Dimensionnement robuste des réseaux de télécommunications face à l'incertitude de la demande

Georgios Petrou

THÈSE

présentée pour l'obtention du titre de

Docteur de l'Université Paris 1, Panthéon-Sorbonne

Spécialité : Mathématiques Appliquées

par Georgios PETROU

Sujet : *Dimensionnement robuste des réseaux de télécommunication face à l'incertitude de la demande*

Soutenance le 3 octobre 2008 devant le jury composé de :

Rapporteurs :	Walid BEN AMEUR	TELECOM & Management SudParis
	Phillippe MAHEY	Université Clermont-Ferrand II
Examinateur :	Jean-Marc BONNISSEAU	Université Paris I
Direction :	Claude LEMARÉCHAL	INRIA Grenoble - Rhône-Alpes
Co-direction :	Adam OUOROU	France Télécom R&D

Cette thèse a été effectuée à France Télécom Division R&D, en partenariat avec le CERMSEM de l'Université de Paris I et l'INRIA.

Cette thèse est un travail commun dans le cadre d'une Convention Cifre, signée entre l'ANRT et France Télécom Division R&D sous le numéro 642/2003. Elle a été effectuée sous la supervision d'Adam Ouorou, ingénieur de recherche (représentant France Télécom Division R&D) et Claude Lemaréchal (Inria).

L'essentiel de cette étude consiste en trois articles, présentés dans les chapitres 2, 3 et 4. Le chapitre 1 constitue une présentation générale du support bibliographique et méthodologique des divers articles examinés, de la recherche effectuée, ainsi que de toutes les étapes franchies afin d'arriver à la rédaction finale desdits articles.

Remerciements

Je tiens à remercier mes directeurs de recherche, Adam Ouorou et Claude Lemaréchal. Pour commencer, un grand merci à Adam Ouorou (ingénieur de recherche, France Télécom R&D) qui est à l'origine de cette thèse. En effet, à ma demande d'un stage en télécommunications, il a répondu avec une offre de thèse avec Convention CIFRE au sein de France Télécom R&D sous la direction de lui-même et de Claude Lemaréchal. Il a su m'aider avec tous les – nombreux – problèmes que j'ai rencontrés pendant mes années de thèse.

Je suis également particulièrement reconnaissant à mon deuxième superviseur Claude Lemaréchal (directeur de recherche, Institut National de Recherche en Informatique et en Automatique, Rhônes-Alpes). Je le remercie de m'avoir donné l'opportunité de faire ma thèse avec lui, ce qui s'est avéré une expérience unique : à travers un travail scientifiquement et pratiquement difficile, j'ai eu la chance de rencontrer non seulement le chercheur "Professeur Lemaréchal", mais également l'homme Claude.

Un merci tout particulier à Jacek Gondzio (professeur, Département de Mathématiques, Université d'Edinburgh, Ecosse, Royaume-Uni) qui m'a accepté en master2 de Recherche Opérationnelle à l'Université d'Edinburgh et m'a ainsi offert l'opportunité d'effectuer cette thèse.

Je remercie également Andreas Grothey (chargé de cours, Département de Mathématiques, Université d'Edinburgh, Ecosse, Royaume-uni) pour avoir pris le temps de répondre à mes questions et m'avoir ainsi aidé pendant une période difficile de mes recherches à Paris.

J'adresse ma profonde reconnaissance à mes responsables hiérarchiques à EDF (Yannick Jacquemart, François-Régis Monclar, Luciano Leal-de-Sousa) pour leur confiance et pour m'avoir permis de prendre le temps de finir convenablement ma thèse, malgré la pression qu'ils ont reçue de la part du département RH d'EDF.

J'exprime ma gratitude à Walid Ben-Ameur (professeur, TELECOM & Management Sudparis) et Philippe Mahey (professeur, ISIMA, Université Blaise Pascal Clermont II) pour l'honneur qu'ils m'ont fait d'avoir accepté d'être rapporteurs de cette thèse.

Je tiens à remercier également tous mes amis pour leur manière de me faire oublier le stress et la pression de ma thèse : Giannis, Kostas, Vasilis, Giorgos, Konstantis, Christos, Eirini.

Je salue chaleureusement ma mère et mon frère pour leur compréhension et surtout leur soutien et support indispensables à tous les niveaux, matériel et psychologique.

Pour finir, je remercie Niki pour ses traductions, ses relectures et son tendre soutien.

Table des matières

Chapitre 1

Présentation

Ce premier chapitre constitue une présentation générale et rapide du travail global qui fut effectué pour la rédaction de cette thèse. Commençant par les bases de notre recherche, nous examinons en premier lieu les caractéristiques principales des réseaux de télécommunications. Nous présentons ensuite une compilation d'articles des divers domaines relatifs à notre recherche ; notamment quelques articles concernant la recherche sur divers problèmes de télécommunications. Ces articles ont été utilisés comme guides pour les problèmes étudiés dans notre propre recherche. Nous insistons principalement sur le travail qui a été fait dans le domaine de l'optimisation robuste et sur ses différences avec l'optimisation stochastique. De plus, nous introduisons les outils mathématiques utilisés pour nos algorithmes. Ainsi, ce chapitre introduit les articles constituant l'essentiel de la thèse en présentant toute l'activité qui les a précédés.

1.1 Réseaux de télécommunication

Pourquoi l'incertitude et pourquoi France Télécom ? Bien évidemment, l'incertitude existe partout et il existe également de nombreuses entreprises qui fournissent des services de télécommunication. Cependant, le rôle de France Télécom est plus crucial, dès lors qu'elle constitue le fournisseur principal en France dans le domaine des télécommunications et que la fiabilité de ses services est une de ses plus hautes priorités. Par ailleurs, le secteur des télécommunications se développe approximativement au même rythme que celui des ordinateurs, puisqu'il s'agit de deux technologies étroitement liées, et qu'en plus, de nos jours, les ordinateurs dépendent des réseaux. Ce phénomène oblige les principaux fournisseurs à porter une grande attention à la façon de développer leur réseau et leurs services, afin de se maintenir à la page par rapport à l'expansion des technologies informatiques et des services en technologie de l'information. Il est essentiel que, quelle que soit la demande et l'usage de la bande passante dans le futur, le réseau soit fiable et assure la haute qualité de ses services (QoS : *Quality of Service*).

Lorsqu'on se réfère aux réseaux de télécommunication, on doit avoir en tête que ce domaine concerne une partie essentielle de notre vie quotidienne. Dès les années 70, l'internet permettait la communication entre les universités et entre divers centres de recherche. Jusqu'à cette époque, les entreprises de télécommunication étaient en charge uniquement des lignes des clients publics. Par la suite (fin des années 80), un changement majeur a eu lieu : l'internet est devenu public et cela a rapidement changé le rôle des télécommunications. Au fur et à mesure que la technologie informatique avance, l'internet doit supporter une quantité de plus en plus grande d'informations et de données. Les lignes téléphoniques traditionnelles sont en voie de disparition, supplantées

par la "Voix sur Internet" (VoIP : *Voice Internet Protocol*). La télévision par internet (IPTV : *Internet Protocol Television*) fait partie de toute offre "triple play" des entreprises, offre qui inclut la ligne téléphonique, l'internet et la télévision. Les magasins "en ligne" sont en train de remplacer les magasins traditionnels. Les systèmes pair-à-pair, les magasins en ligne de musique et de vidéo deviennent plus populaires chaque jour et cela entraîne un usage accru de bande passante. De plus, le débit disponible offert par toutes les compagnies aux clients particuliers croît constamment. C'est ainsi que le débit moyen offert en France atteignait à peine 1Mb en 2003, alors qu'aujourd'hui le standard d'une offre intéressante est d'une vingtaine de Mb.

Deux modèles principaux décrivent les réseaux de télécommunication : le modèle TCP/IP (*Transmission Control Protocol/Internet Protocol*) et le modèle Référence OSI (*Open Systems Interconnection Basic Reference Model*). Le modèle TCP/IP a été créé en 1970 et divise le réseau en 5 couches : couche d'application, couche de transport, couche du réseau internet, couche des données et couche physique. Le modèle Référence OSI fut défini en 1979 et utilise 7 couches : couche d'application, couche de présentation, couche de session, couche de transport, couche du réseau, couche des données et couche physique. Les divers protocoles qui existent pour la gestion des informations passant par les réseaux utilisent l'un des deux modèles ci-dessus. De plus, il s'agit d'un système extrêmement hétérogène puisque les caractéristiques physiques des interconnections et les vitesses de transfert d'information varient énormément. Nous pourrions ajouter plusieurs caractéristiques du modèle ; cependant, notre but n'est pas de présenter explicitement ce genre d'informations mais de montrer la difficulté et la complexité de ce domaine.

Par ailleurs, une difficulté additionnelle est l'intense compétition du marché. Le rythme rapide d'évolution du domaine fait des télécommunications un secteur intéressant pour de nouveaux investissements. En France en particulier, nous avons un opérateur majeur (France Télécom) et plusieurs fournisseurs alternatifs (Neuf, Free, etc.). Ces derniers sont obligés de faire des offres très attirantes mais risquées aux clients particuliers afin de les inciter à abandonner leur opérateur majeur ; cela a comme résultat un marché très instable et empêche la prévision du nombre futur de clients et de leurs demandes respectives. Pour illustrer le risque important des offres des fournisseurs, il suffit de rappeler la bulle Internet qui a eu lieu en 1995-2001 concernant le marché des actions pour plusieurs compagnies internet.

Lorsqu'on prend en considération toutes ces informations, il est évident qu'on ne peut pas optimiser simultanément l'ensemble du secteur. En revanche, on peut décomposer le problème global en plusieurs sous-problèmes et étudier chacun localement. Parmi ces sous-problèmes, citons

- le dimensionnement des réseaux,
- l'optimisation de la qualité de service,
- la migration de la téléphonie classique au VoIP,
- l'aménagement des divers protocoles internet.

1.2 Introduction à la bibliographie

Pendant la première année de cette thèse, nous avons principalement étudié des manuels et articles pouvant être classés en 3 catégories :

Analyse convexe et optimization. Nos ouvrages principaux ont été ici ceux de Rockafellar [53] et Hiriart-Urruty & Lemaréchal [30]. On y trouve une étude des ensembles et fonctions convexes, continuité, fonctions conjuguées et dualité, ainsi que les éléments structuraux de la théorie différentielle, la théorie convexe et l'algèbre convexe. Outre ces notions, exposées d'une

manière relativement simple, [30] contient une vaste quantité d'informations concernant le domaine de l'optimisation : algorithmes d'optimisation, relaxation lagrangienne, théorie de la descente et méthodes de faisceaux sont explicitement examinées ; plusieurs méthodes basées sur ces prémisses sont prises en considération.

Optimisation robuste. Dédié plus particulièrement à l'optimisation robuste discrète, l'ouvrage [37] de Kouvelis & Yu présente différents problèmes d'application (par exemple les problèmes robustes du plus court chemin, de l'arbre couvrant minimal, du sac à dos) avec des contraintes de faisabilité indépendantes des scénarios. Certains de ces problèmes ont été prouvés solvables de manière polynomiale. De plus, un algorithme de substitution type "branch & bound", basé sur la relaxation, est présenté pour les problèmes avec scénarios uniques équivalents. Enfin, la location robuste 1-médiane, le planning robuste et le problème de dimensionnement robuste des réseaux sont discutés de façon plus détaillée.

Flots dans les réseaux. Dans [1], Ahuja, Magnanti et Orlin fournissent une introduction substantielle à ce domaine. D'abord, la théorie des graphes, le dimensionnement et l'analyse des algorithmes sont établis. Ensuite, les problèmes du plus court chemin, du flot maximal et du flot de coût minimal sont étudiés en profondeur et plusieurs autres champs sont également examinés ; notamment le problème d'arbre courvrant minimal, la relaxation lagrangienne, l'optimisation des réseaux et les flots multi-produit.

Les ouvrages ci-dessus nous ont fourni la base mathématique dont nous avions besoin pour notre recherche, en nous présentant les outils mathématiques de l'optimisation convexe et les algorithmes en théorie des graphes. Notre choix fut motivé par notre intention de travailler sur la construction des réseaux de télécommunication et sur l'optimisation de la qualité de service en cas d'incertitude sur les données (principalement la demande). Le but était de trouver des algorithmes permettant aux entreprises de télécommunication d'optimiser dans un futur proche les investissements qu'elles avaient faits, pour les clients particuliers mais aussi les compagnies plus grandes.

Dans une étape suivante, nous avons élargi notre recherche bibliographique pour inclure les modèles de programmation mathématique qui pourraient être utilisés pour nos propres modèles, ainsi que le travail qui avait été fait sur les problèmes des télécommunications.

Ces travaux vont maintenant être précisés et commentés dans les sections suivantes.

1.3 Optimisation robuste et stochastique

L'optimisation robuste et l'optimisation stochastique constituent deux secteurs majeurs de la programmation mathématique, fournissant les outils et méthodes qui peuvent être appliqués aux problèmes avec incertitude sur les données. Cette question est fréquente dans les problèmes d'optimisation pour les raisons suivantes :

- Les données sont inconnues lorsque la décision doit être prise, et elles ne se réalisent que dans le futur. Ces données peuvent représenter des demandes futures inconnues, des prix futurs, des localisations possibles etc.
- Les données ne peuvent pas être mesurées, estimées ou calculées au moment où la décision doit être prise ; par exemple les températures, la pression, etc.
- Les données sont certaines mais la décision optimale ne peut être correctement implémentée.

– De façon analogue, dans un programme linéaire, la matrice des contraintes et/ou les coefficients de la fonction-objectif sont incertains. On dira par exemple : “les données peuvent être multipliées par une matrice diagonale dont les coefficients prennent des valeurs entre 0.95 et 1.05”.

Les problèmes provenant des applications ont souvent un très grand nombre de solutions sous-optimales ; si, ignorant l'incertitude, on les résout en utilisant des données nominales, on finit par trouver un point sur la la frontière dudit ensemble de solutions (surtout avec le simplexe, qui fournit toujours un point extrême de l'ensemble réalisable). Un tel point peut être une très mauvaise solution d'un problème avec des données à peine perturbées. L'analyse de sensibilité, un outil utile de post-optimisation, est insuffisante dans cette situation : elle ne fait qu'analyser les propriétés de stabilité d'une solution déjà générée, pour une perturbation infinitésimale des données nominales. Autrement dit, elle décrit le mal mais ne propose pas de remède.

Nous nous sommes d'abord concentrés sur divers articles concernant l'optimisation robuste. Il s'agit d'un domaine en plein développement, ce qui présente des avantages et des désavantages. Pour résumer les avantages, le fait que ce domaine est nouveau nous permet d'innover et de contribuer à son développement, soit en proposant des méthodes nouvelles, soit en appliquant des méthodes actuellement au stade de la recherche, les améliorant et leur apportant de nouvelles idées ; soit encore en suivant des chemins de recherche qui ont été tracés par les chercheurs actuels du secteur. En ce qui concerne les désavantages, ils sont dus au fait que le secteur n'a pas encore pris sa forme finale et n'est pas encore défini par des limites spécifiques et claires ; ainsi, plusieurs résultats et méthodes devront-ils être par la suite validés et généralisés.

L'article pionner dans l'esprit de l'optimisation robuste a été écrit en 1973 par Soyster [58] et il concerne des problèmes de programmation mathématique incertaine avec contraintes qui doivent être satisfaites (donc pas de fonctions de pénalité, par exemple). Il traite des problèmes linéaires avec incertitude sur les colonnes et il préserve la robustesse pour toutes les réalisations possibles. Adoptant une stratégie pire cas, la méthodologie est trop conservatrice et conduit à des solutions exagérément dégradées par rapport à la solution nominale, correspondant aux données sans incertitude.

Une approche assez différente a été présentée par Mulvey et al. [44], où une solution est *robuste* si elle est approximativement optimale pour l'ensemble de tous les scenarios possibles, et *modèle-robuste* si elle est approximativement faisable pour ce même ensemble. Des pénalités – quadratiques et exactes – sont utilisées afin de permettre les infaisabilités qui vont inévitablement surgir dans le cas où l'on utilise des scénarios multiples pour les données d'entrée. Les auteurs généralisent aussi l'usage de la valeur moyenne dans la fonction objectif en ajoutant des moments d'ordre supérieur de la distribution correspondante. Les modèles robustes finaux sont relativement difficiles, comparés aux problèmes de programmation linéaire, mais ils calculent des solutions plus stables ayant un faible coût additionnel. Cet article se trouve donc à la frontière entre l'optimisation stochastique et robuste, utilisant le terme robuste pour caractériser la qualité de la solution, alors que le matériel théorique utilisé appartient au domaine de l'optimisation stochastique.

La percée décisive s'set produite vers la fin des années 90. Deux leaders de cette recherche, A. Ben-Tal et A. Nemirovski, ont joué un rôle fondamental pour l'établissement de la théorie de l'optimisation robuste. C'est ainsi qu'ils définissent en [7] le problème

$$(p_\zeta) \qquad \begin{array}{rl} \min\limits_{x\in R^n} & f(x,\zeta) \\ \text{s.t.} & F(x,\zeta) \in K \subset R^m; \end{array}$$

$\zeta \in R^M$ symbolise les données ; $x \in R^n$ est le vecteur de décision ; n, m, M, f, F et le cône convexe K sont les éléments structurels. Les auteurs étudient le cas où ζ appartient à un ensemble d'incertitude $U \in R^M$ et les contraintes $F(x, \zeta)$ doivent être satisfaites pour toute réalisation de ζ. Un vecteur x est appelé solution faisable du problème d'optimisation incertain $(P) = \{(p_\zeta)\}_{\zeta \in U}$ s'il satisfait les contraintes pour toutes les réalisations possibles : $F(x, \zeta) \in K$, pour tout $\zeta \in U$. Ils définissent également la "contrepartie robuste" (RC : *Robust Counterpart*) de (P), qui est le problème d'optimisation certaine :

$$(P^*) \quad \min \left\{ \sup_{\zeta \in U} f(x, \zeta) : \ F(x, \zeta) \in K, \, \forall \zeta \in U \right\}.$$

Sans perte de généralité, ils se concentrent sur le programme incertain

$$(P) = \left\{ (p_\zeta) : \min \left\{ c^T x : \ F(x, \zeta) \in K, \, x \in X \right\} \right\}_{\zeta \in U}$$

et à sa contrepartie robuste

$$(P^*) \quad \min \left\{ c^T x : \ x \in X, \, F(x, \zeta) \in K, \, \forall \zeta \in U \right\}.$$

Une solution faisable [resp. optimale] de (P^*) est appelée *robuste faisable* [resp. *robuste optimale*]. En utilisant ce modèle, ils étudient explicitement le cas d'une incertitude ellipsoïdale pour la programmation linéaire, quadratique, conique quadratique, semi définie et pour les programmes incertains dans lesquels l'incertitude intervient de facon affine.

Dans [9], des expériences sur des programmes linéaires de la collection `NETLIB` produisent des solutions fiables à coût acceptable, alors que les solutions des problèmes nominaux sont très peu fiables même pour des petites perturbations des données. Même quand la solution nominale était corrigée a posteriori, dans plusieurs cas elle ne pouvait pas être stabilisée.

La méthodologie de l'optimisation robuste est enrichie de la "Contrepartie Robuste Ajustable" (ARC : *Adjustable Robust Counterpart*), introduite dans [6] par les mêmes auteurs. Certaines variables de décision, dites *ajustables*, représentent des décisions pouvant être prises après que les données sont connues. En conséquence, l'ARC est plus flexible, moins conservatrice et représente mieux certains problèmes de la vie réelle (modèles multi-période, modèles dynamiques par exemple). Cependant, l'ARC est souvent impossible à résoudre. Les auteurs définissent donc la "Contrepartie Robuste Affinement Ajustable" (AARC), où les variables ajustables sont une combinaison affine des données incertaines. On montre alors qu'une incertitude simple ellipsoïdale est tractable ; et une bonne approximation permet de traiter le cas d'une intersection d'ellipsoïdes,

D'autres travaux on paru dans [8, 10].

L'ARC a été également étudiée par Takeda et alii, principalement sur des modèles à deux périodes avec incertitude discrète ou polytopique : [59]. On y montre que, si les variables (ajustables) de la deuxième période varient dans un ensemble satisfaisant une certaine propriété de quasi-convexité, et si la fonction objectif est quasi-convexe, alors l'ARC avec incertitude polyédrale se réduit à un problème d'optimisation à un seul niveau.

En même temps que Ben-Tal et Nemirovski, El Ghaoui et Lebret ont étudié dans [24] les problèmes de moindres carrés avec incertitude dans les données. Ayant déterminé un ensemble d'équations $Ax = b$, où $A \in \mathbb{R}^{m \times n}$, $x \in \mathbb{R}^n$ et $b \in \mathbb{R}^m$, étant donné une norme $\| \cdot \|$ et supposant que b est un vecteur incertain, on formule le problème de moindres carrés

$$\begin{array}{ll} \min\limits_{x, \Delta b} & \|\Delta b\| \\ \text{s.t.} & Ax = b + \Delta b. \end{array}$$

Si A est également incertaine, alors toute la matrice $[A|b]$ est incertaine et on définit de même

$$\begin{aligned} \min_{x,\Delta A,\Delta b} \quad & \|[\Delta A|\Delta b]\| \\ \text{s.t.} \quad & (A+\Delta A)\,x = b+\Delta b. \end{aligned}$$

L'article se concentre sur la robustesse déterministe de ces deux problèmes. Un vecteur $x \in \mathbb{R}^n$ est une solution robuste si elle minimise le *résidu pire-cas* défini par

$$r(A,b,\rho,x) = \max_{\|[\Delta A|\Delta b]\|\leq\rho} \| \,(A+\Delta A)\,x - (b+\Delta b)\,\|.$$

Une telle solution est calculée par des techniques de programmation sur le cône du second ordre et on montre qu'elle est continue par rapport aux données A et b. Quand les données incertaines sont définies par

$$A(\delta) = A_0 + \sum_{i=1}^{p} \delta_i A_i,\ A_0,\ldots,A_p \in \mathbb{R}^{m\times n}, \quad b(\delta) = b_0 + \sum_{i=1}^{p} \delta_i b_i,\ b_0,\ldots,b_p \in \mathbb{R}^m,$$

et le degré de perturbation est déterminé par $\|\delta\| \leq \rho$, le résidu pire-cas est optimisé par programmation semidéfinie. Dans le cas plus général où les données dépendent arbitrairement du paramètre d'incertitude δ, le problème est NP dur et on fournit une borne supérieure.

El Ghaoui, Oustry et Lebret dans [25] cherchent des solutions robustes pour les programmes semidéfinis. Etant donné un vecteur $c \in \mathbb{R}^n$ et des matrices symétriques $F_i = F_i^T \in \mathbb{R}^{n\times n}$, $i = 0,\ldots,m$, un programme semidéfini est écrit comme suit :

$$\begin{aligned} \min_{x} \quad & c^T x \\ \text{s.t.} \quad & F_0 + \sum_{i=1}^{m} x_i F_i \succeq 0. \end{aligned}$$

Si l'incertitude entachant les données n'est pas petite, l'analyse de sensibilité est insuffisante. Afin de modéliser l'incertitude les auteurs définissent :

- $x \in \mathbb{R}^n$: variable de décision
- $\Delta \in \mathbb{R}^{p\times q}$: perturbation appartenant à un sous espace linéaire donné $\mathbb{D}$ de $\mathbb{R}^{p\times q}$ et $\|\Delta\| \leq \rho$, où $\rho > 0$ est donné
- $F(x,\Delta) \in \mathbb{R}^{n\times n}$: matrice symétrique.

Afin d'établir les résultats théoriques, une représentation linéaire fractionnelle est utilisée pour le modèle de perturbation : plus précisément,

$$F(x,\Delta) = F(x) + L\Delta(I - D\Delta)^{-1}R(x) + R(x)^T(I - \Delta^T D^T)^{-1}\Delta^T L^T,$$

où $R(\cdot)$ est une application affine à dans $\mathbb{R}^{q\times n}$, $L \in \mathbb{R}^{n\times p}$ et $D \in \mathbb{R}^{q\times p}$. Les auteurs fournissent des conditions suffisantes pour l'existence des solutions robustes, qui – dans certains cas – ont été prouvées uniques et stables. Il est également noté que les conditions sont aussi nécessaires pour certaines structures de perturbation.

Il faut noter une similitude entre les formulations de Ben-Tal et Nemirovski [10] et de Soyster [58]. Elles trouvent un lien dans la nouvelle formulation proposée par Bertsimas et Sim dans [13]. Leur problème est linéaire et contient un paramètre vectoriel modulant l'augmentation du coût de la solution robuste en fonction de la probabilité de violation des contraintes du problème. On montre expérimentalement qu'avec un coût relativement faible sur la valeur optimale du problème

nominal, la solution peut être robuste avec une très forte probabilité. Ce modèle est également implémenté sur un problème discret – le sac à dos – et donne des résultats similaires. Des problèmes de programmation 0-1 avec fonction objectif incertaine sont étudiés dans [12]. Un algorithme est proposé, qui résout la contrepartie robuste de manière polynômiale si le problème nominal est lui-même polynômial.

Atamtürk [2] utilise la formulation de la contrepartie robuste introduite par Bertsimas et Sim [12, 13] et Ben-Tal et Nemirovski [8, 9] pour les problèmes de programmation 0-1. Il définit trois descriptions linéaires pour l'ensemble de faisabilité convexe qui correspond à la contrepartie robuste. Il étend ces formulations aux problèmes de programmation robuste mixte 0-1 et spécifie des relaxations LP puissantes.

Lewis définit dans [41] la "Régularisation Robuste" (*Robust Regularization*) d'une fonction réelle sur un espace euclidien E comme suit : soit C la boule unité centrée à l'origine pour une norme quelconque (par exemple la boule euclidienne) et $\epsilon \geq 0$ le *paramètre de régularisation*. Alors la régularisation robuste de la fonction $f : E \to [-\infty, +\infty]$ est

$$f_\epsilon(x) := \sup \{ f(y) : y - x \in \epsilon C \} .$$

Si f est localement lipschitzienne sur le complément d'un petit ensemble adéquat et satisfait une condition de croissance près de cet ensemble, alors f_ϵ est localement lipschitzienne pour ϵ suffisamment petit.

