Yeong Sung LIN. Delay constrained routing and link capacity assignment in virtual circuit networks(network). *IEICE transactions on communications*, 88(5) :2004–2014, May 2005.

[86] C. Duhamel and P. Mahey. Multicommodity flow problems with a bounded number of paths : a flow deviation approach. *Networks*, 49 :80–89, 2006.

[87] I.Juva, S.Vaton, and J.Virtamo. Quick traffic matrix estimation based on link count covariances. *2006 IEEE International Conference on Communications (ICC 2006),Istanbul*, 2006.

[88] Mohamed Anouar Rachdi. Optimisation des ressources de réseaux hétérogènes avec coeur de réseau mpls, 2007.

[89] Guanghui. Bellman-Ford algorithm. http ://www.ece.utexas.edu/projects/ece/lca/sponsors/ motorola_docu/0808_Bellman-Ford.pdf.

[90] P.Bermolen, S.Vaton, and I.Juva. Search for optimality in traffic matrix estimation : a rational approcah by cramer-rao lower bounds. *2nd EuroNGI Conference on Next Generation Internet Design and Engineering, Valencia, Spain*, 3-5 april 2006.

[91] Cisco Systems. Low latency queuing (llq).

[92] Cisco Systems. IGRP : technical documentation. http ://www.cisco.com/en/US/tech/tk365/tk352/tech_tech_notes_list.html.

[93] Cisco. Cisco ios netflow
http ://www.cisco.com/en/us/products/
ps6601/products_ios_protocol_group_home.html.

[94] QoSDesign. Nest : Network engineering and simulation tool
http ://www.qosdesign.com.

[95] Wikipedia. http ://fr.wikipedia.org/wiki/qos.

Résumé

Les réseaux IP sont devenus des systèmes réellement critiques, l'interruption du service fourni par le réseau ou même une dégradation significative de la qualité de service étant de moins en moins tolérables. Une nouvelle approche de l'ingénierie des réseaux devient des lors nécessaire, pour concevoir, planifier et contrôler les architectures IP sur la base des informations de supervision. Nos contributions a cette nouvelle approche portent sur l'estimation du trafic a partir des mesures de charges SNMP, sur l'optimisation des métriques de routage IP et sur le dimensionnement d'infrastructures. Les modèles et algorithmes développés prennent en compte de nombreuses contraintes technologiques dans le but de fournir des solutions opérationnelles.

IP networks have become critical systems in the last decade : service interruptions or even significant service degradations are less and less tolerable. Therefore, a new network engineering approach is required to help design, plan and control IP architectures on the basis of supervision information. Our contributions to this new approach are related to traffic matrix estimation from SNMP link loads, to IP routing weights optimization and to network dimensioning. The models and algorithms proposed in this thesis take into account many technological constraints in order to provide operational solutions.

Mots clés : Optimisation, Routage, Dimensionnement, Internet, Réseaux IP, MPLS, Ingénierie des réseaux.