1.4 Pourquoi l'optimisation robuste ?

Ici nous devons préciser pourquoi nous avons décidé d'utiliser dans notre étude des modèles d'optimisation robuste et non pas d'optimisation stochastique. Le premier argument est le fait que l'optimisation robuste, comparée à l'optimisation stochastique, est plus efficace en termes de stabilité. Dans l'optimisation stochastique, ce qui est habituellement optimisé est la valeur moyenne d'une certaine fonction et ceci peut être un inconvénient majeur. En effet, le résultat final ne montre pas ce qui peut arriver dans un cas extrême et ceci peut avoir des répercussions catastrophiques en télécommunication, puisque le réseau doit être parfaitement fonctionnel pour toute demande future. Par contraste, dans l'optimisation robuste l'objectif est souvent la valeur absolue du pire cas, et ce résultat est plus sûr pour l'ensemble incertain que nous définissons, comparé au résultat du modèle stochastique.

Le deuxième argument mis en avant est le fait que l'optimisation stochastique est limitée au cas où l'incertitude est de nature stochastique. Ainsi, les distributions sous-jacentes de probabilité doivent être identifiées, et dans certains modèles la solution peut violer les contraintes liées à l'incertitude, avec telle pénalité ou telle probabilité. En conséquence, l'optimisation stochastique est mal adaptée pour les contraintes "dures".

Dans les deux cas, nous devons bien évidemment préciser la forme de l'incertitude posée qui joue un rôle fondamental pour le modèle développé comme nous allons le voir ci-dessous. De même dans chaque domaine, il existe des modèles plus sophistiqués et plus compliqués qui améliorent les inconvénients de chaque technique. Par exemple, dans le cas stochastique, la fonction objectif peut inclure aussi un facteur de déviation qui améliore la stabilité de la solution. Autre exemple : dans le modèle robuste, la fonction objectif peut être le modèle robuste relatif, ou le modèle de déviation minimum (voir [37, Sect.2.1] pour les définitions) qui ont en conséquence des solutions moins conservatrices et plus attirantes.

1.5 Divers outils de programmation mathématique

Outre les articles sur l'optimisation robuste, nous avons effectué des recherches bien précises sur les méthodes et modèles d'optimisation que nous voulions utiliser, pour en voir les domaines d'application, les avantages et inconvénients, et pour éventuellement trouver des idées nouvelles que nous pourrions leur incorporer. Nous avons étudié la relaxation langrangienne et diverses méthodes de plans sécants et d'optimisation convexes (Newton, gradient conjugué, faisceaux) et d'autres algorithmes plus spécialisés tels que ACCPM.

Plans sécants La méthode plans sécants de Kelley est présentée dans [33, 15]. Soit le problème de programmation convexe

$$\min_{x \in R} cx,$$

où $R = \{x : \ G(x) \leq 0,\ x \in S\}$, S est un polèdre convexe compact, et $G : S \to \mathbb{R}$ est (continue et) convexe. La méthode de Kelley génère une séquence $\{t_k\}_{k \in \mathbb{N}}$ qui converge vers un optimum. Soit $p(x;t)$ une fonction supportant $G(x)$ au point t (donnée par un sous-gradient de G en t). Alors t_k résout le programme linéaire

$$\min_{x \in S_k} cx,$$

où $S_0 = S$ et $S_k = S_{k-1} \cap \{x : \ p(x;t_{k-1}) \leq 0\}$. Par hypothèse, $p(x;t_k) \leq G(x)$ pour tout $x \in S$ et tout k, donc $S_k \supset R$. Si $t_k \in R$, le problème initial est résolu. Si $t_k \notin R$, la méthode ajoute le plan sécant $p(x;t_k) \leq 0$. Par ailleurs, l'article original de Kelley discute l'application de cette méthode à la programmation convexe en nombres entiers.

Plus grande sphère inscrite Elzinga et Moore proposent en [20] l'algorithme des "Plans Sécants Centrés" (CCPA : *Central Cutting Plane Algorithm*). Le point de départ est un résultat de Nemhauser et Widhelm [45] : étant donné un polyèdre compact non vide décrit par un ensemble d'inégalités affines

$$\alpha_j^T x \geq \beta_j,\ j = 1, \ldots, p,$$

le problème

$$\begin{array}{rl} \max & \sigma \\ \text{s.t.} & \alpha_j^T x - ||\alpha_j|| \sigma \geq \beta_j,\ j = 1, \ldots, p \end{array}$$

a pour solution optimale le rayon σ et le centre x de la plus grande sphère inscrite dans le polyèdre. Pour le problème

$$\begin{array}{rl} \max & c^T x \\ \text{s.t.} & g_i(x) \geq 0,\ j = 1, \ldots, m, \\ & x \in S, \end{array} \tag{1.1}$$

où S est un polyèdre convexe compact de $\mathbb{R}^n$ et g_i sont des fonctions concaves différentiables, l'algorithme est le suivant :

Plus grande sphère inscrite (Elzinga & Moore)

Etape 0 Définir SP_0 comme étant :

$$\begin{array}{rl} \max & \sigma \\ \text{s.t.} & c^T x - \sigma \geq \underline{f}, \\ & x \in S, \end{array}$$

où $\underline{f}$ est une borne inférieure de la valeur optimale dans (1.1). Poser $k = 1$.

Etape 1 Résoudre SP_{k-1} pour obtenir une solution (x^k, σ^k). Si $\sigma^k = 0$ stop : x^k est optimal.

Etape 2 Si x^k est faisable pour le problème initial (1.1), alors ajouter à SP_{k-1} la contrainte :

$$c^T x - \sigma \geq c^T x^k,$$

Sinon, ajouter les contraintes :

$$g_i(x_k) + \nabla g_i(x_k)(x - x_k) - ||g_i(x_k)||\sigma \geq 0,$$

pour tout i tel que $g_i(x_k) < 0$.

Etape 3 Ayant ainsi défini le problème SP_k, remplacer k par $k+1$ et aller à l'Etape 1.

En fait des règles permettent d'abandonner certaines coupes, réduisant ainsi le nombre total de contraintes dans SP_k. Cet algorithme a été utilisé avec succès pour des problèmes d'affecation robuste, voir [49].

Relaxation lagrangienne et méthodes de faisceaux Lemaréchal présente dans [39] la simplicité de la relaxation lagrangienne et sa capacité à résoudre une large variété de problèmes, ou du moins à en borner la valeur optimale. On trouve parmi les applications possibles :

- la programmation linéaire,
- la programmation quadratique, et les problèmes avec contraintes quadratiques,
- les problèmes d'optimisation combinatoire,
- les problèmes de grande taille,
- la maximisation d'entropie.

Il est également démontré que la génération de colonnes et la relaxation lagrangienne sont la même méthode vue sous deux angles différents. L'article contient une discussion sur la qualité des bornes, la possibilité d'obtenir une solution primale optimale à travers la procédure duale, une courte analyse des méthodes numériques de sous-gradient et leurs variantes (par exemple l'algorithme de l'ellipsoïde et le r algorithme de Shor), la méthode des plans sécants et la méthode du centre analytique ACCPM, incluant leurs motivations. Enfin les méthodes des faisceaux sont détaillées : leur schéma algorithmique, leur équivalent primal – le lagrangien augmenté – la justification théorique d'un maître-programme quadratique et quelques expériences numériques.

Un élément crucial dans la méthode des faisceaux est la procédure de mise à jour du paramètre de stabilisation. Une proposition intéressante a été donnée par Kiwiel dans [34], laquelle donne de bons résultats. Supposons le problème

$$\min_{x \in X} f(x).$$

Alors la méthode des faisceaux calcule une suite $\{x^k\}_{k=1,2,\ldots}$ qui converge vers le minimum de f et une suite de points $\{y^k\}_{k=1,2,\ldots}$ pour la génération des linéarisations de f. À l'itération k de l'algorithme, l'approximation polyédrale de f est

$$\hat{f}^k(x) = \max\left\{\bar{f}(x; y^i),\ i = 1, 2, \ldots, k\right\},$$

où $\bar{f}(x; y^i) = f(y_i) + g(y_i)(x - y^i)$ est la linéarisation de f en y^i. Le prochain point de linéarisation y^{k+1} est la solution optimale de

$$\min_{y \in X} \hat{f}^k(y) + \frac{u^k}{2}||y - x^k||^2 \tag{1.2}$$

(u^k étant le paramètre de stabilisation). Appelons $v^k = f(x^k) - \hat{f}^k(y^{k+1})$ la *décroissance espérée* de f. Si

$$f(y^{k+1}) \leq f(x^k) - m_L v^k, \quad m_L \in (0, 0.5),$$

alors le prochain *centre de stabilité* sera $x^{k+1} = y^{k+1}$, sinon il sera $x^{k+1} = x^k$. Quant à u^k, il est diminué si

$$f(y^{k+1}) \leq f(x^k) - m_R v^k, \quad m_R \in (m_L, 1), \tag{1.3}$$

et augmenté si :

$$\bar{f}(x^k; y^{k+1}) > \max\{\epsilon^{k+1}, -10v^k\}, \tag{1.4}$$

où ϵ^k est la variation estimée de

$$V^k = f(x^k) - \min\left\{f(x) : ||x - x^k|| \leq 1\right\}.$$

Noter que (1.3) implique que f et $\hat{f}$ sont voisins en y^{k+1}, donc u^k diminue afin de permettre au prochain x de varier plus ; par contraste, (1.4) indique que l'erreur de la nouvelle linéarisation $\bar{f}(\cdot, y^{k+1})$ est relativement large en x^k, donc u^k augmente afin d'obliger le prochain x à rester près du centre de stabilité et d'améliorer l'approximation de f. Pour plus de détails voir [34].

Algorithme de Falk et Soland Un algorithme de type "branch & bound" est présenté dans [22] pour maximiser une fonction séparable semi-continue supérieurement sur un ensemble fermé borné. Pour présenter brièvement cette méthode, nous devons d'abord donner quelques notations :

- Soit $f(x) = \sum_{i=1}^{N} f_i(x_i) : \mathbb{R}^N \to \mathbb{R}$, la fonction à maximiser et $X \subset \mathbb{R}^N$ l'ensemble de maximisation.
- Un *noeud* k est caractérisé par deux N-vecteurs l^k et L^k qui sont utilisés comme bornes inférieures et supérieures pour la variable x. Au noeud initial, nous commençons par

$$l^1 = \left\{l_1^1, \ldots, l_N^1\right\}, \quad L^1 = \left\{L_1^1, \ldots, L_N^1\right\},$$

  où l_i^1, L_i^1, $i = 1, \ldots, N$ sont respectivement des bornes inférieures et supérieures pour $x \in X$.
- En chaque noeud k, f est remplacée par son *enveloppe concave* g^k sur le pavé $\Pi^k = [l^k, L^k]$. La séparabilité de f rend g^k aisée à calculer :

$$g^k(x) = \sum_{i=1}^{N} g_i^k(x_i),$$

  où g_i^k est l'enveloppe concave de f_i sur $[l_i^k, L_i^k]$.

Le schéma général de l'algorithme est le suivant :

Branch & Bound (Falk & Soland)

Etape 0 Initialiser l'ensemble K des noeuds à $K = \{1\}$.

Etape 1 Calculer l'enveloppe concave g^k de f sur Π^k.

Etape 2 Maximiser $g^k(x)$ pour $x \in X \cap \Pi^k$; obtenir ainsi x^k et la valeur optimale μ^k.

Etape 3 Déterminer $\bar{k} \in K$ tel que $\mu^{\bar{k}}$ soit maximal.

Etape 4 Si $\mu^{\bar{k}} = f(x^{\bar{k}})$ alors stop : $x^{\bar{k}}$ maximise f sur X. Sinon couper $\Pi^{\bar{k}}$ en deux ; obtenir ainsi deux nouveaux noeuds enrichissant K. Brancher sur l'un d'eux et retourner à l'Etape 1.

Chaque fonction g^k majore f sur Π^k et l'ensemble des Π^k recouvre X, donc $\mu^{\bar{k}}$ est un majorant du maximum de f sur X. Si l'algorithme ne stoppe pas à l'Étape 4, il existe certainement i tel que $l_i^{\bar{k}} < x_i^{\bar{k}} < L_i^{\bar{k}}$, ce qui permet de brancher. L'Étape 2 est un problème facile si X est un polyèdre. En pratique, chaque f_i doit être convexe : alors son enveloppe concave est la fonction affine qui connecte les points

$$\left(l_i^k, f_i(l_i^k)\right) \text{ et } \left(L_i^k, f_i(L_i^k)\right).$$

La méthode converge sous deux règles de branchement différentes. Pour plus de détail voir [22].

Problèmes min-max-min Nous nous sommes aussi intéressés aux problèmes min-max-min ; problèmes difficiles, comme il est bien illustré par Demayanov, Demayanov & Malozemov dans [16]. Le résultat principal de cet article est le suivant : sont donnés $\Omega \subseteq \mathbb{R}^n$, $G_1 \subseteq \mathbb{R}^p$ et $G_2 \subseteq \mathbb{R}^q$; appelant T la famille de toutes les appplications τ de G_1 dans G_2, on doit minimiser par rapport à $x \in \Omega$

$$F(x) := \sup_{y \in G_1} \inf_{z \in G_2} \phi(x, y, z).$$

Alors

$$\inf_{x \in \Omega} F(x) = \inf_{\tau \in T} \inf_{x \in \Omega} \sup_{y \in G_1} \phi(x, y, \tau(y)).$$

Autrement dit, un problème arbitraire continu min-max-min peut être réduit à la solution de $|T|$ problèmes de type min-max. Néanmoins T peut être infini ; en conséquence cette approche est difficile à appliquer. Toutefois, elle peut être utile pour des problèmes discrets de min-max-min. Des méthodes numériques sont également données afin d'identifier des points stationnaires.

1.6 Réseaux de télécommunication

Cette section traite de divers sujets concernant les réseaux, et trois catégories peuvent y être distinguées :

- Réseaux privés virtuels (VPN).
- Dimensionnement des réseaux et incertitude.
- Robustesse dans les télécommunications.

Réseaux privés virtuels (VPN) Notre étude des réseaux a surtout concerné les réseaux privés virtuels (VPN : *Virtual Private Network*). Cela consiste à construire des réseaux privés à partir de l'infrastructure publique. Ainsi, une compagnie – une banque par exemple – souhaitant avoir son propre réseau sécurisé n'a pas à le construire : elle demande simplement un VPN à un fournisseur internet en précisant certains paramètres. Ces paramètres peuvent être :

- noeuds d'entrée et terminaux du VPN,
- débit minimum et maximum aux noeuds d'entrée et de sortie,
- qualité de service.

Cette configuration a une large variété d'applications, non seulement pour les grandes compagnies mais également pour les clients particuliers. Un modèle qui est souvent appliqué dans de tels cas est le *hose* model, pour lequel plusieurs études ont été effectuées. Le secteur se développe toujours rapidement et la majorité du travail utilise principalement la théorie des graphes.

Le modèle hose fut introduit en 1999 par Duffield et alii [17, 18]. Au lieu d'utiliser les demandes point à point, le modèle hose prend en considération uniquement le trafic d'entrée et de sortie pour

chaque noeud terminal du VPN, ce qui rend plus facile la définition des paramètres du réseau VPN, comparée à l'approche classique des demandes point-à-point. Les arbres de Steiner sont utilisés afin de connecter les noeuds d'entrée/sortie du VPN, et la capacité de réservation qui préexiste sur le réseau est réajustée dynamiquement.

Ce modèle innovateur fut suivi par plusieurs chercheurs et suscita de nombreuses études par la suite. Kumar et alii dans [38] proposent deux méthodes pour résoudre le problème de l'arbre VPN optimal : l'algorithme BFS (*Breadth-First Search*) et l'algorithme *primal-dual*. Ils comparent ces deux méthodes, en termes de qualité de la solution trouvée et de temps de calcul, avec l'arbre de Steiner donné dans [17]. L'algorithme BFS, développé initialement pour des débit d'entrée et de sortie symétriques, calcule l'arbre optimal en un temps polynômial. Dans le cas général, le problème est NP-dur et les deux algorithmes ont une meilleure performance que la méthode par arbre de Steiner.

Italiano, Leonard et Oriolo dans [31] s'intéressent au problème de la capacité de réservation dans un réseau afin de satisfaire des demandes appariées. Pour représenter l'incertitude de la demande, ils utilisent des noeuds terminaux et des bornes sur le trafic entre les terminaux, ce qui est très similaire à la méthode hose. La solution satisfait à toute matrice de demande et le vecteur des capacités forme un arbre. Il en résulte que les flux ne peuvent pas être découpés et, pour chaque matrice de trafic, le flux de chaque demande utilise le même chemin. Dans [32] Italiano et alii ont étudié théoriquement le problème du rétablissement rapide de la qualité de service sur un réseau VPN en cas de défaillance d'un lien. Ils font l'hypothèse que le VPN a une structure d'arbre et ils définissent un algorithme de 16-approximation, de complexité polynômiale.

Erlebach et Ruegg proposent dans [21] un algorithme qui calcule en temps polynômial le routage multi-chemin optimal pour les réseaux VPN sous le modèle hose. En théorie, cet algorithme utilise l'ellipsoïde (Kachiyan [1, 54, 60]) mais en pratique ce sont les plans sécants qui sont utilisés, ce qui peut conduire à des temps de calcul exponentiels. Cependant, les auteurs notent qu'un tel comportement est peu probable. Les résultats des expériences montrent que leur algorithme est considérablement plus rapide que la méthode de l'arbre optimal de [38]. Une variante de leur algorithme peut calculer le mono-routage optimal en forçant certaines variables réelles à être entières. Les auteurs montrent également qu'en général, un routage multi-chemin est considérablement plus économique en termes de capacité des liens que l'arbre et le mono-routage. Pourtant, plusieurs tests ont été faits dans le cas de demandes symétriques et le coût d'un routage-arbre n'a jamais été plus haut que le coût d'un routage multi-chemin. Il semble empiriquement avéré que, dans le cas de demandes symétriques, un routage-arbre optimal n'est jamais plus coûteux qu'un multi-chemin ; mais d'un point de vue théorique, le problème est ouvert.

Une étude approfondie des réseaux VPN sous le modèle hose a été conduite dans [57] par Shepherd et Winzer. Ils présentent une nouvelle approche qui combine les deux structures de flot suivantes :

VPN-arbre/VPN-hub. Le but de la méthode VPN-arbre est d'identifier l'arbre VPN de coût minimal capable de router toutes les demandes sous le schéma hose. La structure VPN-hub est calculée comme suit : pour chaque arbre T et chaque noeud $v \in T$, on calcule le coût d'envoi de chaque demande, à v d'une part, et à la destination correspondante d'autre part. La désignation de la capacité minimale pour tout arbre T et tout noeud $v \in T$ est le VPN-hub. Il est prouvé dans [28] que le VPN-arbre et le VPN-hub sont identiques.

Le Randomized Load Balancing (RLB) de Valiant. Dans ce cas les demandes sont routées en deux étapes : d'abord, chaque noeud distribue sa demande d'une manière équivalente et uniforme à tous les noeuds du réseau ; ensuite, les noeuds transfèrent chaque demande à sa destination finale.

En combinant ces deux méthodes, un nouvel algorithme est introduit : le Selective Randomized Load Balancing (SRLB). L'idée est de n'effectuer RLB que pour les hubs qui définissent un certain nombre de VPN-hubs plus courts. Cette méthode a de meilleurs résultats que RLB pour un nombre limité de noeuds ; son but est de garder les avantages de RLB (fiabilité, faible coût) et diminuer ses désavantages (retard du jitter, garanties de la qualité du service). L'article inclut plusieurs tests, insistant surtout sur l'architecture multi-hop – où le plus court chemin et la structure de flot d'arbre VPN sont utilisés – pour RLB et SRLB. Les tests comparent le coût d'architecture, l'utilisation des ressources, les avantages du multiplex et le premium de robustesse, qui est défini comme le rapport des coûts des liens quand toutes les matrices hose interviennent, par rapport au coût quand une seule matrice intervient.

Ben Ameur et Kerivin ont également travaillé sur les réseaux VPN sous le modèle hose. Dans [4] ils présentent le fonctionnement des réseaux VPN, leur routage et stratégie de prix ; ils donnent également une comparaison entre les modèles *pipeline* et hose. Dans [5] un modèle polyédrique est utilisé pour l'incertitude de la demande et ils introduisent leur formulation mathématique, partant d'une formulation arc-chemin. Une méthode des plans sécants est appliquée en créant un maître programme et deux procédures itératives qui identifient les inégalités non valides ; ces dernières utilisent les demandes non satisfaites et l'amélioration de la valeur de la fonction objectif, en sélectionnant des chemins différents. La solution finale précise la capacité de réservation, les chemins disponibles pour le routage des demandes et les coefficients de découpage des chemins qui sont indépendants des valeurs des demandes.

Dimensionnement des réseaux et incertitude Une approche stochastique pour le problème de dimensionnement de réseau avec incertitude de la demande est présentée dans [55]. Sen et alii adoptent une approche avec contrainte de budget. Ils formulent un programme stochastique linéaire à deux étapes qui est résolu par décomposition stochastique (SD : *Stochastic Decomposition* [29]). SD est une méthode d'optimisation basée sur l'échantillonnage, qui a certaines propriétés asymptotiques et un test d'arrêt motivé par des arguments statistiques. La première étape de la méthode minimise les demandes moyennes non satisfaites et détermine la capacité du graphe. La deuxième étape résout un problème multiflot qui détermine l'usage le plus efficace de la capacité. À chaque itération l'algorithme calcule une nouvelle capacité, une nouvelle demande aléatoire et une approximation linéaire (coupe) pour la valeur moyenne de la fonction objectif, d'après la capacité actuelle et les demandes générées antérieurement. Le modèle est validé par des simulations. Les auteurs notent également que, si les variables d'incertitude sont remplacées par leurs valeurs moyennes, le résultat est médiocre.

Lisser et alii présentent dans [42] une approche stochastique pour le problème de dimensionnement de réseau. L'incertitude de la demande est formulée par scénarios, à chacun desquels est attachée une probabilité de réalisation. Le problème est non différentiable et il est résolu par ACCPM (*Analytic Center Cutting Plane Method* [26]). La formulation s'y prêtant, les auteurs implémentent le calcul parallèle.

L'optimisation du dimensionnement ne peut pas être totalement déconnectée du *fonctionnement*, c'est-à-dire du problème de routage : pour évaluer une capacités données, il faut bien calculer

comment seront écoulées les demandes à travers le réseau. C'est pourquoi nous avons également étudié des articles sur ce dernier problème. En particulier, [50] par Ouorou, Mahey et Vial passe en revue les méthodes de résolution pour le cas non linéaire convexe. Diverses techniques tirent parti des diverses structures apparaissant dans le problème, sachant qu'une résolution "aveugle" avec les logiciels existants serait maladroite. Toutes ces techniques décomposent d'une façon ou d'une autre le problème non linéaire en sous-problèmes plus petits qui sont linéaires, ou convexes, ou se ramènent à des plus courts chemins. Là encore, le calcul parallèle peut être utilisé.

Une autre étude intéressante de Knippel et Lardeux [36] concerne le problème de construction de réseau multi-couche. Deux algorithmes sont présentés pour résoudre ce problème. Le premier est une méthode de décomposition semblable à Benders [11], basée sur les inégalités métriques [46] et restreinte au cône métrique de tous les vecteurs positifs satisfaisant les inégalités triangulaires. Le deuxième algorithme utilise principalement les coupes biparties, tirant parti du fait [56] que le "cut cône" et le cône métrique sont identiques pour un graphe n'ayant aucun sous-graphe contractable en K_5 (le graphe complet à 5 sommets).

Oriolo dans [48] traite également du problème de dimensionnement sous incertitude de la demande. Il définit la relation de *dominance* entre les matrices de trafic : la matrice D_1 domine la matrice D_2 lorsque toute capacité d'installation écoulée par D_1 peut être également écoulée par D_2. L'article inclut quelques théorèmes très intéressants concernant la dominance ; le plus significatif est que D_1 domine D_2 si et seulement si D_1 peut soutenir D_2 en tant que vecteur de capacités. Cela peut être très utile – dans le cas où l'incertitude peut être formulée via des scénarios de demandes – pour éliminer des scénarios et simplifier le problème. Cependant, les hypothèses de cet article sont restrictives ; par exemple, le réseau doit être complet et orienté, les scénarios ont tous une source commune unique avec plusieurs destinations.

Pour terminer cette sous-partie, signalons le travail de [27], où se trouvent des résultats théoriques de complexité concernant le problème du multiflot.

Robustesse dans les télécommunications Dans [49] Ouorou utilise trois modèles d'optimisation robuste introduits par Kouvelis et Yu [37] et propose une nouvelle approche pour le problème de dimensionnement de réseau sous incertitude de la demande. Les algorithmes de résolution sont de type plans sécants : étant convexes, les fonctions impliquées peuvent être exprimées comme enveloppes supérieures de leurs hyperplans d'appui. L'article compare les méthodes de Kelley et d'Elzinga-Moore (voir §1.5) ; Elzinga-Moore obtient de meilleurs résultats que Kelley, aussi bien en nombre d'itérations qu'en temps de calcul. L'incertitude est modélisée par des scénarios mais l'annexe inclut une approche théorique plus générale pour une incertitude appartenant à un nombre fini d'ensembles compacts.

Motivés par les applications en télécommunication, Karasan, Pinar et Yaman [19] étudient le problème du plus court chemin avec données dans un intervalle. Une formulation par programmation en variables mixtes est proposée, qui calcule le chemin de déviation robuste tel que défini en [37] ; une procédure est alors construite pour éliminer des arcs du graphe et ainsi réduire la taille du problème. Cette procédure utilise le concept d'arcs *faibles* et *non-faibles*, défini dans l'article, et sachant que les chemins de déviation robuste incluent uniquement des arcs faibles. Les résultats numériques établissent l'efficacité de cet algorithme. Une recherche similaire est effectuée dans [61] pour les problèmes d'arbre couvrant robuste avec données dans un intervalle. Après avoir défini les solutions robustes absolue et relative, les auteurs proposent un algorithme polynômial pour trouver la première et un programme en variables mixtes pour la deuxième. Ils implémentent également une procédure de prétraitement polynômiale qui simplifie le problème, utilisant les notions d'arcs

forts et *faibles*.

Ordonez et Zhao présentent dans [47] une étude sur l'extension de capacité robuste dans des réseaux avec incertitude sur les demandes, mais aussi les coûts. La robustesse est définie comme en [7] et des problèmes d'ARC tractables sont identifiés pour des incertitudes assez générales sur le coût ; quant a à la demande, on traite le cas d'intervalles d'incertitude indépendants. Les cas suivantss sont considérés :

- mutliflots ayant tous la même source,
- multiflots ayant tous la même paire source - destination.

Les auteurs notent que le cas d'une source unique a également une ARC tractable. Les expériences numériques ont montré une réduction de 20% pour le coût pire cas et, pour des données nominales, une perte d'optimalité de 5% par rapport à la solution optimale déterministe correspondante.

Geary et alii présentent dans [23] une approche de la robustesse en 5 étapes. Sachant que le trafic est incertain, ils fondent la construction du réseau sur une prédiction, de la façon suivante :

- Construire le réseau et formuler une prédiction de trafic.
- Router le trafic à coût minimal.
- Suivant la prédiction, produire un nouveau scénario de trafic en termes de volume, distribution et type de service transporté.
- Router le nouveau trafic et minimiser le nombre de longueurs d'onde utilisées.
- Consigner la quantité de trafic non routé.

Les étapes 3 à 5 sont exécutées jusqu'à ce que "suffisamment" de données soient collectées. Une analyse statistique est alors faite du pourcentage de trafic réel non satisfait, et des facteurs qui l'influencent.

1.7 Dimensionnement de réseaux

Pendant notre recherche bibliographique, nous avons rencontré une idée novatrice dans l'article [43] concernant la modélisation de la construction d'un réseau face à l'incertitude. Lucertini et Paletta utilisent un budget fixé b pour définir les vecteurs de réservation de capacité réalisables pour un réseau donné. L'ensemble X des vecteurs-capacités réalisables est donc l'ensemble des $x \in \mathbb{R}_+^E$ (E nombre d'arêtes) vérifiant $c^T x \leq b$; ici, c est le vecteur des coûts unitaires de capacité. La demande $\omega \in \mathbb{R}_+^K$ (K nombre de produits) est incertaine et peut se réaliser n'importe où dans un polyèdre Ω donné. L'investissement x étant décidé, appelons $\Gamma(x) \subset \mathbb{R}_+^K$ l'ensemble de tous les vecteurs-trafics pouvant transiter dans le réseau. Une réalisation ω de la demande et un trafic γ induisent alors un coût de défaillance $f(\omega, \gamma)$, qui est nul si $\gamma = \omega$ mais positif si $\gamma \not\geq \omega$. Supposons l'investissement x donné ; pour une réalisation ω, le meilleur γ minimise $f(\omega, \cdot)$ sur $\Gamma(x)$. Appelant $\phi_x(\omega) = \min_{\gamma \in \Gamma(x)} f(\omega, \gamma)$ la valeur optimale, la défaillance associée à x est le maximum de $\phi_x(\cdot)$ sur Ω. Le but est donc de minimiser cette défaillance, sous la contrainte de budget : il faut résoudre

$$\min_{x \in X} \max_{\omega \in \Omega} \min_{\gamma \in \Gamma(x)} f(\omega, \gamma). \tag{1.5}$$

Cependant, [43] ne propose pas réellement d'approche pour résoudre (1.5) – et encore moins des résultats numériques ; et de notre point de vue ce modèle pourrait être plus largement exploité. C'est là que notre contribution eut lieu.

De fait, (1.5) est un problème min-max-min, impossible à résoudre dans toute sa généralité. Nous avons procédé en deux étapes que nous exposons maintenant, et qui font l'objet respectivement des chapitres 2 et 3 ci-dessous.

Incertitude simplifiée Normalement, $\Gamma(x)$ est un polyèdre défini par des contraintes où x apparaît dans le second membre : le trafic satisfait aux équations de réseau et est limité par les capacités. Par ailleurs, la convexité de f est une hypothèse raisonnable : c'est le cas par exemple de $f(\omega, \gamma) = |\omega - \gamma|$, qui pénalise bien l'écart entre l'offre et la demande. Dans ces conditions, le minimum de $f(\omega, \cdot)$ est encore convexe par rapport à ω et son maximum, c'est à dire la défaillance

$$d(x) = \max_{\omega \in \Omega} \min_{\gamma \in \Gamma(x)} f(\omega, \gamma) \tag{1.6}$$

est une fonction convexe de x. Le sous-différentiel de d est donné par des résultats standards d'analyse convexe.

La défaillance $d(x)$ ne peut être calculée commodément que si Ω est donné comme l'enveloppe convexe d'un nombre raisonnables de scénarios ω^1, ..., ω^S : il suffit alors de minimiser $f(\omega^s, \cdot)$ pour $s = 1, \ldots, S$; $d(x)$ est alors la plus grande des valeurs obtenues. C'est cette approche qui a été développée dans notre article [52], reproduit au Chapitre 2 ci-dessous. Nous y développons la théorie ci-dessus, proposons des algorithmes de minimisation de d (Kelley, Elzinga & Moore, faisceaux) et donnons des résultats numériques, qui confirment le médiocre comportement de Kelley par rapport à ses concurrents.

Ces résultats illustrent également un point intéressant que nous résumons maintenant. La méthode de faisceaux résout à chaque itération un programme quadratique ; un logiciel spécialisé est normalement utilisé pour cela. Ici nous avons utilisé Cplex 9.0, ce qui a entraîné des temps de calcul élevés pour certaines instances de grande taille. Ceci contredit [14], où il est observé que le temps de calcul du QP est voisin du temps de calcul du LP correspondant (dans les méthodes de Kelley ou Elzinga & Moore). L'utilité d'un solveur quadratique *spécialisé* est ainsi confirmée.

Incertitude polytopique générale Lorsque Ω est défini par un ensemble de contraintes, on ne peut guère imaginger comment calculer $d(x)$ dans (1.6). Notre article [51], reproduit au Chapitre 3 ci-dessous, esquive la difficulté et propose des bornes inférieures et supérieures sur la valeur optimale de d.

Supposons tout d'abord un échantillonnage ω^1, ..., ω^S de points de Ω. D'après ce qui précède, on sait calculer et minimiser la fonction

$$d^S(x) = \max_{\omega \in \Omega^S} \min_{\gamma \in \Gamma(x)} f(\omega, \gamma).$$

où Ω^S est l'enveloppe convexe des ω^s. Comme d^S minore clairement d, cette minimisation fournit une borne inférieure de (1.5). Un processus itératif peut alors être construit si un nouveau scénario ω^{S+1} peut ensuite être généré. Pour cela nous proposons de résoudre un certain programme linéaire sur Ω, dont la fonction objectif est un sous-produit de la minimisation de d^S. Ce programme donne un $\omega^{S+1} \notin \Omega^S$, permettant ainsi d'itérer le processus, ou alors il reproduit l'un des ω^s déjà générés, et alors le processus stoppe. On ne peut malheureusement rien conclure quant à la qualité de la borne ainsi produite.

Admettant que le calcul exact de $d(x)$ est impossible, nous proposons également dans [51] des bornes supérieures, obtenues simplement en inversant le min et le max dans (1.6). Il est bien connu que $d(x) \leq \min_{\gamma \in \Gamma(x)} W(\gamma)$, où nous avons posé

$$W(\gamma) = \max_{\omega \in \Omega} f(\omega, \gamma). \tag{1.7}$$

Une borne supérieure sera donc obtenue si nous résolvons

$$\begin{array}{ll} \min\limits_{(x,\gamma)} & W(\gamma) \\ \text{s.t.} & \gamma \in \Gamma(x),\ x \in X. \end{array}$$

Là encore, W est une fonction convexe (c'est une enveloppe supérieure de fonctions convexes). Le problème ci-dessus peut donc être résolu par tout algorithme de la Section 1.5, à condition de savoir calculer la fonction $W(\gamma)$. Or (1.7) montre que cela consiste à maximiser une fonction convexe sur un polyèdre – problème difficile... Toutefois, une simplification par rapport à (1.5) est que, γ étant maintenant fixé, la fonction à maximiser est *explicite* et *séparable*. L'algorithme de Falk & Soland (fin de la Section 1.5) est donc bien adapté à la situation présente.

1.8 Optimisation de la qualité de service

L'article [40] reproduit au Chapitre 4 ci-dessous concerne un problème venant en aval du dimensionnement : ayant décidé des capacités (non nécessairement optimales) et connaissant un scénario de demandes, comment router ces demandes ? Plusieurs critères d'optimisation peuvent être considérés, nous nous intéressons ici à ce qui est appelé en télécommunications la qualité de service (QoS : *Quality of Service*). Lorsqu'une arête a supporte un flot y_a, cela induit une *congestion* $f_a(y_a)$, nulle en 0 et tendant généralement vers $+\infty$ lorsque y_a tend vers la capacité c_a de l'arête.

Une fonction de congestion communément utilisée est la fonction dite de Kleinrock

$$f_a(y_a) := \frac{y_a}{c_a - y_a},$$

définie pour $y_a \in [0, c_a[$. L'optimisation de la QoS est donc un problème non linéaire ; on parle aussi de muliflot non linéaire. S'il est couplé au problème du dimensionnement, il devra être résolu un grand nombre de fois au cours des itérations optimisant les capacités du graphe. Il convient donc de disposer de méthodes très performantes, en robustesse et rapidité de convergence. Une approche répondant à la question est la relaxation lagrangienne, qui décompose le problème en autant de sous-problèmes qu'il y a de demandes. L'essentiel du travail est alors la maximisation sans contraintes d'une fonction concave (la fonction duale), qui peut être notée

$$\Theta(u) = \Pi(u) + \Phi(u),$$

où $u \in \mathbb{R}^E$ si E est le nombre d'arêtes. Ici, Π est une fonction linéaire par morceaux (résultant de problèmes de plus court chemins paramétrés par u), alors que Φ est une fonction régulière (donnée explicitement par f). Pour maximiser Θ, nous développons un algorithme hybride qui approche Π par plans sécants (méthode de Kelley, §1.5) et utilise l'approximation de Newton pour Φ. Cet algorithme est très similaire à la méthode des faisceaux ; cependant, le terme stabilisateur quadratique, relativement artificiel dans la méthode des faisceaux, est remplacé par l'approximation quadratique de Φ, bien plus naturelle.

Nous définissons donc une variante de la méthode de faisceaux adaptée à la présent situation. Nous en établissons la convergence, et nous l'illustrons par des expériences numériques sur différents réseaux de télécommunication, dont certains sont réels et d'autres construits de manière artificielle. Il faut noter qu'une idée similaire est appliquée dans [3] à l'algorithme ACCPM, donnant des résultats spectaculaires. D'autre part, K.C. Kiwiel prolonge dans [35] notre travail, en définissant une variante un peu différente de la nôtre ; d'intenses expérimentations comparatives révèlent des performances de sa méthode au moins aussi satisfaisantes que celles de [3].

Bibliographie

[1] R.V. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows : Theory, Algorithms and Applications*. Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[2] A. Atamtürk. Strong formulations of robust mixed 0-1 programming. *Math. Program.*, 108(2-3) :235–250, September 2006.

[3] F. Babonneau and J.Ph. Vial. ACCPM with a nonlinear constraint and an active set strategy to solve nonlinear multicommodity flow problems. *Mathematical Programming, à paraître*, 2007.

[4] W. Ben-Ameur and H. Kerivin. New economical virtual private networks. *Communications of the ACM*, 46(6) :69–73, 2003.

[5] W. Ben-Ameur and H. Kerivin. Routing of uncertain traffic demands. *Optimization and engineering*, 3 :283–313, 2005.

[6] A. Ben-Tal, A. Goryashko, E. Guslitzer, and A. Nemirovski. Adjustable robust solutions of uncertain linear programs. *Mathematical Programming*, 99(2) :351–376, 2004.

[7] A. Ben-Tal and A. Nemirovski. Robust convex optimization. *Mathematics of Operations Research*, 23(4), 1998.

[8] A. Ben-Tal and A. Nemirovski. Robust solutions of uncertain linear programs. *Operation Research Letters*, 25 :1–13, 1999.

[9] A. Ben-Tal and A. Nemirovski. Robust solutions of linear programming problems contaminated with uncertain data. *Mathematical Programming*, 88 :411–424, 2000.

[10] A. Ben-Tal and A. Nemirovski. Robust optimization - methodology and appliacations. *Mathematical Programming*, 92 :453–480, 2002.

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

[12] D. Bertsimas and M. Sim. Robust discrete optimization and network flows. *Mathematical Programming*, 98 :49–71, 2003.

[13] D. Bertsimas and M. Sim. The price of robustness. *Operations Research*, 52(1) :35–53, 2004.

[14] O. Briant, C. Lemaréchal, Ph. Meurdesoif, S. Michel, N. Perrot, and F. Vanderbeck. Comparison of bundle and classical column generation. *Mathematical Programming*, 113(2) :299–344, June 2008.

[15] E. Cheney and A. Goldstein. Newton's method for convex programming and Tchebycheff approximations. *Numerische Mathematik*, 1 :253–268, 1959.

[16] A.V. Demyanov, V.F. Demyanov, and V.N. Malozemov. Minimaxmin problems revisited. *Optimization Methods and Software*, 17 :783–804, 2002.

[17] N.G. Duffield, P. Goyal, A. Greenberg, P. Mishra, K.K. Ramakrishnan, and J.E. van der Merwe. Resource management with hoses : Point-to-cloud services for virtual private networks. *IEEE/ACM Transactions on Networking*, 10(5) :679–692, October 2002.

[18] N.G. Duffield, P. Goyal, A.G. Greenberg, P.P. Mishra, K.K. Ramakrishnan, and J.E. van der Merive. A flexible model for resource management in virtual private networks. In *Proceedings of SIGCOMM '99*, pages 95–108, 1999.

[19] O. Ekin-Karaşan, M.Ç. Pinar, and H. Yaman. The robust shortest path problem with interval data, August 2001. `http://ie.bilkent.edu.tr/~mustafap/pubs`.

[20] J. Elzinga and T.J. Moore. A central cutting plane algorithm for the convex programming problem. *Mathematical Programming*, 8 :134–145, 1975.

[21] T. Erlebach and M. Ruegg. Optimal bandwidth reservation in hose-model VPNs with multi-path routing. In *Proceedings of INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 4, pages 2275–2282, March 2004.

[22] J.E. Falk and R.M. Soland. An algorithm for separable nonconvex programming problems. *Management Science*, 15(9) :550–569, May 1969.

[23] N. Geary, A. Antonopoulos, E. Drakopoulos, J.J. O'Reilly, and J.E. Mitchell. A framework for optical network planning under traffic uncertainty. In *Proceedings of 3rd International Workshop on the Design of Reliable Communication Networks (DRCN)*, Budapest Hungary, October 2001.

[24] L. El Ghaoui and H. Lebret. Robust solutions to least-squares problems with uncertain data. *SIAM Journal on Matrix Analysis and Applications*, 18(4) :1035–1064, 1997.

[25] L. El Ghaoui, F. Oustry, and H. Lebret. Robust solutions to uncertain semidefinite programs. *SIAM Journal on Optimization*, 9(1) :33–52, 1998.

[26] J.-L. Goffin, A. Haurie, and J.-Ph. Vial. Decomposition and nondifferentiable optimization with the projective algorithm. *Management Science*, 38(2) :284–302, 1992.

[27] O. Günlük. A new min-cut max-flow ratio for multicommodity flows. *SIAM Journal of Discrete Mathematics*, 21(1) :1–15, 2007.

[28] A. Gupta, J. Kleinberg, A. Kumar, R. Rastogi, and B. Yener. Provisioning a virtual private network : a network design problem for multicommodity flow. In *ACM Symposium on Theory of Computing (STOC'01)*, pages 389–398, 2001.

[29] J.L. Higle and S. Sen. Stochastic decomposition : An algorithm for two-stage linear programs with recourse. *Mathematics of Operations Research*, 16(3) :650–669, August 1991.

[30] J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms*. Springer-Verlag, Berlin, 1993.

[31] G.F. Italiano, S. Leonardi, and G. Oriolo. Design of networks in the hose model. In *Proceedings of 2nd International Workshop on Approximation and Randomization Algorithms in Communication Networks*, pages 65–76. Carleton Scientific Press, 2002.

[32] G.F. Italiano, R. Rastogi, and B. Yener. Restoration algorithms for virtual private networks in the hose model. In *Proceedings of 21st Annual Joint Conference of the IEEE Computer and Communications Societies*, New York, USA, 2002. IEEE INFOCOM 2002.

[33] J.E. Kelley. The cutting plane method for solving convex programs. *Journal of the SIAM*, 8 :703–712, 1960.

[34] K. Kiwiel. Proximity control in bundle methods for convex nondifferentiable minimization. *Mathematical Programming*, 46(1) :105–122, 1990.

[35] K. Kiwiel. An alternating linearization bundle method for convex optimization and nonlinear multicommodity flow problems. soumis à Mathematical Programming, 2007.

[36] A. Knippel and B. Lardeux. The multi-layered network design problem. *European Journal of Operational Research*, 127(1) :87–99, November 2007.

[37] P. Kouvelis and G. Yu. *Robust Discrete Optimization and Its Applications*. Kluwer Academic Publishers, 1997.

[38] A. Kumar, R. Rastogi, A. Silberschatz, and B. Yener. Algorithms for provisioning virtual private networks in the hose model. *IEEE/ACM Transactions on Networking*, 10(4) :565–578, 2002.

[39] C. Lemaréchal. Lagrangian relaxation. In *Proceedings of Computational Combinatorial Optimization, Optimal or Provably Near-Optimal Solutions [based on a Spring School]*, volume 2241, pages 112–156, London, UK, 2001. Springer-Verlag.

[40] C. Lemaréchal, A. Ouorou, and G. Petrou. A bundle-type algorithm for routing in telecommunication data networks. *Computational Optimization and Applications, à paraître*, 2008.

[41] A.S. Lewis. Robust regularization. Technical Report, Department of Mathematics, Simon Fraser University, BC V5A 1S6, Canada, September 2002. Research supported by NSERC.

[42] A. Lisser, A. Ouorou, J.-Ph. Vial, and J. Gondzio. Capacity planning under uncertain demand in telecommunication networks. Technical Report 99.13, Logilab, Department of Management Studies, University of Geneva, Switzerland, October 1999.

[43] M. Lucertini and G. Paletta. A class of network design problems with multiple demand : Model formulation and an algorithmic approach. *Mathematical Programming Study*, 26 :225–228, 1986.

[44] J.M. Mulvey, R.J. Vanderbei, and S.A. Zenios. Robust optimization of large-scale systems. *Operations Research*, 43(2) :264–281, March-April 1995.

[45] G.L. Nemhauser and W.B. Widhelm. A modified linear program for columnar methods in mathematical programming. *Operations Research*, 19 :1051–1060, 1971.

[46] K. Onaga and O. Kakusho. On feasibility conditions of multicommodity flows in networks. *IEEE Transactions on Circuit Theory*, 18(4) :425–429, 1971.

[47] F. Ordóñez and J. Zhao. Robust capacity expansion of network flows. *Networks*, 50(2) :136–145, 2007.

[48] G. Oriolo. Domination between traffic matrices. Technical Report, Centro Vito Volterra, Università di Roma "Tor Vergata", 2004.

[49] A. Ouorou. Robust capacity assignment in telecommunications. *Computational Managment Science*, 3(4) :285–305, 2006.

[50] A. Ouorou, P. Mahey, and J.-Ph. Vial. A survey of algorithms for convex multicommodity flow problems. *Managment Science*, 46(1) :126–427, 2000.

[51] G. Petrou, C. Lemaréchal, and A. Ouorou. Robust network design in telecommunications under polytope demand uncertainty. *under construction*, 2006.

[52] G. Petrou, C. Lemaréchal, and A. Ouorou. An approach to robust network design in telecommunications. *RAIRO Operations Research*, 41(4) :411–427, October-December 2007.

[53] R.T. Rockafellar. *Convex Analysis*. Princeton University Press, 1996.

[54] A. Schrijver. *Theory of Linear and Integer Programming*. Wiley, 1999.

[55] S. Sen, R.D. Doverspike, and S. Cosares. Network planning with random demand. *Telecommunications Systems*, 3(1) :11–30, 1994.

[56] P.D. Seymour. Matroids and multicommodity flows. *European Journal of Combinatorics*, 2 :257–290, 1981.

[57] F.B. Shepherd and P.J. Winzer. Selective randomized load balancing and mesh networks with changing demands. *Journal of Optical Networking*, 5 :320–339, 2006.

[58] A.L. Soyster. Convex programming with set-inclusive constraints and applications to inexact linear programming. *Operations Research*, 21 :1154–1157, 1973.

[59] A. Takeda, S. Taguchi, and R.H. Tutuncu. Adjustable robust optimization models for nonlinear multi-period optimization. Research Report 04-CNA-013, Department of Mathematical Sciences, Carnegie Mellon University, Pittsburgh, PA 15213, USA, August 2004.

[60] S.P. Tarasov, L.G. Khachiyan, and I.I. Erlikh. The method of inscribed ellipsoids. *Soviet Mathematics Doklady*, 37 :226–230, 1988.

[61] H. Yaman, O. Ekin-Karaşan, and M.Ç. Pinar. The robust spanning tree problem with interval data. *Operations Research Letters*, 29(1) :31–40, August 2001.

Chapitre 2

Dimensionnement Robuste I

Ce chapitre reproduit notre article [52], paru dans RAIRO.

Un des problèmes majeurs dans le domaine des télécommunications est de construire des réseaux robustes qui puissent faire face à l'incertitude de la demande. Ayant un réseau $G = (E, V)$, une capacité unitaire c_e pour chaque lien $e \in E$ du réseau et un budget b pour le problème d'allocation de la capacité, le but est d'identifier une capacité faisable qui minimise le pire cas de demande insatisfaite.

Nous formulons dans cet article l'incertitude de la demande comme un polytope engendré par un nombre fini de scénarios de la demande ; voir Section 1.7 ci-dessus, paragraphe "incertitude simplifiée". Nous montrons que le problème peut alors se ramener à la minimisation d'une fonction convexe (en fait linéaire par morceaux) sur un polyèdre. Nous calculons alors une solution optimale par trois méthodes de plans sécants : Kelley, Elzinga & Moore et faisceaux. Ces trois méthodes sont comparées en termes de temps de calcul et nombre d'itérations.

RAIRO Operations Research
Will be set by the publisher
DOI: (will be inserted later)

AN APPROACH TO ROBUST NETWORK DESIGN IN TELECOMMUNICATIONS

Georgios Petrou[1], Claude Lemaréchal[2] and Adam Ouorou[1]

Abstract. In telecommunications network design, one of the most frequent problems is to adjust the capacity on the links of the network in order to satisfy a set of requirements. In the past, these requirements were demands based on historical data and/or demographic predictions. Nowadays, because of new technology development and customer movement due to competitiveness, the demands present considerable variability. Thus, network robustness w.r.t demand uncertainty is now regarded as a major consideration. In this work, we propose a min-max-min formulation and a methodology to cope with this uncertainty. We model the uncertainty as the convex hull of certain scenarios and show that cutting plane methods can be applied to solve the underlying problems. We will compare Kelley, Elzinga-Moore and bundle methods.

Keywords. Telecommunications network design, robust optimization, min-max-min problems, cutting plane methods.

Mathematics Subject Classification.

1. Introduction

In the present competitive markets, robustness is regarded as a major consideration in telecommunication networks. The demand presents considerable variability because of customers movement, introduction of new services and product

Received February 9, 2006. Accepted February 26, 2007.

[1] France Télécom Division R&D, MCN-OTT, 38-40 rue du Général Leclerc, 92794 Issy-Les-Moulineaux Cedex 9, France; {georgios.petrou, adam.ouorou}@orange-ftgroup.com

[2] Inria, 655 avenue de l'Europe, Montbonnot, 38334 Saint Ismier, France; Claude.Lemarechal@inrialpes.fr

development. Historical data for predicting the future is useless since the area is constantly changing or because there is no history for new services. In this work, we study the problem of assigning capacity to the links of a network in order to satisfy a set of requirements. We consider that the demand is an uncertain parameter and propose a new formulation and a methodology to cope with this uncertainty. To model the uncertainty, the demand is assumed to belong to an uncertainty set and we seek for a capacity assignment of the links that is *good enough* for every possible demand in the uncertainty set. Thus, our approach fits in the framework of robust optimization as termed in [2, 16]. We analyze the case where the uncertainty set is the convex hull of given demand scenarios. Our approach can be viewed as worst-case oriented since we consider a min-max-min criterion which will result in conservative decisions, with the purpose of contributing to the learning process of decision makers by seeking solutions that hedge against demand uncertainty.

The paper is organized as follows. After reviewing the literature in Section 2, we present our min-max-min formulation for the general case in Section 3. In Section 4 we introduce the resolution procedures when the uncertainty set is described by the convex hull of a reasonable number of scenarios. The computational results are reported in Section 5, and Section 6 concludes this study.

2. Related works

Stochastic programming and robust optimization are mathematical tools to deal with uncertainty. They have been both used for telecommunication network design under uncertainty.

A stochastic approach for the network design problem with uncertain demand is presented in [23]. Sen *et al.* deal with the problem under the budget constrained approach. They formulate a two-stage stochastic linear program and they solve it with the use of *stochastic decomposition* [10]. This method is based on sampling which has asymptotic properties and a statistically motivated stopping rule. The first stage problem minimizes the expected unserved demands and provides the capacity of the graph, while the second stage problem strives for an efficient use of the capacity. The model is validated with the use of simulation, and it is also noted that the proposed solution is better than replacing the uncertain variables by their expected values.

Another stochastic optimization method is presented by Lisser *et al.* [19]. Scenario representation is used in order to formulate the uncertainty of the demand, and every scenario corresponds to a given probability for the possible realization. The resulting non-smooth problem is solved with the *analytic center cutting plane* method [9]. The subproblem is decomposed into multicommodity flow problems by demand scenarios. The authors take advantage of this formulation by using parallel computing.

Italiano *et al.* [12] are concerned with the problem of reserving capacity in a network in order to satisfy pairwise demands. To remove uncertainty of the

demands they use terminal nodes and bounds for the traffic between the terminals. This model is known as *hose model*. The solution has to satisfy any traffic matrix for the demand, and the capacity vector should form a tree. Hence, the flows are not split, and for any traffic matrix the flow of every demand uses the same path.

Duffield *et al.* [6] have studied a similar problem, but they concentrate on Virtual Private Networks (VPN) based on the hose model. Under this concept they take into account only the ingress and the egress traffic for each endpoint of the VPN, which is more easily specified compared to the conventional point-to-point approach, and Steiner trees are employed in order to connect the VPN endpoints. However, they consider the capacity assignment on a pre-existing network, which is dynamically resized.

Another study in the framework of the hose model is done by Kumar *et al.* [17]. They propose two new methods for solving the problem of the VPN tree computation: the *breath-first search* algorithm (BFS) and a primal-dual algorithm. They compare these two methods in terms of cost and computational time, with the Steiner tree which is given as solution in [6]. The BFS algorithm was developed initially for the case of symmetric ingress and egress bandwidths and it computes the optimal tree in polynomial time. In the general case the problem is NP-hard, and both algorithms outperform the Steiner tree method.

Ben-Ameur and Kerivin [1] consider VPN networks with a polyhedral model for the demand uncertainty. Their mathematical formulation is built on the arc-path flow formulation and a cutting plane method is devised to deal with the large size of the problem. There are two procedures which identify violated inequalities for the master problem. These inequalities are based on unsatisfied demands and improvement of the value of the objective function by selecting sequentially different paths. The final solution specifies a capacity vector, the supporting paths of the demands, and the splitting coefficients which are independent of the possible values of the demands.

In [22] Ouorou proposes three models for a robust capacity assignment in telecommunication networks with demand uncertainty in the framework of robust optimization as defined by Kouvelis and Yu [16]. The algorithmic solutions are based on cutting plane methods. Some computational experiments indicate that the Elzinga-Moore cutting plane method [7] can be a more valuable choice when compared with Kelley's method [13]. Since different possible uncertainty sets may exist in some circumstances, a generalization of these models is proposed in order to cope with a finite number of plausible uncertainty sets, and a weight is associated with each uncertainty set to determine its relative importance or worth.

3. Problem formulation

We follow the approach introduced in [20] by Lucertini and Paletta, who studied a similar problem in which the investment for installing the capacity is limited by a fixed budget. Suppose that we have a telecommunication network represented by a graph $G = (V, E)$ where V is the set of nodes (*e.g.* terminals) and E is

the set of edges (*e.g.* optical links). We also have a set K of pairs of nodes, which are defined as origin-destination pairs (OD-pairs). The commodities that have to be transferred between the OD-pairs are not given but they belong to an uncertainty set Ω. Let $\Gamma(x)$ be the set of all possible demand vectors which can be satisfied given a capacity vector x. A robust decision can be obtained by minimizing a function $\phi(\Omega, \Gamma(x))$ with respect to the capacity vector x, where ϕ is an appropriate measure of the set of unsatisfied demands. In this study we suppose that the feasible capacities are continuous, as in [23]. If $c \in \mathbb{R}_+^{|E|}$ is the cost vector, then

$$X := \left\{ x \in \mathbb{R}_+^{|E|} : \ c^T x \leq B \right\} \tag{1}$$

will denote the set of feasible capacities available for the network, under a given budget B. The problem we consider is as follows:

$$\min_{x \in X} \phi(\Omega, \Gamma(x)).$$

To define the measure ϕ, we introduce a penalty function $f(\omega, \gamma)$, which expresses the gap between a demand vector ω that the graph *has to* support and a demand vector γ that the graph *can* support. For any capacity vector $x \in X$, we set

$$\phi(\Omega, \Gamma(x)) := \max_{\omega \in \Omega} \min_{\gamma \in \Gamma(x)} f(\omega, \gamma)$$

and we formulate the problem as

$$\min_{x \in X} \phi(\Omega, \Gamma(x)), \quad \text{i.e.} \quad \min_{x \in X} \max_{\omega \in \Omega} \min_{\gamma \in \Gamma(x)} f(\omega, \gamma). \tag{2}$$

There are several ways to choose the function f. In this study, we consider that it is suitable to take as penalty function the weighted sum of unsatisfied commodities. So, if π_k is the penalty for not satisfying one unit of demand for the k^{th} OD-pair, then we set f to be

$$f(\omega, \gamma) := \sum_{k \in K} \pi_k (\omega_k - \gamma_k)^+,$$

where $(y)^+ := \max(0, y)$. As for $\Gamma(x)$, we introduce the following notation: P_k is the set of the possible paths considered to support the deman between the k^{th} OD-pair, $z_{k,p}$ is the flow of path $p \in P_k$, γ is the vector of demands which can be carried through the graph given the capacity vector x. Then,

$$\Gamma(x) = \left\{ \gamma \in \mathbb{R}_+^{|K|} : \exists z \in \mathbb{R}_+^{|P|}, \sum_{k \in K} \sum_{p \in P_k : j \in p} z_{k,p} \leq x_j, \ \forall j \in E, \sum_{p \in P_k} z_{k,p} = \gamma_k, \ \forall k \in K \right\}.$$

The first constraint implies that the total flow on every path does not exceed the corresponding capacity, and the second constraint indicates that the vector of demands γ is supported from the flow z.

Introducing $\delta_k := (\omega_k - \gamma_k)^+$ for every $k \in K$, problem (2) can be rewritten as:

$$\begin{array}{rl} \min\limits_{x\in X}\max\limits_{\omega\in\Omega}\min\limits_{\gamma,\delta,z} & \sum\limits_{k\in K}\pi_k\delta_k \\ \text{s.t.} & \sum\limits_{k\in K}\sum\limits_{p\in P_k:j\in p} z_{k,p} \le x_j,\ j\in E, \\ & \sum\limits_{p\in P_k} z_{k,p} = \gamma_k,\ k\in K, \\ & \delta_k \ge \omega_k - \gamma_k,\ k\in K, \\ & \gamma,\delta,z \ge 0, \end{array}$$

or in a compact form as

$$\begin{array}{rl} \min\limits_{x\in X}\max\limits_{\omega\in\Omega}\min\limits_{\gamma,\delta,z} & \pi^T\delta \\ \text{s.t.} & Az \le x, \\ & Fz = \gamma, \\ & \delta \ge \omega - \gamma, \\ & \gamma,\delta,z \ge 0, \end{array} \tag{3}$$

where A and F are suitable 0-1 matrices. This formulation is defined as *arc-path formulation.*

Remark. Since we assume that the supporting paths for the commodities are not restricted, we consider in our implementation the following equivalent formulation:

$$\begin{array}{rl} \min\limits_{x\in X}\max\limits_{\omega\in\Omega}\min\limits_{\gamma,\delta,X} & \sum\limits_{k\in K}\pi_k\delta_k \\ \text{s.t.} & \sum\limits_{k\in K}(X_j^{k+} + X_j^{k-}) \le x_j,\ j\in E, \\ & \mathcal{A}(X^{k+} - X^{k-}) = \gamma_k l_k,\ k\in K, \\ & \delta_k \ge \omega_k - \gamma_k,\ k\in K, \\ & \gamma,\delta,X \ge 0, \end{array}$$

where $\mathcal{A}$ is the node-arc incidence matrix of G, X^{k+} and X^{k-} are the "positive" and the "negative" flow vectors – respectively – which are used to satisfy commodity k, and l_k is the vector of $\mathbb{R}^{|V|}$ defined by

$$(l_k)_i = \begin{cases} -1, & \text{if } i \text{ is the } k\text{th origin node,} \\ 1, & \text{if } i \text{ is the } k\text{th destination node,} \\ 0, & \text{otherwise.} \end{cases}$$

This formulation is defined as *node-arc formulation.*

4. Solution methods

Problem (3) is difficult because of non-convexity (due to the internal min) and nonsmoothness (due to the intermediate max). We refer to [5] for a discussion

of these difficulties. However, an algorithm can be constructed for the solution of (3) when Ω is a bounded polyhedron with a limited number of extreme points. Thus we limit our study to the case where Ω is defined as the convex hull of $|S|$ scenarios $\omega^1, \ldots, \omega^{|S|}$ of possible demands; then the extreme points of Ω will be among these scenarios.

For fixed x and ω in (3), the inner linear minimization problem is compactly written as

$$\begin{array}{ll} \min\limits_{\delta,z} & \pi^T \delta \\ \text{s.t.} & Az \leq x, \\ & \delta \geq \omega - Fz, \\ & \delta, z \geq 0, \end{array}$$

whose dual problem is

$$\begin{array}{rll} P(x,\omega) := \max\limits_{u,v} & \omega^T u - x^T v & \\ \text{s.t.} & F^T u \leq A^T v, & \\ & u \leq \pi, & \\ & u, v \geq 0. & \end{array} \tag{4}$$

We have substituted Fz for γ and eliminated the resulting constraint $Fz \geq 0$ since z and F are positive by definition. By LP duality the above two programs have the same optimal value. Thus (3) is equivalent to solving

$$\min_{x \in X} \max_{\omega \in \Omega} P(x, \omega).$$

Note that $P(x,\omega)$ is easily computed for every (x,ω) through a linear maximization program.

Setting

$$Q(x) := \max_{\omega \in \Omega} P(x, \omega)$$

we have to minimize Q over X and the next result gives crucial properties of function Q.

Theorem 1. *Function Q is convex and we have*

$$Q(x) = \max_{i=1,2,\ldots,|S|} \mathcal{P}_i(x), \quad \textit{where} \quad \mathcal{P}_i(x) := P(x, \omega^i),\ i = 1, 2, \ldots, |S|. \tag{5}$$

Proof. Let $g_{u,v} : X \times \Omega \mapsto \mathbb{R}$ be the linear function $g_{u,v}(x,\omega) = u^T\omega - v^T x$, where

$$(x,\omega) \in (X,\Omega) \quad \text{and} \quad (u,v) \in U \times V = \big\{(u,v) :\ F^T u \leq A^T v,\ u \leq \pi,\ u, v \geq 0\big\}.$$

We have that $X \times \Omega \neq \emptyset$ for every $(u,v) \in U \times V$, so $g_{u,v}$ is well-defined. Moreover $g_{u,v}$ is convex as a linear function, and for an arbitary $(x',\omega') \in (X,\Omega)$ we have:

$$\max_{(u,v) \in U \times V} g_{u,v}(x',\omega') \leq \sum_{k \in K} \pi_k \omega'_k < +\infty.$$

So, if we rewrite P as

$$P = \max_{(u,v)\in U\times V} g_{u,v},$$

then we deduce that P is convex as the maximum of a family of convex functions (see [11], Prop. IV.2.1.2).

Now, the convex function $P(x,\cdot)$ attains its maximum at some extreme point of the set Ω (see [11], Prop. III.2.4.6), and these extreme points are among the $|S|$ possible scenarios. Hence:

$$\max_{\omega\in\Omega} P(x,\omega) = \max\left\{P(x,\omega^1), P(x,\omega^2), \ldots, P(x,\omega^{|S|})\right\}.$$

This establishes (5), and also shows that $Q(x) < +\infty$ for all $x \in X$. Thus Q is convex for the same reason as P. □

Then the convex nonsmooth problem

$$\min_{x\in X} Q(x) \tag{6}$$

is equivalent to (3).

We will analyze three cutting plane algorithms for the solution of (6). The basic idea underlying these algorithms can be described as follows. Having computed t sets of $|S|$ values $\mathcal{P}_i(x_k)$, $i = 1,\ldots,|S|$ and corresponding subgradients $g_{ki} = -v_{ki}$ (see Th. 2 below) for $i = 1,\ldots,|S|$ and $k = 1,\ldots,t$, the true function Q is approximated (from below) by the polyhedral function

$$x \mapsto \max\{\mathcal{P}_i(x_k) + g_{ki}^T(x - x_k) : i = 1,\ldots,|S|,\ k = 1,\ldots,t\}. \tag{7}$$

A cutting plane method uses this function to select a new point x_{t+1}, so as to improve the current approximation (7). The way this point is chosen determines whether and how fast the algorithm converges.

Theorem 2. *For fixed $x_0 \in X$ and $i \in \{1,\ldots,|S|\}$, let (u_{0i}, v_{0i}) be an optimal solution of $\mathcal{P}_i(x_0)$. Then $-v_{0i} \in \partial\mathcal{P}_i(x_0)$.*

Proof. The theorem results from elementary convex calculus, but a simple direct proof can be given:

If (u_{0i}, v_{0i}) is an optimal solution of the mathematical programming problem $P(x_0,\omega^i)$ of (4), then $\mathcal{P}_i(x_0) = u_{0i}^T\omega^i - v_{0i}^T x_0$. Let (u_{si}, v_{si}) be the optimal solution of $\mathcal{P}_i(x)$ for an arbitrary $x \in X$. Since the constraints of (4) do not depend on x, (u_{0i}, v_{0i}) is feasible for problem $P(x,\omega^i)$ and

$$\begin{aligned}\mathcal{P}_i(x) = u_{si}^T\omega^i - v_{si}^T x &\geq u_{0i}^T\omega^i - v_{0i}^T x \\ &= \mathcal{P}_i(x_0) - v_{0i}^T(x - x_0), \quad \forall\, x, x_0 \in X,\ i \in \{1,\ldots,|S|\}.\end{aligned}$$

Hence $-v_{0i} \in \partial\mathcal{P}_i(x_0)$. □

We consider and compare the three following cutting plane methods, which differ in the way they construct the trial points x_k. They all use an *oracle* which,

at a given x, solves (4) $|S|$ times to compute the $\mathcal{P}_i(x)$'s and then $Q(x)$, as well as the optimal v_i's to provide the subgradients stipulated in Theorem 2. The optimal v_i's computed at $x = x_k$ will be denoted by v_{ki} for $i = 1, \ldots, |S|$.

4.1. Kelley cutting plane method

This is one of the first cutting plane methods proposed in the literature. The next trial point x_{t+1} is obtained by solving the following relaxed problem (Master Problem) of (6):

$$\begin{array}{ll} \min\limits_{x,\mathcal{Z}} & \mathcal{Z} \\ \text{s.t.} & \mathcal{Z} \geq \mathcal{P}_i(x_k) - v_{ki}^T(x - x_k),\ i = 1, 2, \ldots, |S|,\ k = 0, 1, \ldots, t, \\ & c^T x \leq B, \\ & x \geq 0. \end{array} \tag{8}$$

The method iterates until a "good" approximation of the solution is found, and the resulting algorithm can be stated as follows:

Kelley Algorithm (KA)

Step 0 Initialize $x_0 = 0$, $\varepsilon > 0$, $\overline{\mathcal{Z}} = +\infty$ and $t = 0$.
Step 1 Call the oracle at x_t to obtain $\mathcal{P}_i(x_t)$ and v_{ti} for $i = 1, \ldots, |S|$, and $Q(x_t)$.
Step 2 Let $\overline{\mathcal{Z}} = \min\left\{\overline{\mathcal{Z}},\, Q(x_t)\right\}$.
Step 3 Solve (8) and let $(x_{t+1}, \underline{\mathcal{Z}})$ be its optimal solution.
Step 4 If $\overline{\mathcal{Z}} - \underline{\mathcal{Z}} \leq \varepsilon\left(1 + |\overline{\mathcal{Z}}|\right)$, then stop. Otherwise set $t = t + 1$ and loop to Step 1.

This procedure converges, and the proof is given in [13]. Note that X is bounded, so (8) has always an optimal solution. This substantially simplifies the issue (see [11], Th. XII.4.2.3). However, if convergence is slow, then the Master Problem can be uncomfortably large. For further discussion see Sections 5 and 6.

4.2. Elzinga-Moore cutting plane method

This method is based on the following result by Nemhauser and Widhelm [21].

Theorem 3. *Given a bounded, non-empty polyhedron described from a set of linear inequalities,*

$$\alpha_j^T x \geq \beta_j,\ j = 1, \ldots, p,$$

the optimal solutions in σ and x of the problem

$$\begin{array}{ll} \max\limits_{\sigma,x} & \sigma \\ \textit{s.t.} & \alpha_j^T x - ||\alpha_j||\sigma \geq \beta_j,\ j = 1, \ldots, p, \end{array}$$

are respectively the radius and the center of the largest sphere inscribed in the polyhedron.

Proof. See [21]. □

Naturally, this result has a meaning only for a bounded polyhedron. Now consider in the $(x, \mathcal{Z})$-space the polyhedron defined by the constraints in (8) – *i.e.* the epigraph of the current approximation (7). It is unbounded but it can be truncated from above by appending the constraint $\mathcal{Z} \leq \overline{\mathcal{Z}}$. The polyhedron thus obtained clearly contains any optimal solution $(x^*, Q(x^*))$ of (6): the center revealed by Theorem 3 may be deemed an adequate approximation of such an optimal solution. This is the rationale for Elzinga-Moore cutting plane method, which computes the next trial point by solving

$$\begin{array}{ll} \max\limits_{x,\sigma,\mathcal{Z}} & \sigma \\ \text{s.t.} & \mathcal{Z} + \sigma \leq \overline{\mathcal{Z}} \\ & \mathcal{P}_i(x_k) - v_{ki}^T(x - x_k) - \mathcal{Z} + (||v_{ki}||^2 + 1)^{\frac{1}{2}}\sigma \leq 0, \\ & \quad i = 1, 2, \ldots, |S|,\ k = 0, 1, \ldots, t, \\ & c^T x \leq B, \\ & x \geq 0. \end{array} \tag{9}$$

Note again that this linear program has a non-empty bounded feasible set, and therefore an optimal solution $(x_{t+1}, \sigma_{t+1}, \mathcal{Z}_{t+1})$ (with $\sigma_{t+1} > 0$). This method has little usage in the literature even though it is not much harder to be implemented than Kelley's method. It was shown to be more efficient than Kelley's for some min-max problems, see [22]. Elzinga-Moore cutting plane algorithm for the solution of (6) is as follows:

Elzinga-Moore Algorithm (EMA)

Step 0 Initialize $x_0 = 0$, $\varepsilon > 0$, $\overline{\mathcal{Z}} = +\infty$ and $t = 0$.
Step 1 Call the oracle at x_t to obtain $\mathcal{P}_i(x_t)$ and v_{ti} for $i = 1, \ldots, |S|$, and $Q(x_t)$.
Step 2 Let $\overline{\mathcal{Z}} = \min\left\{\overline{\mathcal{Z}},\, Q(x_t)\right\}$.
Step 3 Solve (9) to get x_{t+1} and σ_{t+1}.
Step 4 If $\sigma_{t+1} \leq \varepsilon$, then stop. Otherwise set $t = t + 1$ and loop to Step 1.

For further details about this procedure and the proof of its convergence, see [7].

4.3. Bundle method

Kelley cutting plane method is known to be unstable. Bundle methods [11, 14, 18] aim at overcoming this instability by computing the Moreau-Yosida regularization of the piecewise linear approximation of (7). More precisely, the master problem to consider is obtained by adding a quadratic term to (8) as follows:

$$\begin{array}{ll} \min\limits_{x,\mathcal{Z}} & \mathcal{Z} + \dfrac{\mu_t}{2}\|x - \widehat{x}_t\|^2 \\ \text{s.t.} & \mathcal{Z} \geq \mathcal{P}_i(x_k) - v_{ki}^T(x - x_k),\ i = 1, 2, \ldots, |S|,\ k = 0, 1, \ldots, t, \\ & c^T x \leq B, \\ & x \geq 0, \end{array} \tag{10}$$

where $\widehat{x}_t$ is the *stability center*: a point which is known to be relatively good; μ_t is a positive parameter which controls the tradeoff between minimizing $\mathcal{Z}$ and staying close to $\widehat{x}_t$. The stability center is updated if x_{t+1} is significantly better than $\widehat{x}_t$ in the sense that

$$Q(x_{t+1}) \leq Q(\widehat{x}_t) - \kappa(Q(\widehat{x}_t) - \mathcal{Z}_{t+1}), \tag{11}$$

where $(x_{t+1}, \mathcal{Z}_{t+1})$ is the solution of (10), and $\kappa \in]0, 0.5[$. If (11) holds, then we have a *descent step* and set $\widehat{x}_{t+1} = x_{t+1}$, otherwise we have a *null step* and the stability center is left as it is. The resulting cutting planes are added to (10) in both cases. The bundle algorithm (BA) stops when

$$Q(\widehat{x}_t) - \mathcal{Z}_{t+1} \leq \varepsilon(1 + |Q(\widehat{x}_t)|), \tag{12}$$

where ε is the desired accuracy.

Bundle Algorithm (BA)

Step 0 Initialize $x_0 = \widehat{x}_0 = 0$, $\varepsilon > 0$, $\kappa \in]0, 0.5[$ and $t = 0$. Call the oracle at x_0 to obtain $\mathcal{P}_i(x_0)$ and v_{0i} for $i = 1, \ldots, |S|$, and $Q(x_0)$.

Step 1 Solve (10) and let $(x_{t+1}, \mathcal{Z}_{t+1})$ be its optimal solution.

Step 2 Call the oracle at x_{t+1} to obtain $\mathcal{P}_i(x_{t+1})$ and $v_{t+1,i}$ for $i = 1, \ldots, |S|$, and $Q(x_{t+1})$.

Step 3 If (11) is true then set $\widehat{x}_{t+1} = x_{t+1}$, otherwise set $\widehat{x}_{t+1} = \widehat{x}_t$.

Step 4 Compute μ_{t+1}.

Step 5 If (12) is true then stop. Otherwise set $t = t + 1$ and loop to Step 1.

Remark. Step 3 is defined as *weight updating* and it is crucial for the speed of convergence of the algorithm. We use a procedure described in [14], where the weight μ_t is decreased if the approximation of Q is close to Q at x_{t+1}, and it is increased if the errors of the new linearizations are greater than a variation estimate of $Q(\widehat{x}_t)$. See [14] for further details.

5. Results and numerical experiments

We consider the above three methods for the following reasons. Kelley's method is one of the most methods used in nonsmooth optimization while Elzinga-Moore algorithm has been, to the best of our knowledge, used rarely even if it is not much more hard to implement than the former. It outperformed Kelley's method on some min-max problems arizing in robust capacity planning problems, see [22]. Our first aim was to confirm this efficiency. We also consider the proximal bundle method because it is known as one of the best methods for nonsmooth optimization and we think it is extremely informative to compare it with the two above methods. Its subproblem is a quadratic problem which is considered to more difficult than the linear subproblems of Kelley and Elzinga-Moore methods.

We have written an experimental code based on the above development, under Eclipse Platform 2.1.1; we have used Java 1.4.2 and ILOG Cplex 9.0 for the solution

TABLE 1. Graphs specifications.

	Nodes	Edges	OD-pairs	Scenarios
Graph 1	12	25	50	21
Graph 2	19	34	50	21
Graph 3	26	30	100	21
Graph 4	16	49	89	10
Graph 5	26	53	100	21
Graph 6	60	40	140	11

of the subproblems (4), (8), (9) and (10). All the runs are performed on a two-processor Xeon Intel server of 2.4 GHz CPU speed and 1.5 GB of RAM memory, running under Linux.

The networks of our experiments derive from actual networks with given nominal demands. Their sizes are given in Table 1, as well as the numbers of OD-pairs and the number $|S|$ of scenarios considered in each case. Those scenarios have been generated in two ways as follows:

(1) From the available nominal vector of demands ω and a positive parameter ρ, we have generated random $\epsilon_k^s \in [-\rho, \rho]$ for $s = 1, \ldots, |S|$ and $k = 1, \ldots, |K|$ and set

$$\omega_k^s = \epsilon_k^s \omega_k, \quad k = 1, 2, \ldots, |K|.$$

(2) From the nominal vector of demands ω we have generated random $\epsilon_k^s \in [0, 1]$ for $s = 1, \ldots, |S|$ and $k = 1, \ldots, |K|$, and set

$$\omega_k^s = \epsilon_k^s \frac{\sum_j \omega_j}{\sum_j \epsilon_j^s}, \quad k = 1, 2, \ldots, |K|.$$

The first case results in scenarios that are variations around the nominal demand, and we use it for graphs 1, 2, 3, 4 and 6. In the second case the sum of the demands is the same for every scenario, and we use it for graphs 5 and 6. Finally, the nominal demand is included in the set of scenarios.

In Table 2 we report the results obtained by the three algorithms presented in Section 4. The column headed "Budget" specifies the parameter B in each test problem, see (1). This budget is derived arbitrarily, having as indicators the solutions of the multicommodity flow problems, formed by every graph and its corresponding scenarios. Different values for the budget are tested, but it is always requested that finally at least one objective value equals to 0 (zero) and various positive solutions for every graph are acquired. Then, for every method we report the value of the objective function, the number of oracle calls and the CPU time in seconds. Graph 6 appears twice because both scenario generation methods 1 and 2 were used for it (the notations 6a and 6b correspond to the first and second method respectively).

TABLE 2. Results obtained by the different algorithms.

	Budget	KA			EMA			BA		
		obj. value	# oracle calls	CPU time	obj. value	# oracle calls	CPU time	obj. value	# oracle calls	CPU time
Graph 1	612.3	63.33	90	18.45	63.33	85	23.63	63.33	16	5.21
	614.0	44.44	94	21.47	44.44	79	20.07	44.44	15	4.21
	616.0	22.22	85	17.26	22.22	71	17.23	22.22	15	4.45
	618.0	0.00	94	22.12	0.00	75	16.23	0.00	14	3.67
Graph 2	190.0	94.44	189	239.33	94.44	120	159.69	94.44	17	35.24
	194.0	50.00	189	177.94	50.00	113	114.14	50.00	17	29.26
	197.0	16.67	180	151.40	16.67	95	84.29	16.67	20	35.31
	199.0	0.00	165	159.94	0.00	87	68.72	0.00	13	18.26
Graph 3	12165.0	97.22	179	280.11	97.25	138	221.64	97.22	39	93.48
	12175.0	41.67	181	274.68	41.67	144	240.34	41.67	41	103.16
	12182.0	2.78	202	307.01	2.79	137	222.82	2.78	47	112.26
	12187.0	0.00	173	269.13	0.00	128	201.06	0.00	43	102.38
Graph 4	7380.0	389.76	265	2049.12	389.77	167	1167.71	389.76	35	289.49
	7395.0	202.26	181	1224.03	202.28	161	1120.62	202.26	39	335.48
	7408.0	39.76	172	1201.59	39.78	160	1038.69	39.76	36	311.50
	7415.0	0.00	244	2244.69	0.00	130	1025.08	0.00	38	303.00
Graph 5	8150.0	1930.00	687	15131.10	1930.01	664	16826.14	1930.00	60	2266.53
	8300.0	612.50	479	8628.33	612.51	288	5238.21	612.50	49	922.21
	8375.0	46.43	425	7437.13	46.43	257	3785.32	46.43	52	815.31
	8390.0	0.00	281	3180.40	0.00	254	3882.62	0.00	37	506.80
Graph 6a	40000.0	152.91	1176	22616.64	152.91	581	14216.79	152.91	216	16438.04
	40400.0	82.65	1111	23615.07	82.65	533	11110.81	82.65	198	13079.14
	40800.0	17.10	1064	19324.76	17.11	541	10882.63	17.10	167	9633.43
	40900.0	2.13	1155	23065.01	2.14	562	9485.44	2.13	160	9878.38
	41000.0	0.00	1157	22113.83	0.00	525	7134.68	0.00	144	8663.29
Graph 6b	45000.0	121.38	659	24628.16	121.38	325	8103.27	121.38	125	13348.54
	45600.0	39.44	633	19361.97	39.44	345	7021.62	39.44	93	8893.14
	45800.0	13.68	643	24012.30	13.68	355	7260.20	13.68	106	9383.37
	45900.0	0.87	720	25288.75	0.87	371	12712.30	0.87	95	8986.39
	45920.0	0.00	609	16217.29	0.00	330	7134.05	0.00	100	9657.57

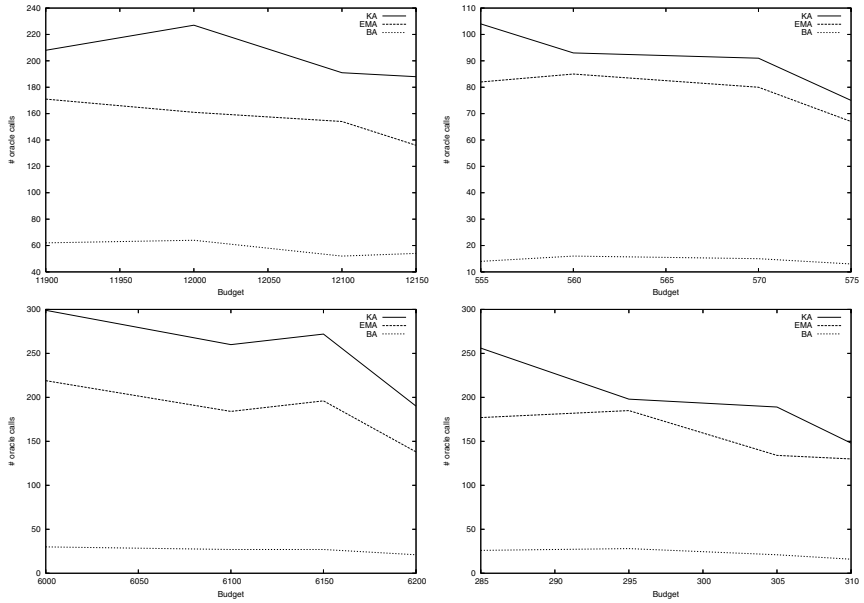

FIGURE 1. Illustration of the number of oracle calls for the first 4 graphs.

It appears from these results that, in terms of number of calls to the oracle, the bundle algorithm outperforms the other two methods: it divides the number of Kelley's [resp. Elzinga-Moore] oracle calls by a factor of 4.2 to 12.7 [resp. 2.6 to 11.1]. However, this reduction is not reflected in the CPU times: in particular for large problems, the bundle algorithm can become substantially slower than Elzinga-Moore. The reason is that the quadratic subproblem in our implementation becomes time consuming when the sizes of the problem and of the bundle increase. This behaviour contradicts the experiments of [4], which clearly show that quadratic programming is hardly more expensive than linear programming, and thus demonstrates the usefulness of specialized algorithms such as [8, 15]. On the other hand, Elzinga-Moore method behaves efficiently when compared to Kelley's, as reported in [22]. We hope that the readers will be encouraged to test this method when only a linear solver is at their disposal. Figure 1 illustrates in another way the above observations, where we plot the number of oracle calls vs different budget for the first four graphs using other 20 traffic scenarios in each case.

Using the first four graphs, we conduct some numerical experiments with the proximal bundle algorithm, to analyse the model in terms of number of scenarios and budget. The number of scenarios to be considered in the model is itself a difficult issue which must be considered at a first stage before applying the model. Interesting studies about the demand modeling exist in the literature, (see for instance [3]) and can be used to simulate the demand and provide the scenarios.

TABLE 3. Results with different number of scenarios.

	Budget	obj. value	# oracle calls	CPU time
Graph 1	5 scenarios			
	535	234.54	22	2.79
	540	156.21	20	2.81
	550	13.36	19	2.46
	555	0.00	14	1.92
	10 scenarios			
	550	217.79	17	4.65
	555	154.37	15	3.00
	565	37.83	14	2.71
	570	0.00	13	2.17
	20 scenarios			
	555	199.39	14	4.40
	560	128.23	16	4.89
	570	8.22	15	4.11
	575	0.00	13	3.10
Graph 3	5 scenarios			
	11650	1300.07	65	73.96
	11750	704.29	59	70.05
	11850	132.36	51	49.61
	11900	0.00	48	38.66
	10 scenarios			
	11700	1337.42	50	103.86
	11800	730.56	43	77.27
	11900	151.83	48	82.02
	11950	0.00	38	52.13
	20 scenarios			
	11900	1368.16	62	262.72
	12000	743.16	64	245.15
	12100	180.17	52	104.06
	12150	0.00	54	101.67

	Budget	obj. value	# oracle calls	CPU time
Graph 2	5 scenarios			
	280	288.92	30	8.66
	290	164.68	26	7.59
	300	50.56	29	8.58
	305	0.00	27	7.47
	10 scenarios			
	285	235.79	21	13.53
	295	116.86	18	11.57
	305	5.74	20	8.93
	310	0.00	17	6.77
	20 scenarios			
	285	276.43	26	27.70
	295	151.78	28	34.57
	305	34.82	21	17.42
	310	0.00	16	10.80
Graph 4	5 scenarios			
	5900	2776.79	29	103.03
	6000	1178.93	40	160.77
	6050	552.19	31	120.31
	6100	0.00	33	101.92
	10 scenarios			
	6000	2158.93	29	164.43
	6100	853.38	26	139.90
	6150	228.38	27	126.68
	6200	0.00	25	97.01
	20 scenarios			
	6000	2224.75	30	427.17
	6100	835.63	27	315.88
	6150	210.63	27	272.93
	6200	0.00	21	181.50

The greater the number of scenarios is, the better the uncertainty set minimize the forcast error. However, some traffic scenario may dominates others. One of the main features of the proximal bundle method is its small number of oracle calls that do not depend on the number of scenarios in hand.

6. Conclusions

We have proposed a new approach for the network design problem in telecommunications under demand uncertainty. The uncertainty is modelled as a convex set of a moderate number of demand's scenarios, which results in a convex non-smooth problem. For its solution, we considered and compared three algorithms: the proximal bundle method and the cutting plane algorithms by Kelley and Elzinga-Moore. The proximal bundle method appears to be the most efficient in terms of number of calls to the oracle, but is impeded by the use of a general-purpose quadratic solver. An extension of our proposed methodology to the general case of a polyhedral uncertainty set is currently under study.

References

[1] W. Ben-Ameur and H. Kerivin, Routing of uncertain traffic demands. *Optim. Eng.* **3** (2005) 283–313.
[2] A. Ben-Tal and A. Nemirovski, Robust convex optimization. *Math. Oper. Res.* **23** (1998).
[3] M. Bonatti, A. Gaivoronski, A. Lemonche and P. Polese, Summary of some traffic engineering studies carried out within RACE project R1044. *European Transactions on Telecommunications* **5** (1994) 79–90.
[4] O. Briant, C. Lemaréchal, Ph. Meurdesoif, S. Michel, N. Perrot and F. Vanderbeck, Technical Report 5453, Inria, 2005. accepted for publication at Mathematical Programming.
[5] A.V. Demyanov, V.F. Demyanov and V.N. Malozemov, Minimaxmin problems revisited. *Optim. Methods Softw.* **17** (2002) 783–804.
[6] N.G. Duffield, P. Goyal, A. Greenberg, P. Mishra, K.K. Ramakrishnan and J.E. van der Merwe, Resource managment with hoses: Point-to-cloud services for virtual private networks. *IEEE/ACM Trans. Networking* **10** (2002) 679–692.
[7] J. Elzinga and T.J. Moore, A central cutting plane algorithm for the convex programming problem. *Math. Program.* **8** (1975) 134–145.
[8] A. Frangioni, Solving semidefinite quadratic problems within nonsmooth optimization algorithms. *Comput. Oper. Res.* **23** (1996) 1099–1118.
[9] J.-L. Goffin, A. Haurie and J.-Ph. Vial, Decomposition and nondifferentiable optimization with the projective algorithm. *Manage. Sci.* **38** (1992) 284–302.
[10] J.L. Higle and S. Sen. Stochastic decomposition: An algorithm for two-stage linear programs with recourse. *Math. Oper. Res.* **16** (1991) 650–669.
[11] J.-B. Hiriart-Urruty and C. Lemaréchal, *Convex Analysis and Minimization Algorithms*. Springer-Verlag, Berlin (1993).
[12] G.F. Italiano, S. Leonardi and G. Oriolo, Design of networks in the hose model, in *2nd International Workshop on Approximation and Randomization Algorithms in Communication Networks*, Carleton Scientific Press (2002) 65–76.
[13] J.E. Kelley, The cutting plane method for solving convex programs. *J. Appl. Math. SIAM* **8** (1960) 703–712.
[14] K. Kiwiel, Proximity control in bundle methods for convex nondifferentiable minimization. *Math. Program.* **46** (1990) 105–122.

[15] K. Kiwiel, A cholesky dual method for proximal piecewise linear programming. *Numer. Math.* **68** (1994) 325–340.
[16] P. Kouvelis and G. Yu, *Robust Discrete Optimization and Its Applications*. Kluwer Academic Publishers (1997).
[17] A. Kumar, R. Rastogi, A. Silberschatz and B. Yener, Algorithms for provisioning virtual private networks in the hose model. *IEEE/ACM Trans. Networking* **10** (2002) 565–578.
[18] C. Lemaréchal, Lagrangian relaxation, in *Computational Combinatorial Optimization, Optimal or Provably Near-Optimal Solutions [based on a Spring School]* **2241**, London, UK (2001). Springer-Verlag 112–156.
[19] A. Lisser, A. Ouorou, J.-Ph. Vial and J. Gondzio, *Capacity planning under uncertain demand in telecommunication networks*. Technical Report 99.13, Logilab, Department of Management Studies, University of Geneva, Switzerland, October 1999.
[20] M. Lucertini and G. Paletta, A class of network design problems with multiple demand: Model formulation and an algorithmic approach. *Math. Program. Study* **26** (1986) 225–228.
[21] G.L. Nemhauser and W.B. Widhelm, A modified linear program for columnar methods in mathematical programming. *Oper. Res.* **19** (1971) 1051–1060.
[22] A. Ouorou, Robust capacity assignment in telecommunications. *Comput. Manage. Sci.* **3** (2006) 285–305.
[23] S. Sen, R.D. Doverspike and S. Cosares, Network planning with random demand. *Telecommun. Syst.* **3** (1994) 11–30.

Chapitre 3

Dimensionnement Robuste II

Ce chapitre reproduit notre article [51], soumis à OR Letters.

La problématique est la même qu'au chapitre précédent. La différence principale est qu'ici l'ensemble d'incertitude est un polyèdre décrit par un nombre fini d'inégalités linéaires, ce qui résulte en un problème considérablement plus difficile. Par conséquent, nous ne cherchons pas la solution optimale – qui s'est avérée difficilement calculable – mais uniquement des bornes supérieures et inférieures. Quelques idées novatrices sont présentées et l'algorithme de type "branch & bound" de Falk & Soland est utilisé afin de calculer le maximum d'une fonction convexe additive ; de plus, nous définissons une variante de cet algorithme, adaptée à notre situation particulière. L'approche est illustrée par des exemples.

manuscript No.
(will be inserted by the editor)

Claude Lemaréchal · Adam Ouorou · Georgios Petrou

Robust Network Design in Telecommunications under Polytope Demand Uncertainty

June 3, 2008

Abstract We consider a model for robust network design in telecommunications, in which we minimize the cost of the maximum mismatch between supply and demand. In the present study, the demand is uncertain and takes its values in a polytope defined by constraints. This problem is hardly tractable, so we limit ourselves to computing lower and upper bounds. A crucial building block for this construction is an algorithm due to Falk and Soland for maximizing a separable convex function over a polytope.

Keywords Telecommunication network design · Polyhedral uncertainty · Robust optimization · Min-max-min problems

Mathematics Subject Classification (2000) 65K05 · 90C25 · 90C27

1 Introduction

We consider the robust model for telecommunication network design presented in [8]. Let $G = (V, E)$ be a graph representing a telecommunication network, with $m = |V|$ nodes and $n = |E|$ links. A number K of origin-destination pairs is given and the demands lie somewhere in an uncertainty set Ω. Capacities $x \in \mathbb{R}^n_+$ can be installed on the arcs, let us denote abstractly by X the set of possible capacities. Let $\Gamma(x)$ be the set of all possible supply vectors that can be supported given a capacity vector x. An $\omega \in \Omega$ and a $\gamma \in \Gamma(x)$ induce a mismatch cost $f(\omega, \gamma)$ (which is 0 if $\gamma \geqslant \omega$); whenever ω is fixed, it is of course convenient to choose γ so as to minimize this mismatch.

C. Lemaréchal
INRIA Rhônes-Alpes, 655 avenue de l'Europe, Montbonnot, 38334, Saint Ismier

A. Ouorou, G. Petrou
France Telecom R&D, CORE/MCN, 38-40 rue du Général Leclerc, 92794 Issy-Les-Moulineaux cedex 9

Then robustness is measured by the function

$$\mu(x) := \max_{\omega\in\Omega} \min_{\gamma\in\Gamma(x)} f(\omega,\gamma) \tag{1.1}$$

and our robust design problem is the computation of

$$\underline{\mu} := \min_{x\in X} \mu(x) \quad \text{i.e.} \quad \min_{x\in X} \max_{\omega\in\Omega} \min_{\gamma\in\Gamma(x)} f(\omega,\gamma)\,. \tag{1.2}$$

In general, μ is nonconvex (due to the inner min) and nonsmooth (due to the outer max), so (1.1) is a a hard problem. See for example [2] for the difficulties appearing in min-max-min problems.

In [8], the case of Ω being the convex hull of a finite (and reasonable) number of scenarios was considered; this made the problem tractable and several cutting-plane methods were analyzed to solve the corresponding (1.2). Here we will consider a general polyhedral set. Then the approach of [8] can no longer be applied and we will propose two separate techniques: lower bounds of the optimal value are obtained by applying the method of [8] to inner approximations of Ω; and upper bounds are obtained by inverting the inner min and max in (1.2).

The paper is organized as follows. The model is more precisely specified in the next section. Sections 3 and 4 are respectively devoted to lower and upper bounds.

2 The model

Our uncertainty set is

$$\Omega = \big\{\omega \in \mathbb{R}^K_+ :\ R\omega \leqslant r\big\}. \tag{2.1}$$

Naturally, Ω must be bounded for the problem to make sense (otherwise the mismatch will usually be infinite); say $\Omega \subset B$, where $B = [\ell, L]$ is a box in $\mathbb{R}^K$.

Concerning X, we will assume that the cost of installing capacity x_i on arc i is linear with marginal cost c_i. A budget b for total investment is given, so the set of feasible capacities is

$$X = \big\{x \in \mathbb{R}^n_+ :\ c^\top x \leqslant b\big\}\,. \tag{2.2}$$

The exact forms of Ω and X have actually little importance for our development. Much more important are the forms of $\Gamma(x)$ and of f.

Choose an orientation of the arcs and call $A \in \mathbb{R}^{m\times n}$ the resulting node-arc incidence matrix. The flow vectors transiting the amount γ^k of commodity k are those $z^k \in \mathbb{R}^n$ satisfying the set of balance equations $Az^k = \gamma^k e^k$; here the components of $e^k \in \mathbb{R}^m$ are 0 except at the k^{th} source and destination nodes (where they are respectively -1 and $+1$). These z^k require a capacity $\sum_k |z^k_i|$ on each arc i.

In other words, $\gamma \in \Gamma(x)$ if and only if there exists $z \in \mathbb{R}^{nK}$ such that

$$Az^k = \gamma^k e^k, \;\; k = 1, \ldots, K\,, \quad \sum_{k=1}^{K} |z^k| \leqslant x \tag{2.3}$$

($z^k \in \mathbb{R}^n$ is the k^{th} subvector of z). This defines a polyhedron, whose form is specified by the following result:

Proposition 2.1 *$\gamma \in \Gamma(x)$ if and only if there exists $(z', z'') \in \mathbb{R}^{nK} \times \mathbb{R}^{nK}$ satisfying*

$$A(z' - z'')^k = \gamma^k e^k\,, \quad \sum_{k=1}^{K} (z' + z'')^k \leqslant x\,, \quad z' \geqslant 0\,,\; z'' \geqslant 0\,. \tag{2.4}$$

As a result: for three appropriate matrices $\mathcal{A}$, $\mathcal{A}_0$, $\mathcal{I}$,

$$\Gamma(x) = \mathcal{A}Y(x)\,, \quad \textit{where} \quad Y(x) = \{y \geqslant 0 :\; \mathcal{A}_0 y = 0, \mathcal{I}y \leqslant x\} \subset \mathbb{R}^{2nK}\,; \tag{2.5}$$

besides, the entries of $\mathcal{I}$ are 0 or 1.

Proof Given x, let $\gamma \in \Gamma(x)$ and a corresponding z satisfying (2.3). Set $z' := \max\{0, z\}$ and $z'' := \max\{0, -z\}$, so that $z = z' - z''$ and $|z| = z' + z''$; then (z', z'') satisfies (2.4).

Conversely, let (z', z'') satisfy (2.4). Then $z := z' - z''$ satisfies the balance equations. On the other hand,

$$|z_i^k| = \max\{(z')_i^k, (z'')_i^k\} \leqslant (z' + z'')_i^k\,, \quad \text{for } k = 1, \ldots, K \text{ and } i \in E\,;$$

so z is compatible with the capacity vector x.

Now introduce the K vectors $y^k \in \mathbb{R}^{2n}$ obtained by concatenating $(z')^k$ and $(z'')^k$, and write (2.4) with matrix notation. The particular form of the balance equations allows the computation of each γ^k in terms of y^k (for example, replace the two equations involving the source and destination by their sum and difference). It follows that the balance equations can be written $\gamma = \mathcal{A}y$, $\mathcal{A}_0 y = 0$. Besides, writing the capacity constraints as $\mathcal{I}y \leqslant x$ exhibits a matrix $\mathcal{I}$, obtained by concatenating $n \times n$ identity matrices. □

As for the penalty function f, there are several ways to choose it. We take a weighted sum of unsatisfied commodities. So, π^k being the cost of not satisfying one unit of demand for the corresponding OD-pair, we set

$$f(\omega, \gamma) = \sum_{k=1}^{K} \pi^k (\omega^k - \gamma^k)^+, \tag{2.6}$$

where $(r)^+ = \max\{0, r\}$. In view of Proposition 2.1, $f(\omega, \cdot)$ depends on y rather than γ: we are only interested in $f(\omega, \mathcal{A}y)$ in this paper.

Plugging (2.5) and (2.6) into the expression (1.1) of μ specifies completely the robust problem (1.2) and here are some crucial observations.

(i) The constraints defining $\Gamma(x)$ depend on x through their righthand side only and f is convex, jointly with respect to its argument (ω, γ).

(ii) From convexity theory, μ is then a convex function. Indeed, minimizing $f(\omega,\cdot)$ over $\Gamma(x)$ of (2.5) provides a convex function of x, for each ω; maximizing it over Ω preserves its convexity.

(iii) Besides, minimizing $f(\omega,\cdot)$ as above gives a function which is also convex in ω. Its maximization mentioned in (ii) is therefore an inherently difficult operation.

3 Computing lower bounds

Clearly enough, restricting ω to a set contained in Ω will result in underestimating the mismatch function μ of (1.1). Call Ω_t such a set:

$$\forall x \in X,\ \mu_t(x) := \max_{\omega\in\Omega_t} \min_{\gamma\in\Gamma(x)} f(\omega,\ \gamma) \leqslant \mu(x) \quad \text{if } \Omega_t \subseteq \Omega \tag{3.1}$$

and this inequality is transmitted to the infima:

$$\underline{\mu}_t := \min_{x\in X} \mu_t(x) \leqslant \min_{x\in X} \mu(x) = \underline{\mu}\,. \tag{3.2}$$

From convexity theory, μ_t is a convex function of x (just remember (ii) above). Besides, it can be computed relatively quickly if Ω_t is a polyhedron given by its extreme points; say $\Omega_t = \mathrm{conv}\,\{\omega_0,\ldots,\omega_t\}$. Then, computing $\mu_t(x)$ just amounts to minimizing $f(\omega,\cdot)$ for ω taking the $t+1$ values ω_0, $\ldots$, ω_t. In this situation, μ_t can be minimized by a cutting-plane algorithm; see [8] for details.

On the other hand, $\underline{\mu}_t$ increases if Ω_t increases. Our proposal is therefore to find a best possible $\underline{\mu}_t$ by a *column-generation* mechanism, in which Ω_t is iteratively inflated by appending new points to the list $\{\omega_\tau\}_{\tau=0}^t$:

Algorithm 3.1 (Lower Bound Computation)

Step 0 (Initiation). Choose an extreme point $\omega_0 \in \Omega$, set $\Omega_0 = \{\omega_0\}$ and $t = 0$.

Step 1 (Restricted Master). Compute $\underline{\mu}_t$ of (3.2).

Step 2 (Subproblem). Compute an extreme point ω_{t+1} of Ω not lying in Ω_t.

Step 3 (Stopping Test). If no such ω_{t+1} has been found, stop.

Step 4 (Loop). Otherwise set $\Omega_{t+1} = \mathrm{conv}\,\{\Omega_t, \omega_{t+1}\}$. Replace t by $t+1$ and go to Step 1. □

In addition to the value $\underline{\mu}_t$, Step 1 produces also various optimal arguments, namely:

– some $x_t \in X$ which minimizes μ_t of (3.1),

– some $\bar\omega_t$ such that

$$\min_{\gamma\in\Gamma(x_t)} f(\bar\omega_t,\gamma) = \mu_t(x_t)$$

(as already mentioned, $\bar\omega_t$ is one of the points in the list $\{\omega_\tau\}_{\tau=0}^t$),

– some $\gamma_t \in \Gamma(x_t)$ minimizing $f(\bar\omega_t,\cdot)$.

These objects will be useful for Step 2.

Remark 3.2 Another idea to generate lower bounds is to invert the outer min and max in (1.2). In fact, standard min-max theory says that

$$\nu(\omega) := \min_{x\in X} \min_{\gamma\in\Gamma(x)} f(\omega,\gamma) \leqslant \underline{\mu} \quad \text{for all } \omega \in \Omega\ .$$

Again convex analysis says that ν is a convex function of ω. Its maximization is not an easy problem and a possible approach is again by column generation: ν is maximized over some polyhedron $\Omega_t \subset \Omega$, which is iteratively inflated. Using again min-max theory, one sees that

$$\underline{\nu}_t := \max_{\omega\in\Omega_t} \min_{x\in X} \min_{\gamma\in\Gamma(x)} f(\omega,\gamma) \leqslant \min_{x\in X} \max_{\omega\in\Omega_t} \min_{\gamma\in\Gamma(x)} f(\omega,\gamma) = \underline{\mu}_t\,.$$

In other words, Algorithm 3.1, which does not involve any min-max inversion, provides better lower bounds at each iteration. □

We now turn to the computation of ω_{t+1} and the stopping test in Algorithm 3.1.

3.1 Generating the new scenario

Our proposal for the subproblem uses the following function (see Proposition 2.1 for the notation):

$$\varphi_t(\omega) := \min_{\gamma\in\Gamma(x_t)} f(\omega,\gamma) = \min_{y\in Y(x_t)} f(\omega,\mathcal{A}y)\,. \tag{3.3}$$

In fact, φ_t is a marginal function (see [5, § B.2.4] for example). From (2.6), f is jointly convex, so φ_t is convex as well. Our suggestion is to maximize over the whole of Ω an affine lower approximation of φ_t: we compute ω_{t+1} by solving

$$\max_{\omega\in\Omega} u_t^\top \omega\,, \quad \text{where } u_t \in \partial\varphi_t(\bar\omega_t)\ . \tag{3.4}$$

In view of (2.1), our subproblem is thus a linear program. It produces a new column satisfying $u_t^\top\omega_{t+1} \geqslant u_t^\top\bar\omega_t$ and bringing new information indeed:

Theorem 3.3 *There holds $\varphi_t(\omega_{t+1}) \geqslant \varphi_t(\omega)$ for all $\omega \in \Omega_t$.*

In fact, $\varphi_t(\omega_{t+1}) > \varphi_t(\bar\omega_t)$ and $\omega_{t+1} \notin \Omega_t$, unless $\bar\omega_t$ is an optimal solution of (3.4).

Proof Because u_t is a subgradient,

$$\varphi_t(\omega_{t+1}) \geqslant \varphi_t(\bar\omega_t) + u_t^\top(\omega_{t+1}-\bar\omega_t) \geqslant \varphi_t(\bar\omega_t)\,,$$

where the second inequality comes from the definition (3.4) of ω_{t+1}. On the other hand, set $x = x_t$ in the optimization problem (3.2) to realize that $\bar\omega_t$ maximizes φ_t over Ω_t. Thus, for all $\omega \in \Omega_t$,

$$\varphi_t(\omega_{t+1}) \geqslant \varphi_t(\bar\omega_t) + u_t^\top(\omega_{t+1}-\bar\omega_t) \geqslant \varphi_t(\bar\omega_t) \geqslant \varphi_t(\omega)\,,$$

which proves the first statement.

The second statement comes when the second inequality above is strict, i.e. when $u_t^\top\omega_{t+1} > u_t^\top\bar\omega_t$. □

As a result, the algorithm is guaranteed to strictly inflate Ω_t at each iteration.

3.2 Subdifferentiating the marginal function

Now we have to specify how u_t is computed in (3.4); this is just a technical exercise in convex analysis, using [5, § D.4.5]. For simplicity, we limit ourselves to the polyhedral case described in §2.

In fact, $\varphi_t(\bar\omega_t)$ is the optimal value of the linear program

$$\min_{(y,\delta)\geqslant 0} \pi^\top \delta\,, \quad \text{s.t. } \delta \geqslant \bar\omega_t - \mathcal{A}y,\ \mathcal{A}_0 y = 0,\ \mathcal{I}y \leqslant x_t\,,$$

where the notation comes from Proposition 2.1. By LP duality, $\varphi_t(\bar\omega_t)$ is also the optimal value of the dual

$$\max_{(u,v)\geqslant 0} \bar\omega_t^\top u - x_t^\top v\,, \quad \text{s.t. } \mathcal{A}_0^\top w - \mathcal{A}^\top u + \mathcal{I}^\top v \geqslant 0,\ u \leqslant \pi\,. \tag{3.5}$$

Theorem 3.4 *Any u forming an optimal solution of* (3.5) *lies in $\partial\varphi_t(\omega_t)$.*

Proof In the notation of [5, § D.4.4], call J the set of feasible $j = (u, v, w)$ in (3.5). Then φ_t is the largest f_j over J; here $f_{(u,v,w)}(\omega) = u^\top\omega - x_t^\top v$, whose (sub)differentiation with respect to ω gives u. The statement is then Lemma D.4.4.1 of [5]. □

Thus, $\varphi_t(\bar\omega_t)$ is obtained by a linear program, and the required subgradient for (3.4) is obtained as the multiplier of the appropriate constraint.

3.3 Implementation

Putting together the above results, the description of Algorithm 3.1 can be completed:

Step 2 (Subproblem). Having x_t and $\bar\omega_t$ from Step 1, obtain a multiplier u_t, for example by solving (3.5).
Solve (3.4) to obtain ω_{t+1}.
Step 3 (Stopping Test). If $\varphi_t(\omega_{t+1}) = \varphi_t(\bar\omega_t)$ stop (see Theorem 3.3). □

A tolerance of the type $\varphi_t(\omega_{t+1}) \leqslant \varphi_t(\bar\omega_t) + \varepsilon$ can be inserted in Step 3; it preserves the crucial property $\omega_{t+1} \neq \bar\omega_t$ anyway. Since a new extreme point of the polyhedron Ω is appended to the list at each iteration, the algorithm must eventually stop.

On the other hand, little can be said when the algorithm stops: $\underline{\mu}_t$ need not be the optimal value of (1.2), then. To illustrate this, take for example $\omega_0 = 0$ (assumed to lie in Ω). The first iteration will probably compute $\gamma_0 = 0$, $x_0 = 0$, $u_0 = 0$, so chances are that (3.4) will produce $\omega_1 = 0 = \omega_0$: the algorithm answers the trivial bound $\mu \geqslant \mu_0 = 0$. More generally, the algorithm can hardly proceed when the scenarios in Ω_t are so cheap that the budget constraint is inactive. Various heuristic techniques can be suggested to improve the final lower bound.

(i) When the current u_t no longer produces a useful ω_{t+1}, inject a random u into (3.4),
(ii) or diminish the budget b until a useful u comes out from (3.5).

(iii) In addition to such "emergency restarts", a more elaborate initialization can also be used, with more than one ω_0. Our numerical experiments of §5.2 will present a possibility.

The fragility of our present approach is actually an inherent difficulty of the problem: the only way out would be to compute the actual value $\mu(x_t)$, i.e. to maximize φ_t over Ω; but φ_t is convex... The next section will introduce a simplified version of this hard problem.

4 Computing upper bounds

Our starting operation to compute upper bounds is to invert the min and max in (1.1): standard min-max theory says that

$$\min_{\gamma\in\Gamma(x)} \max_{\omega\in\Omega} f(\omega,\gamma) \geqslant \mu(x) \tag{4.1}$$

for all $x \in X$.

Remark 4.1 The lefthand side in this relation is the worst mismatch that could possibly occur if supply had to be decided before knowing the demand. In fact, given a supply vector γ, we compute the most expensive ω; and then we compute the best γ according to this objective.

Such a strategy is far from reflecting reality. It can therefore be thought that we will obtain very pessimistic overestimates of the true mismatch μ. □

The lefthand side in (4.1) must now be minimized over $x \in X$; but we may as well perform the minimizations with respect to x and γ simultaneously: we will obtain a minimization problem in (x,γ), whose objective function is the maximum of f over $\omega \in \Omega$.

4.1 The algorithm

As explained above, our upper bound is obtained by solving

$$\min_{x\in X,\gamma\in\Gamma(x)} W(\gamma)\,, \quad \text{where} \quad W(\gamma) := \max_{\omega\in\Omega} f(\omega,\gamma)\,. \tag{4.2}$$

Lemma 4.2 *For all $x \in X$ and $\gamma \in \Gamma(x)$, we have $W(\gamma) \geqslant \mu(x)$. Besides, W is convex.*

Proof Take $\omega \in \Omega$ and $\gamma \in \Gamma(x)$ (with $x \in X$). By definition of W,

$$W(\gamma) \geqslant f(\omega,\gamma) \geqslant \min_{\gamma'\in\Gamma(x)} f(\omega,\gamma')$$

and the inequality is transmitted to the supremum of the righthand side, which is $\mu(x)$.

Convexity is a classical result: W is the pointwise maximum of $f(\cdot,\gamma)$, which is convex in γ. □

Thus, solving (4.2) does provide an upper bound of $\underline{\mu}$. Let us admit that a suitable algorithm can compute $W(\gamma)$. Standard convex analysis then allows the computation of a subgradient:

Lemma 4.3 *For given γ, let ω_γ maximize $f(\cdot,\gamma)$ over Ω. Then the vector $g \in \mathbb{R}^K$ defined by*

$$g^k = \begin{cases} 0 & \text{if } \omega_\gamma^k < \gamma^k, \\ -\pi^k & \text{otherwise} \end{cases}$$

lies in $\partial W(\gamma)$.

Proof With the expression (2.6) of f, this is [5, Lemma D.4.4.1]. □

Note that $W(\gamma)$ is defined over the whole of $\mathbb{R}^K$, independently of the constraint $\gamma \in \Gamma(x)$. Note also that this latter constraint is "explicit" (Prop. 2.1), while information about $W(\gamma)$ is only available through the oracle maximizing $f(\cdot,\gamma)$. In this situation, a cutting-plane approach to solve (4.2) is appropriate: at iteration t, it replaces W by the polyhedral function

$$W_t(\gamma) := \max\left\{W(\gamma_\tau) + g_\tau^\top(\gamma - \gamma_\tau) : \tau = 1, \dots, t\right\}$$

obtained by successive linearizations based on Lemma 4.3. Accordingly, (4.2) is replaced by the minimization of W_t with respect to $(x,\gamma) \in X \times \Gamma(x)$. Using Proposition 2.1, this is the linear program

$$\begin{array}{ll} \min\limits_{(x,y,r)\geqslant 0} r\,, & \text{s.t. } c^\top x \leqslant b\,,\ \mathcal{A}_0 y = 0\,,\ \mathcal{I} y \leqslant x, \\ & \qquad r \geqslant W(\gamma_\tau) + g_\tau^\top(\mathcal{A}y - \gamma_\tau),\ \tau = 0, \dots, t. \end{array} \tag{4.3}$$

Note that we have inserted the constraint $[W_t(\cdot) =]\, r \geqslant 0$, which can only do good, indeed: from (2.6) and (4.2), W is obviously a nonnegative function.

A usual difficulty in cutting-plane algorithms is initialization: they need many enough initial linearizations so that W_t does have a minimum. Here, this difficulty is eliminated by the constraint $r \geqslant 0$; anyway, note that X and $\Gamma(x)$ are bounded sets: (4.3) would always have a finite minimum even without the constraint $r \geqslant 0$. In fact, a natural initial iterate is $\gamma_0 = 0$; because $\Omega \subset \mathbb{R}^K_+$, we certainly have $f(\omega, 0) = \pi^\top\omega$ for all $\omega \in \Omega$; computing $W(0)$ is thus an ordinary linear program and then $g_0 := -\pi$ lies in $\partial W(0)$, as predicted by Lemma 4.3.

Let us also mention that a quadratic term could be added to each polyhedral model W_t, to stabilize the cutting-plane algorithm by a bundle variant. Neglecting here this latter option for simplicity, we obtain the following pattern.

Algorithm 4.4 (Upper Bound Computation)

Step 0 (Initiation). Start with $\gamma_0 = 0$; set $t = 0$. Fix a stopping tolerance $\varepsilon \geqslant 0$. Compute $W(0) = \max\limits_{\omega\in\Omega} \pi\omega$ and $g_0 = -\pi$.

Step 1 (Restricted Master). Compute an optimal solution $(x,y,r)_{t+1}$ of (4.3) and set $\gamma_{t+1} = \mathcal{A}y_{t+1} \in \Gamma(x_{t+1})$.

Step 2 (New Scenario). Maximize $f(\omega, \gamma_{t+1})$ for $\omega \in \Omega$ to obtain ω_{t+1} and $W(\gamma_{t+1})$. Set g_{t+1} as described in Lemma 4.3.

Step 3 (Stopping Test). If $W(\gamma_{t+1}) \leqslant r_{t+1} + \varepsilon$, stop.
Step 4 (Loop). Replace t by $t+1$ and go to Step 1. □

Note that $r_{t+1} = W_t(\gamma_{t+1})$, which underestimates the optimal value in (4.2). Eeach ω_t is an extreme point of Ω, so this algorithm should stop after finitely many iterations, even with $\varepsilon = 0$. A strictly positive tolerance is nevertheless advisable. In contrast with Algorithm 4.4, we have here a "safe" method, which solves an "ordinary" convex problem by an "ordinary" cutting-plane algorithm [1,7]. When it stops, (4.2) is really solved within ε accuracy; as mentioned in Remark 4.1, the corresponding capacities would be (approximately) optimal if supply had to be computed before demand were known. Little can be said concerning the true optimal value since the exact value $\mu(x_t)$ is still unknown. Even a small gap between the optimal value of W and the best $\underline{\mu}_t$ from Algorithm 3.1 can hardly be hoped.

Let us also mention that the variable x in Step 1 is superfluous: we can just take $x = \mathcal{I}y$ (a nonnegative vector) and the first three constraints in (4.3) are reduced to $c^\top \mathcal{I} y \leqslant b$, $\mathcal{A}_0 y = 0$.

4.2 Computing worst scenarios: the algorithm of Falk-Soland

It remains to specify the implementation of Step 2 in Algorithm 4.4, i.e. the maximization over Ω of the function $\varphi := f(\cdot, \gamma_{t+1})$.

Remember that φ is defined on a box $B = [\ell, L]$ and note that it is a sum $\sum_k \varphi^k$ of univariate convex functions. The concave hull $\bar{\varphi}^k$ of each φ^k over an interval $[\ell^k, L^k]$ is very simple (see Fig. 4.1): $\bar{\varphi}^k$ is the affine function passing through $\varphi^k(\ell^k)$ and $\varphi^k(L^k)$. Then it can be seen that the concave hull $\bar{\varphi}$ of φ is likewise a sum:

$$\bar{\varphi}(\omega) = \sum_{k=1}^{K} \bar{\varphi}^k(\omega^k)\,.$$

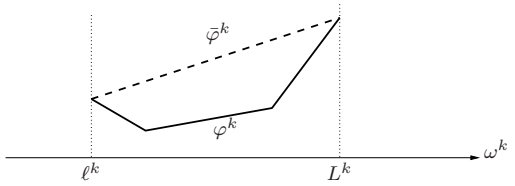

Fig. 4.1 The concave hull of a 1-dimensional convex function

It will therefore be easy to compute concave functions (actually affine) larger than φ; being affine, such functions will easily be maximized over a polyhedron. These observations suggest an arborescent algorithm [4] to maximize $\varphi = f(\cdot, \gamma_{t+1})$ over $\Omega \subset B$.

(0) At iteration 0 (the root node), φ is replaced by its concave hull $\bar{\varphi}_0$ on B. Then $\bar{\varphi}_0$ is maximized over $B \cap \Omega = \Omega$, thus providing an overestimate Φ_0 of the true maximal value $W(\gamma_{t+1})$.

(i) Each iteration s of the algorithm (each node s of the tree) is characterized by a box $B_s := [\ell_s, L_s] \subset B$. One constructs the concave hull $\bar{\varphi}_s$ of φ over B_s and one solves the linear program

$$\max \bar{\varphi}_s(\omega)\,, \quad \omega \in B_s \cap \Omega$$

to obtain an optimal solution ω_s and the maximal value $\Phi_s = \bar{\varphi}_s(\omega_s)$.

(ii) Then a particular node s^* is chosen among those already explored, namely one for which Φ_{s^*} is largest; note that $\Phi_{s^*} \geqslant \Phi_0 \geqslant W(\gamma_{t+1})$.

(iii) If $\varphi(\omega_{s^*}) = \bar{\varphi}_{s^*}(\omega_{s^*})\,[= \Phi_{s^*}\,]$, stop: ω_{s^*} maximizes φ over Ω.

(iv) Otherwise, one chooses a coordinate k such that $\ell^k_{s^*} < \omega^k_{s^*} < L^k_{s^*}$ (there is certainly one) to split B_{s^*} into two subboxes; this defines two branches issued from node s^* and the algorithm can proceed.

In the general algorithm, B_{s^*} is simply split into equal subboxes but we propose a variant. In fact, each function φ^k is simpler than suggested by Fig. 4.1, as it is made up of exactly two affine pieces which meet at the known point γ^k_{t+1}. In Step (iv), the most natural way of splitting a box $B_s = [\ell_s, L_s]$ along a direction k therefore consists in replacing the segment $[\ell^k_s, L^k_s]$ by the two segments $[\ell^k_s, \gamma^k_{t+1}]$ and $[\gamma^k_{t+1}, L^k_s]$ (instead of splitting $[\ell^k_s, L^k_s]$ through its middle): see Fig. 4.2. On each of the resulting subboxes, φ^k will coincide with its concave hull.

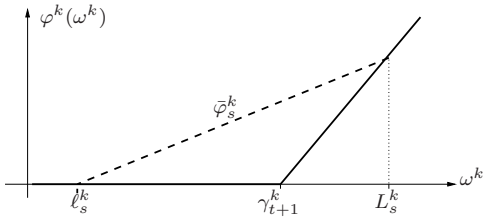

Fig. 4.2 Splitting a box B_s along a direction k

In at most 2^K such splits, φ will be accurately represented and the algorithm will stop in Step (iii). This is still a big number, though: the algorithm is still combinatorial and its execution time is unpredictable; so Algorithm 4.4 is potentially fairly expensive. Such a shortcoming seems hard to avoid anyway, as long as f in (1.1) is not a concave-convex function.

5 Numerical illustration

To assess our approach, we have performed a number of numerical tests, which are now reported.

5.1 The environment

We worked on a Intel PC Core 2 Duo E6550 of 2.34 GHz CPU speed and 2 GB of RAM memory, running under Microsoft Windows XP Home Edition, with Service Pack 2. The compiler was Microsoft Visual C++ 2008 Express Edition. The various linear programs appearing in our algorithms were solved by COIN-OR Open Solver Interface and COIN-OR LP solver.

We used a fixed actual network, on which we defined three sets of OD-pairs, giving birth to three instances. The corresponding uncertainty sets Ω were derived from the *hose* model [3], considering only the ingress and outgress traffic for every terminal node of the network. With this formulation, the constraints in (2.1) take the form

$$\omega \geqslant 0\,, \quad \sum_{k=1}^{K} R_{jk}\omega^k \leqslant r_j\,,\ j = 1, \ldots, N\,,$$

where $R_{jk} \in \{0, 1\}$ and $r_j \geqslant 0$.

Table 5.1 summarizes the dimensions of the instances A, B, C thus generated. For future use (see §5.3 below), the table also gives for each instance the largest possible penalty, resulting from satisfying no demand at all. Indeed, note that $f(\omega, 0)$ is linear, so the problem

$$W(0) = \max_{\omega \in \Omega} \pi^\top \omega$$

is easy (here W is the function from (4.2)). The last column of Table 5.1 gives its optimal value.

Instance	m	n	K	N	$W(0)$
A	12	25	13	15	2935.5
B	12	25	26	18	5050.
C	12	25	40	20	8800.

Table 5.1 Number of nodes, of edges, of OD-pairs, and of constraints defining Ω; maximal mismatch cost

For each of the three instances thus generated, we used four different budgets b in (2.2). Altogether, this made 12 test-problems.

5.2 Lower bounds

We have implemented Algorithm 3.1 in the above environment, with a number of algorithmic features aimed at coping with its heuristic character.

(i) Two different methods were used to compute μ_t of (3.2): Kelley and Elzinga-Moore, that we presented in [8]. We thus doubled the number of experiments for each test-problem.

(ii) As was alluded to in (iii) of §3.3, the algorithm was initialized with more than one point as follows. First, setting $u_0 = (1, \ldots, 1)^\top$ in (3.4) produced ω_0. Then, for $k = 1, \ldots, K$, (3.4) was solved again with the k^{th} basis vector as cost row. This produced an ω^k, which was appended to the list of initial points if it did not lie in their convex hull. Up to $K + 1$ initial points were therefore produced. More precisely: 9, 20 and 33 initial points were produced for instances A, B and C respectively (out of the 12, 27 and 41 possible).

To check whether a given ω lies in a list $\{\omega_t\}_{t=0}^T$, one solves the linear program

$$\min \sum_{t=0}^{T} \lambda_t \,, \quad \sum_{t=0}^{T} \lambda_t \omega_t = \omega \,, \quad \sum_{t=0}^{T} \lambda_t = 1 \,, \quad \lambda_t \geqslant 0 \,,\ t = 0, \ldots, T \,; \tag{5.1}$$

the answer is yes if it has a feasible solution.

(iii) Likewise, more than one new scenario was produced by Step 2 at each iteration. In fact, having the current x_t solving (3.2), we took all the optimal $\bar{\omega}_t$ from the current list. Each of them was injected into (3.5), producing a u_t which was in turn injected into (3.4). The new scenario thus produced was appended to the list, unless it lied in the current Ω_t – see (5.1).

(iv) Additionally, the stopping test mentioned in §3.3 was overlooked: the algorithm was stopped only when (5.1) indicated that no new ω_{t+1} from (iii) above lied out of Ω_t.

Budget	#Scen	Kelley		Elzinga-Moore	
		Bound	CPU	Bound	CPU
120	9	22.06	0.23	22.06	0.25
	25	48.13	4.98	*25.22	*2.00
	40	69.75	13.33	—	—
116	9	72.06	0.19	72.06	0.28
	25	119.75	4.76	131.79	7.38
	40	119.75	14.43	131.79	18.48
112	9	124.93	0.21	124.93	0.37
	25	157.36	5.00	157.36	13.18
	40	157.36	12.45	177.14	23.00
108	9	182.07	0.24	182.07	0.28
	25	219.75	5.06	239.79	8.06
	40	221.43	13.83	247.97	20.98

*Natural stop after 15 scenarios

Table 5.2 Lower bounds for instance A up to 40 scenarios

Tables 5.2 to 5.4 present the results: best f-value and computing time in minutes, for the 3 instances and with the 4 values of b. These times are rather long, so the runs were actually stopped after a maximum number of scenarios were generated. We present the results for the initial set of scenarios,

for the maximum number of scenarios (respectively 40, 40 and 45) and for an intermediate number. For example, the first row of Table 5.2 says that the inital set of 9 scenarios gives the lower bound 22.06 with both methods. Kelley [resp. E-M] improves this bound to 48.13 [resp. 25.22] in 4.98 minutes [resp. 2 min.] with 25 [resp. 15] scenarios. Then Kelley continues to obtain the best bound of 69.75 in 13.33 minutes.

Budget	#Scen	Kelley		Elzinga-Moore	
		Bound	CPU	Bound	CPU
198	20	8.07	1.01	8.07	1.48
	30	54.57	13.27	*8.07	*3.68
	40	100.57	31.28	—	—
192	20	93.79	1.09	93.79	1.93
	30	167.58	18.92	148.34	28.35
	40	196.57	39.58	288.08	65.23
186	20	179.50	1.24	179.50	3.96
	30	295.67	20.05	253.57	43.00
	40	404.42	50.78	406.43	84.30
180	20	271.42	1.07	271.42	0.99
	30	390.00	17.81	357.50	15.28
	40	494.17	44.12	506.04	42.63

*Natural stop after 21 scenarios

Table 5.3 Lower bounds for instance B up to 40 scenarios

Budget	#Scen	Kelley		Elzinga-Moore	
		Bound	CPU	Bound	CPU
350	33	1.72	4.03	1.72	3.38
	38	291.56	24.41	291.79	35.09
	45	293.13	56.33	303.86	78.63
342	33	98.13	4.02	98.13	3.92
	38	324.71	28.08	194.82	59.11
	45	376.71	68.85	376.33	107.18
334	33	198.13	4.40	198.13	4.63
	38	451.07	35.14	627.93	38.38
	45	478.26	80.38	694.64	77.47
326	33	302.90	6.18	302.90	4.75
	38	471.42	33.56	529.29	43.71
	45	475.92	77.50	596.00	90.47

Table 5.4 Lower bounds for instance C up to 45 scenarios

The disparity of these results demonstrates the heuristic character of our algorithm. This is especially spectacular as both methods solve the same problems and should obtain similar results. In fact, starting with the same initial Ω_0, they produce the same initial bound. But μ_t – in particular μ_0 – may have several minimum points: the two methods compute two different

optimal x_0 and then start to diverge. A question of interest is then: how significant are these disparities? This question is related with the computation of upper bounds, which we report now.

5.3 Upper bounds and final comments

We have implemented Algorithm 4.4 on the same 12 test-problems. The results are reported on Tables 5.5 to 5.7: number of iterations, upper bound and computing time in minutes. For example, the first row of Table 5.5 shows that for instance A and $b = 120$, 45 iterations of the alogorithm take 0.07 minute and produce the upper bound 354.63.

Budget	#Iter	Bound	CPU
120	45	354.63	0.07
116	46	404.63	0.07
112	51	454.63	0.10
108	43	504.63	0.05

Table 5.5 Upper bounds for instance A

Budget	#Iter	Bound	CPU
198	113	1395.87	19.82
192	120	1470.87	16.00
186	123	1545.88	18.15
180	119	1620.88	14.07

Table 5.6 Upper bounds for instance B

Budget	#Iter	Bound	CPU
350	207	1879.38	59.28
342	184	1988.29	48.97
334	207	2130.80	176.88
326	208	2223.80	207.72

Table 5.7 Upper bounds for instance C

Note that one iteration involves the maximization of $\varphi = f(\cdot, \gamma)$ over Ω, by the combinatorial algorithm of §4.2. Nevertheless, the computing times are reasonable. Comparing with those of Algorithm 3.1, the present times are negligible (instance A), definitely smaller (instance B) or a little bigger (instance C). We believe that taking advantage of the particular form of φ (see Fig. 4.2) is crucial: the original version of Falk-Soland's algorithm is indeed fairly expensive.

Let us conclude with some comments on our numerical results. We see that the upper bounds produced in this section are grossly larger than the lower bounds in §5.2. Keeping in mind Remark 4.1, this could somehow be expected. On the other hand, the differences should be assessed in relative terms. We propose to use for this the maximal mismatch $W(0)$ of Table 5.1 as a reference value. Even though it grossly overestimates the optimal cost, it should at least provide a reasonable order of magnitude.

Table 5.8 reports the differences between the various bounds compared to $W(0)$: U [resp. K, resp. EM] denotes the upper bound [resp. lower bound of Kelley, resp. Elzinga-Moore]. For example, the last column of its first row says from Tables 5.1, 5.2, 5.5 that

$$\frac{\text{EM lower bd. — K lower bd.}}{W(0)} = \frac{25.22 - 69.75}{2935.5} = -0.0152 = -1.52\% \ .$$

In other words, Kelley's bound is 1.25% better than Elzinga-Moore's on this instance.

Instance	Budget	100 x (difference of bounds)/$W(0)$		
		U vs. K	U vs. EM	EM vs. K
A	120	9.70	11.22	−1.52
	116	9.70	9.29	0.41
	112	10.13	9.45	0.67
	108	9.64	8.73	0.90
B	198	25.65	27.48	−1.83
	192	25.23	23.42	1.81
	186	22.60	22.56	0.04
	180	22.31	22.08	0.24
C	350	18.03	17.90	0.12
	342	18.31	18.32	−0.00
	334	18.78	16.32	2.46
	326	19.86	18.50	1.36

Table 5.8 Comparison of bounds

With this criterion, we see that the discrepancy between Kelley's and Elzinga-Moore's bounds is much less significant than suggested by §5.2: in fact, both methods agree within a few percents.

On the other hand, the difference between lower and upper bounds is substantial, although not disastrous. As already mentioned (Remark 4.1), the blame is probably on the upper bound; altogether, lower bounds seem "reasonable". However, to solve (1.2) accurately and/or reliably, new ideas are needed.

A possibility is d.c. optimization [6]. Indeed a cutting-plane algorithm could be used to minimize μ over X if this latter function could be computed. Now this computation is the maximization of a convex function over Ω – a typical d.c. problem. Such an approach, however, is bound to be fairly heavy because

– the d.c. algorithm must be executed at each iteration of the outer cutting-plane algorithm minimizing μ,
– convergence of this latter algorithm is usually slow (minimizing a nonsmooth convex function is never easy).

In other words, we have a slowly convergent algorithm, whose each iteration is expensive. These considerations demonstrate once more that (1.2) is a hard problem when f is a convex function.

References

1. E. Cheney and A. Goldstein. Newton's method for convex programming and Tchebycheff approximations. *Numerische Mathematik*, 1:253–268, 1959.
2. A. V. Demyanov, V. F. Demyanov, and V. N. Malozemov. Minmaxmin problems revisited. *Optimization Methods and Software*, 17:783–804, 2002.
3. N. G. Duffield, P. Goyal, A. Greenberg, P. Mishra, K. K. Ramakrishnan, and J. E. van der Merwe. Resource management with hoses: Point-to-cloud services for virtual private networks. *IEEE/ACM Transactions on Networking*, 10(5):679–692, October 2002.
4. J. E. Falk and R. M. Soland. An algorithm for separable nonconvex programming problems. *Management Science*, 15(9):550–569, May 1969.
5. J.-B. Hiriart-Urruty and C. Lemaréchal. *Fundamentals of Convex Analysis*. Springer-Verlag, 2001.
6. R. Horst and H. Tuy. *Global optimization. Deterministic approaches*. Springer Verlag, 1996.
7. J. E. Kelley. The cutting plane method for solving convex programs. *Journal of the SIAM*, 8:703–712, 1960.
8. G. Petrou, C. Lemaréchal, and A. Ouorou. An approach to robust network design in telecommunications. RAIRO Recherche Opérationnelle, to appear, 2007.

Chapitre 4

Optimisation de la Qualité de Service

Ce chapitre reproduit notre article [40], a paraître dans Computational Optimization and Applications.

Après avoir défini la capacité d'un réseau, l'étape suivante est de calculer le routage optimal dans ce réseau. Ici, nous minimisons la congestion en utilisant comme objectif la fonction moyenne de retard de Kleinrock. Le problème résultant est convexe mais non linéaire ; afin de le résoudre, nous implémentons un algorithme hybride basé sur la relaxation lagrangienne. La fonction duale est la somme d'un terme polyédral et d'un terme différentiable, notre algorithme traite le terme polyédral [resp. différentiable] via le paradigme des plans sécants [de Newton]. Dès lors, une méthode de type faisceaux proximale est utilisée, adaptée à la forme particulière de la fonction duale. Des illustrations numériques montrent la validité de cette approche, et ébauchent une comparaison avec l'approche similaire de [3].

Noname manuscript No.
(will be inserted by the editor)

Claude Lemaréchal · Adam Ouorou · Georgios Petrou

A bundle-type algorithm for routing in telecommunication data networks

Abstract To optimize the quality of service through a telecommunication network, we propose an algorithm based on Lagrangian relaxation. The bundle-type dual algorithm is adapted to the present situation, where the dual function is the sum of a polyhedral function (coming from shortest path problems) and of a smooth function (coming from the congestion function).

Keywords Convex optimization, routing, multicommodity flows, Kleinrock delay function

1 Introduction

1.1 The model

We consider the following problem

$$\begin{array}{ll} \min & f(y) := \sum_{j=1}^{n} f_j(y_j) \\ \text{s.t.} & Ax^k = b^k \in \mathbb{R}^m, \quad k = 1,\ldots,K, \\ & \sum_{k=1}^{K} x^k = y \in \mathbb{R}^n, \\ & 0 \leqslant y < c, \quad x^k \geqslant 0,\ k = 1,\ldots,K, \end{array} \tag{1.1}$$

where

· A is the node-arc incidence matrix of a graph $G = (V,E)$ (m nodes, n arcs),

C. Lemaréchal
INRIA Rhônes-Alpes, 655 avenue de l'Europe, Montbonnot, 38334, Saint Ismier

A. Ouorou, G. Petrou
France Telecom R&D, CORE/MCN, 38-40 rue du Général Leclerc, 92794 Issy-Les-Moulineaux cedex 9

· x^k are flow vectors representing K commodities between source nodes s_k and sink nodes t_k, $k = 1,\dots,K$,
· $b^k \in \mathbb{R}^m$ are vectors with two nonzero components (corresponding to an origin s_k and destination t_k) such that $\sum_{i=1}^m b_i^k = 0$,
· y is the total link flow vector,
· f_j is proportional to the Kleinrock average delay function:

$$f_j(y_j) = \frac{y_j}{c_j - y_j} \quad \text{if } 0 \leqslant y_j < c_j \text{ (and } +\infty \text{ otherwise)}\ , \tag{1.2}$$

· $c \in \mathbb{R}^n$ is the vector of arc capacities.

Problem (1.1), often called (convex) *multicommodity flow*, occurs in data communication networks and plays an important role in the optimization of network performances. We mention here that the delay function may assume other forms than (1.1). Our approach is significant only when the f_j's are *nonlinear*.

1.2 Numerical solution methods: outline

Various methods have been proposed in the literature to solve the multicommodity flow problem, we refer to [20] for a review. They can be classified according to three basic paradigms:

(i) Direct methods exploit the problem's block structure. Most popular is flow deviation [7] because of its simplicity; it is a special case of the Frank-Wolfe method [6], which works on a sequence of linearized problems. It has slow convergence and many authors have tried to improve it.
(ii) Other classical mathematical programming algorithms (Newton, conjugate gradient) have been adapted to the structure of (1.1); see [2] for example.
(iii) Some proposals adopt a dual point of view: Lagrangian relaxation is applied to the constraints linking x and y. This results in a concave (nonsmooth) dual function to be maximized, which can be done by a suitable algorithm such as proximal, ACCPM, subgradient, ...

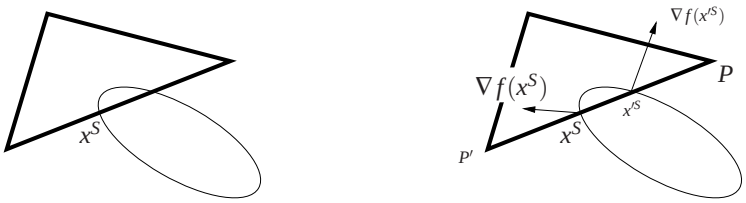

Fig. 1.1 Instability of Frank-Wolfe

Judged from a nonlinear optimization point of view, methods of type (i) suffer serious convergence deficiencies; we illustrate them on Fig. 1.1, which assumes $K = 1$ for simplicity (then $y = x$). The left part of the picture displays the polyhedron $\{Ax = b, x \geqslant 0\}$, as well as the level set $f(x) = f(x^S)$ passing through the

current iterate x^S (which is close to optimality). The essential idea of flow deviation is to linearize f at x^S, so that (1.1) becomes a linear program (let us neglect the difficulty coming from the constraint $x < c$). Now the right part of Fig. 1.1 shows that the solution of this LP may jump from one extreme point to another, called P and P' on the picture, if x^S moves to x'^S; and P' may be far from P, even if x^S and x'^S are close together. The effect of this *instability* on the convergence is disastrous, as is demonstrated in [22]: if f^* is the optimal value, we typically have $f(x^S) - f^* \simeq 1/S$.

Approximating each f_j to first order is thus not accurate enough, and this motivates methods of type (ii), based on second-order approximations. As for methods of type (iii), their motivation is the decomposable structure of (1.1). The resulting solution algorithm is made of two parts. One (minimizing the Lagrangian) treats each flow separately; the other (maximizing the Lagrangian dual) has a complexity depending only on the number n of arcs in the network. Now the aim of the present work is to introduce in this second part the second-order approximation that flow deviation is lacking. A similar idea was given quite recently by [1]; maximizing the dual by the ACCPM algorithm was then tremendously improved, both in terms of power (solving problems with n and m up to 10^5 and K up to 10^6) and of speed (CPU times divided by hundreds).

1.3 The proposed algorithm

The crucial ingredient for the methods of class (iii) is the algorithm maximizing the dual function. Here we do for bundle what [1] does for ACCPM, in a manner which can be explained as follows.

A standard approach to maximize a concave function – call it $\theta(u)$ – is *cutting planes* [3,10], in which θ is iteratively approximated by richer and richer polyhedral functions $\hat{\theta}$. These $\hat{\theta}$ are successively maximized; but this results in instabilities. A possible stabilization mechanism (the proximal bundle idea) appends to $\hat{\theta}$ a quadratic term centered at some "favored" iterate $\hat{u}$ (essentially the best current iterate).

Here, θ contains a well-isolated smooth component: $\theta = \Phi + \Pi$, where Φ is (concave and) twice differentiable, while Π is indeed polyhedral. We therefore use cutting planes to approximate Π *only*; stabilization is obtained thanks to the quadratic approximation of Φ at $\hat{u}$: a definitely natural quadratic term.

This approach can also be explained with no reference to the proximal paradigm. Maximizing a function $\theta(u)$ requires a *model* of θ, valid around the current iterate $\hat{u}$. Here, each component of θ has a natural model:

– the second-order quadratic approximation is best suited for the smooth function Φ, as in Newton's method;
– the cutting-plane approximation is natural for the polyhedral function Π, as in Kelley's method.

Our approach thus appears as quite natural, as it totally eliminates the need for a (somewhat artificial) proximal stabilization. By contrast, [1] keeps intact the (just as artificial) interior-point stabilization.

The paper is organized as follows. We propose in §2 a (classical) Lagrangian relaxation of (1.1) and in §3 our model of the dual function θ; it uses a second-

order oracle for Φ and an ordinary first-order oracle for Π. Our adaptation of the bundling technique to cope with this special model is the subject of §4, while §5 recalls the aggregation mechanism, important for primal recovery. The algorithm is detailed in §6, its convergence is established in §7, while §8 shows how to recover the primal optimal solution from the dual algorithm. Finally, §9 gives some numerical results and we conclude in §10 with a general discussion of our approach.

2 Lagrangian relaxation

Associating with the coupling constraints $\sum_{k=1}^{K} x^k = y$ the dual variables $u \in \mathbb{R}^n$, we define the Lagrangian

$$L(x,y,u) = \sum_{j=1}^{n} f_j(y_j) + \sum_{j=1}^{n} u_j\Big(-y_j + \sum_{k=1}^{K} x_j^k\Big)$$

and we apply Lagrangian relaxation, as explained for example in [14]. We minimize $L(\cdot,\cdot,u)$ for fixed u; here, this amounts to computing

$$\Phi_j(u_j) := \min_{0 \leqslant y_j < c_j} \{ f_j(y_j) - u_j y_j \}, \quad j = 1,\dots,n, \tag{2.1}$$

$$\Pi^k(u) := \min \big\{ u^\top x^k : \, Ax^k = b^k, \, x^k \geqslant 0 \big\}. \quad k = 1,\dots,K \tag{2.2}$$

(note that $\Phi_j = -f_j^*$, where f^* is the convex conjugate of the function f). It will be convenient for the sequel to use the notation

$$\Phi(u) := \sum_{j=1}^{n} \Phi_j(u_j), \qquad \Pi(u) := \sum_{k=1}^{K} \Pi^k(u).$$

The dual problem is then to maximize with respect to u the so-called *dual function*, namely: solve

$$\max_{u \in \mathbb{R}^n} \theta(u), \quad \text{where } \theta(u) := \Phi(u) + \Pi(u). \tag{2.3}$$

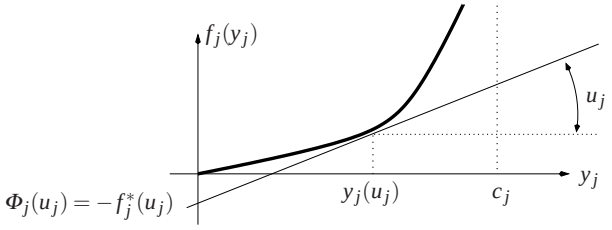

Fig. 2.1 Conjugating Kleinrock's function

In (2.1), the optimal y_j is easy to compute (see Fig. 1.2): we obtain

$$y_j(u_j) = \begin{cases} c_j - \sqrt{\frac{c_j}{u_j}} & \text{if } u_j \geqslant \frac{1}{c_j} \\ 0 & \text{otherwise ,} \end{cases} \tag{2.4}$$

so that Φ_j has the expression

$$\Phi_j(u_j) = \begin{cases} -\left(\sqrt{c_j u_j} - 1\right)^2 & \text{if } u_j \geqslant \frac{1}{c_j} \\ 0 & \text{otherwise .} \end{cases} \tag{2.5}$$

On the other hand, computing Π from (2.2) amounts to solving K independent shortest path problems, each of which being posed between s_k and t_k and having arc lengths u_j. The next simple result says that this computation has to be done with *positive* arc lengths only.

Proposition 2.1 *Consider the set*

$$U := \left\{ u \in \mathbb{R}^n : \; u_j \geqslant \frac{1}{c_j}, \; j = 1, \ldots, n \right\}. \tag{2.6}$$

For any $u \notin U$, there is $u' \in U$ such that $\theta(u') \geqslant \theta(u)$. As a result, (2.3) *is not changed if the constraint $u \in U$ is inserted.*

Proof Let $u \in \mathbb{R}^n$ be such that $u_{j_0} < 1/c_{j_0}$ for some j_0 and increase u_{j_0} until the value $1/c_{j_0}$. Because $x^k \geqslant 0$ in (2.2), each Π^k can only increase. As for the Φ_j's, only Φ_{j_0} can change; but (2.5) shows that it remains constantly 0. □

Thus, to ease the computation of the Π^k's, the constraints $u_j \geqslant 1/c_j$ may be inserted into (2.3): this does not prevent the computation of a dual optimal solution and does not change the optimal dual value.

3 Model of the dual function

Taking advantage of Proposition 2.1, we reformulate (2.3) as

$$\max_{u \in U} \theta(u) := \Phi(u) + \Pi(u) := \sum_{j=1}^{n} \Phi_j(u_j) + \sum_{k=1}^{K} \Pi^k(u). \tag{3.1}$$

To solve it, we propose a hybrid method working as follows:

– Each smooth function Φ_j is approximated by its second-order development, as in Newton's method. This development is made at a point – call it $\hat{u}$ – controlled according to its (dual) objective value $\theta(\hat{u})$.

– Each polyhedral function Π^k is approximated by cutting planes, as in Kelley's method [10,3].

In a way, the above method can be viewed as a bundle variant [16] (see also [1]), in which

– the cutting-plane paradigm is applied to a part of the (dual) objective function, namely Π,

– stabilization around $\hat{u}$ is obtained by the Newtonian term $u^\top \nabla^2 \Phi(\hat{u})u$, instead of an artificial $\|u\|^2$ weighted by a hard-to-tune penalty coefficient.

Since Lagrangian relaxation is column generation, our algorithm can also be viewed as a Dantzig-Wolfe variant where the masters are suitably stabilized.

At each iteration s, (2.2) is solved with the iterate u^s, provides a shortest path $x^k(u^s)$, which in turn provides an upper linearization: by definition, $\Pi^k(u) \leqslant u^\top x^k(u^s)$ for all $u \in \mathbb{R}^n$. Accumulating these shortest paths, we form at the current iteration S the K polyhedral functions

$$\hat{\Pi}^k(u) := \min_{s=1,\dots,S} u^\top x^k(u^s) \geqslant \Pi^k(u), \quad \text{for all } u \in \mathbb{R}^n. \tag{3.2}$$

As for the Φ_j's, note that they have analytic derivatives over the feasible domain:

$$\Phi_j'(u_j) = \sqrt{\frac{c_j}{u_j}} - c_j, \quad \Phi_j''(u_j) = \frac{-1}{2u_j}\sqrt{\frac{c_j}{u_j}} \qquad \text{if } u_j \geqslant \frac{1}{c_j} \tag{3.3}$$

and that $-\Phi_j'(u_j) = y_j(u_j)$ is the optimal y_j of (2.4).

In addition to the $\hat{\Pi}^k$'s, suppose also that a *stability center* $\hat{u} \in U$ is available at the current iteration. Analogously to Π^k, each Φ_j will be replaced by its quadratic approximation near $\hat{u}$:

$$\tilde{\Phi}_j(u_j) := \Phi_j(\hat{u}_j) - y_j(\hat{u}_j)(u_j - \hat{u}_j) + \frac{1}{2}M_j(u_j - \hat{u}_j)^2 \quad [\simeq \Phi_j(u_j)], \tag{3.4}$$

where $y_j(\hat{u}_j) = -\Phi_j'(\hat{u}_j)$ is given by (2.4) and $M_j := \Phi_j''(\hat{u}_j)$ by (3.3).

We will use the notation

$$\tilde{\Phi}(u) := \sum_{j=1}^n \tilde{\Phi}_j(u_j), \quad \hat{\Pi}(u) := \sum_{k=1}^K \hat{\Pi}^k(u), \quad \hat{\theta}(u) := \tilde{\Phi}(u) + \hat{\Pi}(u)$$

and the essential part of an iteration will be to maximize $\hat{\theta}$, which can be viewed as a *model* of the true dual function θ in (3.1). We also find it convenient to use the change of variable $h = u - \hat{u}$: we solve

$$\max\left\{\tilde{\Phi}(\hat{u}+h) + \hat{\Pi}(\hat{u}+h) : \hat{u}+h \geqslant 1/c\right\}.$$

Introducing additional variables π^k (connoting $\hat{\Pi}^k$), this is the quadratic programming problem

$$\begin{aligned} &\max_{h,\pi} \left\{\tilde{\Phi}(\hat{u}+h) + \sum_{k=1}^K \pi^k\right\} \\ &\text{s.t.} \quad \pi^k \leqslant (\hat{u}+h)^\top x^k(u^s), \quad \text{for } \begin{cases} k = 1,\dots,K, \\ s = 1,\dots,S, \end{cases} \\ &\qquad h_j \geqslant \frac{1}{c_j} - \hat{u}_j, \quad j = 1,\dots,n. \end{aligned} \tag{3.5}$$

Note that the somewhat artificial constraint $\hat{u} + h = u \geqslant 1/c$ is useful for a fast computation of $\Pi^k(u)$ in (2.2); but it is even more useful for the dual algorithm:

from (2.5), $\Phi_j''(u_j) = 0$ if $u_j < 1/c_j$. If such a u is a $\hat{u}$, then $\tilde{\Phi}_j$ will degenerate and (3.5) will perhaps be unbounded from above[1].

Proposition 3.1 *Problem* (3.5) *has a unique optimal solution* $(\hat{h}, \hat{\pi})$, *with* $\hat{\pi}^k = \hat{\Pi}^k(\hat{u}+\hat{h})$.

Proof From standard convex analysis, each function $\hat{\Pi}^k$ is concave; and each $\tilde{\Phi}_j$ is a strictly concave quadratic function (from (3.3), $M_j < 0$!): $\hat{\theta}$ has a unique maximum $\hat{u}+\hat{h}$, making up the h-part of the optimal solution in (3.5); and each π^k has to reach its maximal value, namely $\hat{\Pi}^k(\hat{u}+\hat{h})$. □

Note to conclude this section that our approach can of course be applied to the maximization of any concave function θ made up of two parts: a polyhedral one (given by an oracle) and a smooth one (whose Hessian is at hand). In (3.4), M_j is the jj^{th} entry of the Hessian (here diagonal) of the smooth part of θ.

4 Ascent steps, null steps and backtracking

The resolution of (3.5) *predicts* an increase

$$\delta := \hat{\theta}(\hat{u}+\hat{h}) - \theta(\hat{u}) \tag{4.1}$$

in the dual objective function. Of course, $\delta \geqslant 0$ since $\theta(\hat{u}) \leqslant \hat{\theta}(\hat{u}) \leqslant \hat{\theta}(\hat{u}+\hat{h})$. Besides, we will see in Proposition 5.3 that δ is an optimality measure of $\hat{u}$: it is natural to stop the algorithm if δ is small. So δ is indeed positive unless the algorithm is about to stop.

Standard bundle compares the actual increase $\theta(\hat{u}+\hat{h}) - \theta(\hat{u})$ to δ; if it is deemed insufficient, (3.5) is solved again with an enriched model; this is the *bundling* process. For reasons that will soon become apparent, this technique needs amendment. In our variant, the candidate for the next iteration is not the output $\hat{u}+\hat{h}$ from (3.5) but rather $u^+ := \hat{u}+t\hat{h}$, for some suitable stepsize $t \in]0,1]$ and a sufficient increase is quantified by

$$[\theta(u^+) =]\ \theta(\hat{u}+t\hat{h}) \geqslant \theta(\hat{u}) + \kappa t\delta, \tag{4.2}$$

$\kappa \in]0,1[$ being a fixed tolerance. If (4.2) holds, $\hat{u}$ can safely be moved to the definitely better point u^+; iteration S is terminated, this is an *ascent step* in the bundle terminology. Theorem 7.3 will show that infinitely many such updates do imply convergence of the algorithm.

[1] This difficulty can be eliminated, though. Observe in (1.1) that the constraint $y \geqslant 0$ is redundant; each f_j of (1.2) can therefore be extended as we like on $\mathbb{R}_-$. A convenient extension is

$$f_j(y_j) := \frac{y_j^2}{c_j^2} + \frac{y_j}{c_j} \quad \text{for } y_j \leqslant 0,$$

which is C^2 and strongly convex; its conjugate enjoys the same properties and the algorithm can work.

Now assume (4.2) does not hold. Because $\hat{\theta}$ is concave,

$$\begin{aligned}\hat{\theta}(u^+) &\geqslant (1-t)\hat{\theta}(\hat{u}) + t\hat{\theta}(\hat{u}+\hat{h}) \\ &\geqslant (1-t)\theta(\hat{u}) + t\hat{\theta}(\hat{u}+\hat{h}) \\ &= \theta(\hat{u}) + t\delta\,.\end{aligned}$$

Failure of (4.2) therefore implies $\hat{\theta}(u^+) > \theta(u^+) + (1-\kappa)t\delta$; this means that $\hat{\theta}$ approximates θ badly. Then we have to decide which of the approximations $\tilde{\Phi}$ and $\hat{\Pi}$ needs improvement.

Improvement of $\hat{\Pi}$ is done by the bundling process, which appends in the definition of $\hat{\Pi}$ the new data coming from the oracle (2.2), called at $\hat{u}+t\hat{h}$. However, bundling will be of no avail if $(\hat{h},\hat{\pi})$ is still feasible in the next QP (3.5) – see Lemma 7.4 below. To avoid an infinite loop, a sufficient improvement must be required on the next $\hat{\Pi}$; this is the whole business of a bundle method. We quantify the required improvement as

$$\Pi(\hat{u}+t\hat{h}) \leqslant \Pi(\hat{u}) + t[\hat{\Pi}(\hat{u}+\hat{h}) - \Pi(\hat{u})] - \kappa' t\delta, \tag{4.3}$$

$\kappa' \in]0,\kappa]$ being another positive tolerance (see Fig. 4.1). By concavity of $\hat{\Pi}$, this implies that $\Pi(u^+)$ is "substantially" smaller than $\hat{\Pi}(u^+)$; and note that the next $\hat{\Pi}$ is going to have the value $\Pi(u^+)$ at u^+. If (4.3) holds, $\hat{u}$ is kept as it is; again iteration S is terminated, this is a *null step*.

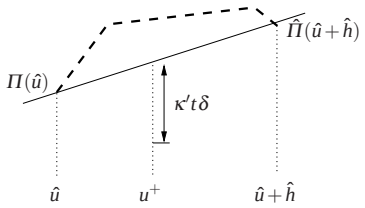

Fig. 4.1 The new linearization passes under $\hat{\Pi}(\hat{u}+t\hat{h})$

When (4.3) does not hold, $\hat{\Pi}(u^+)$ can be considered as a good approximation of $\Pi(u^+)$; so if (4.2) does not hold, it is $\tilde{\Phi}$ that approximates Φ badly. To improve it, we decrease t, compute the new value of θ and test (4.2), (4.3) again; we will make sure in Lemma 7.1 below that this *backtracking phase* cannot go forever.

Let us summarize this section. An iteration of our variant solves (3.5) and tests $u^+ = \hat{u}+t\hat{h}$, with three possible outputs:

– If (4.2) holds, an ascent step is made: $\hat{u}$ is moved to u^+ and the model $\hat{\Pi}$ is updated;
– if (4.2) does not hold but (4.3) holds, a null-step is made: $\hat{u}$ is kept as it is and $\hat{\Pi}$ is updated;
– if neither (4.2) nor (4.3) holds, a backtracking is made: t is decreased and the oracle (2.2) is called at the new u^+, which is closer to $\hat{u}$.

Standard bundle has no backtracking: it merely uses $t=1$ and overlooks (4.3). Actually, if we had $\theta = \Pi$ and $\hat{\theta} = \hat{\Pi}$, (4.3) could not hold when (4.2) does not hold. In the present variant, this argument is destroyed by the $\tilde{\Phi}$-part of $\hat{\theta}$.

5 Toward primal recovery: the aggregate linearization

The necessary material to state the dual algorithm is now available. However, remember that our problem is rather (1.1) than (2.3). To establish the connection between the two resolutions, we need some more sophisticated material from convex analysis. First we introduce some notation.

- The normal cone $\mathrm{N}_U(u)$ is the set of vectors $v \in \mathbb{R}^n$ such that $(v-u)^\top v \leqslant 0$ for all $v \in U$;
- $y(\hat{u}) \in \mathbb{R}^n$ will be the vector whose components are $y_j(\hat{u}_j)$, see (2.4);
- $M := \nabla^2 \Phi(\hat{u})$ will be the diagonal (negative definite) $n \times n$ matrix whose jjth element is $M_j = \Phi_j''(\hat{u}_j)$ of (3.4);
- the unit simplex of $\mathbb{R}^S$ will be $\Delta^S := \{\alpha \in \mathbb{R}^S : \sum_s \alpha^s = 1,\ \alpha \geqslant 0\}$.

Now we recall some elementary subdifferential calculus. Denote by

$$\partial\theta(u) := -\partial(-\theta)(u) = \{g \in \mathbb{R}^n : \theta(v) \leqslant \theta(u) + (v-u)^\top g \ \text{ for all } v \in \mathbb{R}^n\}$$

the "superdifferential" of the concave function θ at u.

- The superdifferential of the smooth concave function $\tilde{\Phi}$ is its gradient: $\partial\tilde{\Phi}(u) = -y(\hat{u}) + M(u-\hat{u})$;
- the superdifferential of $\hat{\Pi}^k$ is the convex hull of the active slopes in (3.2):

$$\partial\hat{\Pi}^k(u) = \Big\{\hat{x}^k = \textstyle\sum_{s=1}^S \alpha^s x^k(u^s) : \alpha \in \Delta^S, \\ \alpha^s = 0 \ \text{ if } \ u^\top x^k(u^s) > \hat{\Pi}^k(u)\Big\}; \tag{5.1}$$

- the superdifferential of $\hat{\theta}$ is the sum of superdifferentials:

$$\partial\hat{\theta}(u) = -y(\hat{u}) + M(u-\hat{u}) + \sum_{k=1}^K \partial\hat{\Pi}^k(u)\,.$$

This allows us to describe the solution of (3.5):

Proposition 5.1 *The unique optimal solution $\hat{h}$ of* (3.5) *is characterized as follows: for some $\hat{x}^k \in \partial\hat{\Pi}^k(\hat{u}+\hat{h})$, $k = 1,\ldots,K$ and $v \in \mathrm{N}_U(\hat{u}+\hat{h})$,*

$$\hat{h} = M^{-1}\hat{g}, \quad \textit{where} \quad \hat{g} := y(\hat{u}) - \hat{x} + v,\ \hat{x} := \sum_{k=1}^K \hat{x}^k \in \partial\Pi(\hat{u}+\hat{h}). \tag{5.2}$$

Proof Watching for various changes of sign, apply the optimality condition [9, Thm. VII.1.1.1(iii)]: there is some supergradient in $\partial\hat{\theta}(\hat{u}+\hat{h})$ lying in $\mathrm{N}_U(\hat{u}+\hat{h})$. In view of the above-mentioned calculus rules, this writes

$$-y(\hat{u}) + M\hat{h} + \sum_{k=1}^K \hat{x}^k = v,$$

which is just (5.2). □

With the particular form of U, the property $v \in \mathrm{N}_U(\hat{u}+\hat{h})$ means that $v \leqslant 0$ is in complementarity with $\hat{u}+\hat{h}-1/c \geqslant 0$.

Note that each $\hat{x}^k$ is a convex combination as described in (5.1). To make up $\hat{x}$, one needs K sets of convex multipliers $\alpha^k \in \Delta^S$, which indeed are the KKT multipliers (not necessarily unique) associated with the constraint involving π^k in (3.5); any reasonable QP solver computes them, in addition to the optimal $(\hat{h}, \hat{\pi})$. In the next statement, this remark could also be used for an alternative proof, based on complementarity slackness:

Lemma 5.2 *With the notation of Proposition 5.1, $\hat{\Pi}^k(u) \leqslant u^\top \hat{x}^k$ for all $u \in \mathbb{R}^n$. Equality holds for $u = \hat{u}+\hat{h}$. In particular, $\hat{\Pi}(\hat{u}+\hat{h}) = (\hat{u}+\hat{h})^\top \hat{x}$.*

Proof Apply (5.1) with $u = \hat{u}+\hat{h}$: $\hat{x}^k = \sum_s \alpha^s x^k(u^s)$ for some $\alpha \in \Delta^S$. The required inequality is therefore clear from the definition (3.2) of $\hat{\Pi}^k$. Besides, this convex combination involves only indices s such that $(\hat{u}+\hat{h})^\top x^k(s) = \hat{\Pi}^k(\hat{u}+\hat{h})$, so the stated equality holds as well; and the last statement follows by summation over k. □

The whole business of dual convergence will be to drive δ to 0, and this has interesting consequences:

Proposition 5.3 *With the notation of Proposition 5.1, $\delta = \delta_h + \delta_x + \delta_v$, where*

$$\delta_h := -\frac{1}{2}\hat{h}^\top M \hat{h} \geqslant 0\,, \quad \delta_x := \hat{u}^\top \hat{x} - \Pi(\hat{u}) \geqslant 0\,, \quad \delta_v := \hat{h}^\top v \geqslant 0. \tag{5.3}$$

Besides, for all $u \in U$,

$$\theta(u) \leqslant \theta(\hat{u}) + \delta_x + \delta_v - (u-\hat{u})^\top \hat{g}. \tag{5.4}$$

Proof Write the definition (4.1) of δ, using (3.4) and Lemma 5.2:

$$\begin{aligned}\delta &= \Phi(\hat{u}) - \hat{h}^\top y(\hat{u}) + \tfrac{1}{2}\hat{h}^\top M \hat{h} + (\hat{u}+\hat{h})^\top \hat{x} - \Phi(\hat{u}) - \Pi(\hat{u}) \\ &= \tfrac{1}{2}\hat{h}^\top M \hat{h} + \left[\hat{u}^\top \hat{x} - \Pi(\hat{u})\right] - \hat{h}^\top (y(\hat{u}) - \hat{x})\end{aligned}$$

and (5.3) follows because $y(\hat{u}) - \hat{x} = M\hat{h} - v$ from (5.2).

Then remember from (3.3) that M is negative semi-definite: $\delta_h \geqslant 0$. The property $\delta_x \geqslant 0$ comes from Lemma 5.2; and $\delta_v = (\hat{u}+\hat{h}-\hat{u})^\top v$ is nonnegative because $v \in \mathrm{N}_U(\hat{u}+\hat{h})$.

Now take an arbitrary $u \in U$. Using feasibility of $\hat{x}^k$ in (2.2), concavity of Φ and definition of normal cones,

$$\begin{aligned}\Pi(u) &\leqslant u^\top \hat{x} = \hat{u}^\top \hat{x} + (u-\hat{u})^\top \hat{x} \\ \Phi(u) &\leqslant \Phi(\hat{u}) - (u-\hat{u})^\top y(\hat{u}) \\ 0 &\leqslant (\hat{u}+\hat{h}-u)^\top v = (\hat{u}-u)^\top v + \hat{h}^\top v.\end{aligned}$$

Summing up and disclosing appropriate δ-values:

$$\theta(u) \leqslant \Pi(\hat{u}) + \delta_x + \Phi(\hat{u}) + (u-\hat{u})^\top(-\hat{g}) + \delta_v,$$

which is just (5.4). □

Thus, when δ is small, δ_h, δ_x and δ_v are small. If M behaves itself, $\|\hat{g}\|$ is also small and (5.4) shows that $\hat{u}$ is approximately optimal in (3.1).

6 The algorithm

We are now in a position to state the algorithm. Knowing the expression of the Kleinrock function, it works with the help of the "oracle" solving (2.2) for given $u \geqslant 1/c$. It uses the improvement parameters κ and κ' satisfying $0 < \kappa' \leqslant \kappa < 1$, and the stopping tolerance $\underline{\delta} \geqslant 0$. The starting point $u^1 \in U$ is given, as well as the initial shortest paths $x^k(u^1)$ forming the initial bundle, and the initial quadratic model $\tilde{\Phi}$.

Algorithm 6.1 (Combined Newton-cutting-plane algorithm (`Ncp`)) Initialize $S = 1$, $\hat{u} = \hat{u}^1 = u^1$.

STEP 1 (*Trial point finding*). Find $\hat{h}$, $\hat{x} = \hat{x}^S$ and $\hat{g} = \hat{g}^S$ as described by Proposition 5.1. Compute $\delta = \delta^S$ by (4.1)

STEP 2 (*Stopping test*). If $\delta^S \leqslant \underline{\delta}$ stop, returning $\hat{u}$ and $\hat{x}$.

STEP 3 (*Line-search*). Set $t = 1$.

STEP 3.1 (*Oracle call*). Set $u^+ := \hat{u} + t\hat{h}$. Compute $x^k(u^+)$ from (2.2) and the resulting values $\Pi(u^+)$, $\theta(u^+)$.

STEP 3.2 (*Ascent test*). If (4.2) holds, set $\hat{u}^{S+1} = u^+$; update the quadratic approximation $\tilde{\Phi}$.
Go to Step 4.

STEP 3.3 (*Null-test*). If (4.3) holds, set $\hat{u}^{S+1} = \hat{u}^S$.
Go to Step 4.

STEP 3.4 (*Interpolation*). Select a new t "well inside" the segment $]0,t[$.
Go to Step 3.1.

STEP 4 (*Bundle updating and loop*). For $k = 1,\ldots,K$, append $x^k(u^+)$ obtained in Step 3.1 to the bundle. Increase S by 1 and go to Step 1. □

In Step 3.4, the simplest is to divide t by 2. More sophisticated interpolation formulae can be designed, in the spirit of cubic fitting in NLP. The expression "well inside" can mean for example "in the segment $[0.1t,\ 0.9t]$": the new t should be not too close to the old t for obvious reasons, but also not too close to 0 for convergence of the overal algorithm.

Remark 6.2 (Sum of max vs. max of sum) Our approximation of Π uses K *individual* approximations $\hat{\Pi}^k$ of (3.2). Traditional bundle methods actually ignore the summation property $\Pi = \sum_k \Pi^k$: they use just one supergradient, say $\xi^s \in \partial\Pi(u^s)$ for each u^s, corresponding to the *compound* linearization $u^\top \xi^s$ of $\Pi(u)$. Here, ξ^s is of course the sum of the shortest paths $x^k(u^s)$.

Storing S linearizations needs Sn elements (as opposed to the KSn elements needed here). Besides, the S^{th} compound quadratic problem (3.5) simplifies to

$$\begin{aligned} &\max_{u,\pi} \left\{ \tilde{\Phi}(u) + \pi \right\} \\ &\text{s.t.} \quad \pi \leqslant u^\top \sum_{k=1}^{K} x^k(u^s), \quad \text{for } s = 1,\ldots,S, \\ &\qquad u_j \geqslant \tfrac{1}{c_j}, \quad j = 1,\ldots,n, \end{aligned}$$

which has just S linking constraints.

Even with the above simplification, Algorithm 6.1 needs potentially infinite memory. However, traditional bundle methods make use of the aggregate linearization $\hat{x}$ revealed by Proposition 5.1: a "minimal" variant would approximate Π at iteration $S+1$ by a polyhedral function made up of two pieces only, namely

$$\min\left\{u^\top \hat{x}^S, u^\top \xi^{S+1}\right\}.$$

Needless to say, these simplifications correspond to less accurate descriptions of Π, and are therefore paid by dual iterates u^+ of probably lesser quality. □

7 Dual convergence

In this section, we pretend that $\underline{\delta} = 0$ in Algorithm 6.1. Then we prove that $\liminf \delta^S = 0$; this implies that the algorithm will eventually stop if $\underline{\delta} > 0$. We use the terminology introduced in §5. First we make sure that each backtracking phase terminates.

Lemma 7.1 *Assume $\kappa + \kappa' \leqslant 1$; let $-L \leqslant -\ell < 0$ be lower and upper bounds on the eigenvalues of M over the segment $[\hat{u}, \hat{u}+\hat{h}]$. Then* (4.2) *or* (4.3) *holds (or both) whenever $t \leqslant \ell/L$.*

Proof Suppose that neither (4.3) nor (4.2) holds. Subtracting and using definitions:

$$\begin{aligned}\Phi(\hat{u}+t\hat{h}) &< \Phi(\hat{u}) - t[\hat{\Pi}(\hat{u}+\hat{h}) - \Pi(\hat{u})] + (\kappa+\kappa')t\delta \\ &= \Phi(\hat{u}) + t[\tilde{\Phi}(\hat{u}+\hat{h}) - \Phi(\hat{u})] + (\kappa+\kappa'-1)t\delta \\ &\leqslant \Phi(\hat{u}) + t[\tilde{\Phi}(\hat{u}+\hat{h}) - \Phi(\hat{u})] \\ &= \Phi(\hat{u}) + t\left[-y(\hat{u})^\top \hat{h} + \tfrac{1}{2}\hat{h}^\top M \hat{h}\right].\end{aligned}$$

Apply some mean-value theorem to Φ: for example, denoting by $\tilde{M}$ the Hessian of Φ at some point between $\hat{u}$ and $\hat{u}+t\hat{h}$

$$\Phi(\hat{u}+t\hat{h}) = \Phi(\hat{u}) - ty(\hat{u})^\top \hat{h} + \frac{t^2}{2}\hat{h}^\top \tilde{M}\hat{h},$$

so that

$$-ty(\hat{u})^\top \hat{h} + \frac{t^2}{2}\hat{h}^\top \tilde{M}\hat{h} < t\left[-y(\hat{u})^\top \hat{h} + \frac{1}{2}\hat{h}^\top M \hat{h}\right].$$

Divide by $t > 0$ and simplify to obtain

$$-tL\|\hat{h}\|^2 \leqslant t\hat{h}^\top \tilde{M}\hat{h} < \hat{h}^\top M \hat{h} \leqslant -\ell\|\hat{h}\|^2 < 0.$$ □

Establishing convergence of a bundle method amounts to proving two distinct results:

– If infinitely many ascent steps are performed, the stability centers $\hat{u}$ form a maximizing sequence of θ.
– If the sequence of stability centers stops at some $\hat{u}$, then this $\hat{u}$ maximizes θ (possibly infinitely many null-steps being needed to prove this property).

These results are proved respectively in the next two sections.

7.1 Case of infinitely many ascent steps

To simplify our analysis, we will assume here that (1.1) is feasible. This guarantees the Slater property, which is a key for an appropriate primal-dual behaviour:

Lemma 7.2 *If* (1.1) *has a feasible point, then* θ *is sup-compact on* U*: for each* $z \in \mathbb{R}$*, the set of* $u \in U$ *such that* $\theta(u) \geqslant z$ *is (closed and) bounded. As a result,* (3.1) *has a unique solution.*

Proof Closedness classically follows from upper semi-continuity of a dual function. Let $\mathring{x}$ and $\mathring{y} = \sum_k \mathring{x}^k$ make a feasible point. Because each $\mathring{y}_j < c_j$, we can find $\varepsilon > 0$ and $B > 0$ such that

$$\text{for } j = 1, \ldots, n, \quad \mathring{y}_j \leqslant y_j \leqslant \mathring{y}_j + \varepsilon \quad \Longrightarrow \quad f_j(y_j) \leqslant B.$$

Take an arbitrary $u \in U \subset \mathbb{R}^n_+$ and set $y^u := \mathring{y} + \varepsilon \frac{u}{\|u\|}$. By definition of the dual function,

$$\theta(u) \leqslant L(\mathring{x}, y^u, u) = f(y^u) + u^\top \Big(-\mathring{y} - \varepsilon \frac{u}{\|u\|} + \sum_{k=1}^{K} \mathring{x}^k \Big) = f(y^u) - \varepsilon \|u\|;$$

but $0 \leqslant y^u_j \leqslant \mathring{y}_j + \varepsilon$ for each j ($0 \leqslant u_j \leqslant \|u\|$!), hence $f(y^u) \leqslant nB$. We have proved that $z \leqslant \theta(u)$ implies $z \leqslant nB - \varepsilon\|u\|$.

Thus, $\theta(u) \to -\infty$ when $\|u\| \to +\infty$ in U. This classically implies that (3.1) has at least one optimal solution. Finally, this solution is unique because of strict concavity: Π is concave and (3.3) shows that Φ is strictly concave. □

Sup-compactness classically eliminates the duality gap and allows the characterization of primal-dual solutions via the superdifferential of the dual function. We will recover these results in a constructive way, by establishing appropriate convergence properties of the sequences $\hat{u}$ and $(y(\hat{u}), \hat{x})$ (the latter being established in §8 below).

Theorem 7.3 *Assume that* (1.1) *has a feasible point and let Algorithm 6.1 generate an infinite sequence* $\mathscr{S}$ *of ascent steps. Then the subsequences* $(\delta^s)_{\mathscr{S}}$, $(\hat{h}^s)_{\mathscr{S}}$ *and* $(\hat{g}^s)_{\mathscr{S}}$ *tend to 0; and the sequence* $\hat{u}^s$ *tends to the optimal solution of* (3.1)[2].

Proof The increasing sequence $\theta(\hat{u}^s)$ has a limit, not larger than the optimal value $\overline{\theta}$ of (3.1). From Lemma 7.2, the sequence $\hat{u}^s$ is bounded; $L > 0$ [resp. $\ell > 0$] of Lemma 7.1 is bounded from above [resp. away from 0], t is bounded away from 0: say $t \geqslant \underline{t} > 0$. Then we have from (4.2)

$$\begin{aligned} &\theta(\hat{u}^{s+1}) \geqslant \theta(\hat{u}^s) + \kappa \underline{t} \delta^s \quad \text{if } s \in \mathscr{S} \\ &\theta(\hat{u}^{s+1}) = \theta(\hat{u}^s) \qquad\quad \text{otherwise} \end{aligned}$$

and we obtain by summation $\sum_{s \in \mathscr{S}} \delta^s \leqslant [\overline{\theta} - \theta(u^1)]/\kappa \underline{t}$: $(\delta^s)_{\mathscr{S}}$ tends to 0. The three components of δ^s in (5.3) tend to 0 as well, and this is also true of the subsequences $(\hat{g}^s)_{\mathscr{S}}$ and $(\hat{h}^s)_{\mathscr{S}}$.

Then take a cluster point of $\hat{u}^s$ and pass to the limit in (5.4): this cluster point has to be the unique optimal solution of (3.1). □

[2] Note that the whole sequence $\hat{u}^s$ coincides with $(\hat{u}^s)_{\mathscr{S}}$, since $\hat{u}^{s+1} = \hat{u}^s$ if $s \notin \mathscr{S}$.

Note that, if (1.1) has no feasible point, then $\theta(\hat{u})$ will typically tend to $+\infty$; δ has no reason to tend to 0, the stop in Algorithm 6.1 will never occur. To prevent this situation, it is wise to insert an "emergency stop" when $\theta(\hat{u})$ is unduly large.

7.2 Case of finitely many ascent steps

First we show that the next QP will modify the present $\hat{h}$ (remember from Proposition 3.1 that $\hat{\Pi}(\hat{u}+\hat{h}) = \hat{\pi}$):

Lemma 7.4 *If* (4.3) *holds, the new linearization* $x(\hat{u}+t\hat{h})$ *satisfies*

$$(\hat{u}+\hat{h})^\top x(\hat{u}+t\hat{h}) \leqslant \hat{\Pi}(\hat{u}+\hat{h}) - \kappa'\delta. \tag{7.1}$$

Proof Use simplified notation: with $u^+ = \hat{u}+t\hat{h}$, set $x^+ := x(u^+)$ and $z := (\hat{u}+\hat{h})^\top x^+$.

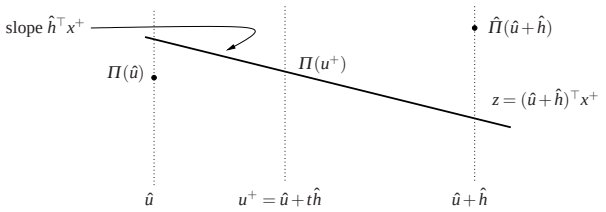

Fig. 7.1 The new linearization passes under $\hat{\Pi}(\hat{u}+\hat{h})$

Because $x^+ \in \partial\Pi(u^+)$, we have by definition $\Pi(\hat{u}) \leqslant \Pi(u^+) - t\hat{h}x^+$; hence $\hat{h}^\top x^+ \leqslant [\Pi(u^+) - \Pi(\hat{u})]/t$, so that

$$z = \Pi(u^+) + (1-t)\hat{h}^\top x^+ \leqslant \Pi(u^+) + \tfrac{1-t}{t}[\Pi(u^+) - \Pi(\hat{u})] = \tfrac{1}{t}\Pi(u^+) - \tfrac{1-t}{t}\Pi(\hat{u}).$$

Now use (4.3) to bound $\Pi(u^+)$:

$$z \leqslant \frac{1}{t}\Pi(\hat{u}) + \hat{\Pi}(\hat{u}+\hat{h}) - \Pi(\hat{u}) - \kappa'\delta - \frac{1-t}{t}\Pi(\hat{u}),$$

which is just (7.1). □

The proof of the next result uses explicitly the fact that *all* linearizations are stored in the bundle: to accommodate the bundle compression alluded to at the end of Remark 6.2, a more sophisticated proof would be required, along the lines of [9, Thm. XV.3.2.4].

Theorem 7.5 *Suppose the stability center stops at some iteration S:* $\hat{u}^{s+1} = \hat{u}^s$ *for all* $s \geqslant S$. *Then* $\delta^s \to 0$ *and* $\hat{u}^S$ *is the optimal solution of* (3.1).

Proof The situation is as follows: at all iterations s following S, $\hat{u} = \hat{u}^S$, M and $y(\hat{u})$ are fixed; $\delta = \delta^s$ forms a nonincreasing sequence since (3.5) has more and more constraints; Proposition 5.3 then guarantees that $\hat{h} = \hat{h}^s$ is bounded. It follows that $u^{s+1} = \hat{u} + t^s h^s$ is also bounded, as lying in the segment $[\hat{u}, \hat{u} + \hat{h}^s]$; hence $x(u^s) \in \partial\Pi(u^s)$ is bounded ([9, Prop. VI.6.2.2]).

Write (7.1) and the definition (3.2) of $\hat{\Pi}$ at the s^{th} iteration: for all $s > S$ and all $r \leqslant s$,

$$(\hat{u} + \hat{h}^s)^\top x(u^{s+1}) + \kappa' \delta^s \leqslant \hat{\Pi}(\hat{u} + \hat{h}^s) \leqslant (\hat{u} + \hat{h}^s)^\top x(u^r),$$

so that

$$\kappa' \delta^s \leqslant (\hat{u} + \hat{h}^s)^\top [x(u^r) - x(u^{s+1})] \leqslant B \|x(u^r) - x(u^{s+1})\|,$$

where we have used the Cauchy-Schwarz inequality and B is a bound for $\|\hat{u} + \hat{h}^s\|$. Now assume $\delta^s \geqslant \varepsilon > 0$ for all s. Then

$$\|x(u^r) - x(u^{s+1})\| \geqslant \frac{\kappa' \varepsilon}{B} \quad \text{for all } s > S \text{ and all } r \leqslant s.$$

In words: around each $x(u^r)$, there is a ball of fixed radius $\kappa' \varepsilon / B$ which cannot contain any other x; because the x's are confined in a bounded set, this is impossible.

It follows that the monotone sequence δ^s tends to 0, pass to the limit in (5.4) to establish optimality of $\hat{u}$. □

8 Primal recovery

It is known that convergence of the dual algorithm has its counterpart concerning the primal problem. However, we solve here (3.1), while the dual of (1.1) is rather (2.3). The issue is therefore more delicate, especially when infinitely many ascent steps are performed; we analyze this case first.

Theorem 8.1 *Make the assumptions of Theorem 7.3. Then:*
– *the subsequence* $\{y(u^s)\}_{s \in \mathscr{S}}$ *tends to the unique y-optimal solution of* (1.1)*;*
– *for* $k = 1, \ldots, K$, *the subsequences* $\{\hat{x}^{k,s}\}_{s \in \mathscr{S}}$ *are bounded and any of their cluster points makes up an x-optimal solution of* (1.1)*.*

Proof We already know from Theorem 7.3 that $(\delta^s)_{\mathscr{S}}$, $(\hat{h}^s)_{\mathscr{S}}$ and $(\hat{g}^s)_{\mathscr{S}}$ tend to 0. We also know that $\hat{u}^s$ has a limit $\bar{u}$; therefore $y(\hat{u}^s) \to y(\bar{u})$ and we proceed to prove that $(\hat{x}^s)_{\mathscr{S}} \to y(\bar{u})$. Note that $(\hat{u}^s + \hat{h}^s)_{\mathscr{S}} \to \bar{u}$.

Define the set $J^* := \{j = 1, \ldots, n : \bar{u}_j = 1/c_j\}$ of artificial constraints that are active at $\bar{u}$.
– For $j \notin J^*$, $\bar{u}_j > 1/c_j$ so that $\hat{u}^s_j + \hat{h}^j_j > 1/c_j$ for $s \in \mathscr{S}$ large enough, The property $v^s \in \mathrm{N}_U(\hat{u}^s + \hat{h}^s)$ therefore implies $v^s_j = 0$, hence $\hat{x}^s_j = y_j(\hat{u}^s) - \hat{g}^s_j$ tends to $y_j(\bar{u})$.
– For $j \in J^*$, $y_j(\bar{u}) = 0$; hence $y_j(\hat{u}^s) \to 0$ and $\hat{x}^s_j \to 0$ because, from (5.2),

$$0 \leqslant \hat{x}^s_j = y_j(\hat{u}^s) - \hat{g}^s_j + v^s_j \leqslant y_j(\hat{u}^s) - \hat{g}^s_j \to 0.$$

Piecing together, we see that

$$(\hat{x}^s - y(\hat{u}^s))_{\mathscr{S}} \to 0. \tag{8.1}$$

Now write

$$\theta(\hat{u}^s) = \Phi(\hat{u}^s) + \Pi(\hat{u}^s) = f(y(\hat{u}^s)) - (\hat{u}^s)^\top y(\hat{u}^s) + \Pi(\hat{u}^s)$$

and pass to the limit for $u \in \mathscr{S}$:

$$\theta(\bar{u}) = f(y(\bar{u})) - \bar{u}^\top y(\bar{u}) + \Pi(\bar{u}).$$

But observe from (8.1) that

$$-\bar{u}^\top y(\bar{u}) + \Pi(\bar{u}) = \lim_{s\in\mathscr{S}} \left[-(\hat{u}^s)^\top \hat{x}^s + \Pi(\hat{u}^s)\right] = \lim_{s\in\mathscr{S}} \delta_x^s$$

where we have used the notation of Proposition 5.3. Since $(\delta_x^s)_{\mathscr{S}} \to 0$,

$$\theta(\bar{u}) = f(y(\bar{u})). \tag{8.2}$$

Finally, the convergent sequence $(\hat{x}^s)_{\mathscr{S}}$ is bounded. Being nonnegative and summing up to $(\hat{x}^s)_{\mathscr{S}}$, the subsequences $(\hat{x}^{k,s})_{\mathscr{S}}$ are also bounded. Consider a cluster point: say, with $\mathscr{S}' \subset \mathscr{S}$, $(\hat{x}^{k,s})_{\mathscr{S}'} \to \bar{x}^k$ for $k = 1,\ldots,K$. The $\bar{x}^k$'s are feasible in (1.1) and they sum up to $y(\bar{u})$: $(\bar{x}, y(\bar{u}))$ makes a feasible point in (1.1). In view of (8.2), weak duality tells us that this point is primal optimal. □

The case of finitely many ascent steps is just easier, as $\hat{u}^s$ reaches its limit $\bar{u}$ for some finite s.

Theorem 8.2 *Suppose that the stability center stops at some iteration S. Then the conclusions of Theorem 8.1 hold, with $\mathscr{S}$ replaced by the whole sequence $S+1,\ldots$ In fact, $\hat{u}^S$ is the optimal solution of* (3.1)*.*

Proof Invoke Theorem 7.5: the whole sequences δ^s, $\hat{h}^s$ and $\hat{g}^s$ converge to 0. Then proceed exactly as for Theorem 8.1, with the simplifying property that $\hat{u}^s = \bar{u}$ for all $s > S$. □

Note that this result makes no assumption about primal feasibility… and yet proves primal existence! This has an interesting consequence:

Corollary 8.3 *Suppose that* (1.1) *has no feasible point. Then the dual function θ does not reach its maximum.*

Proof Suppose for contradiction that (3.1) has an optimal solution $\bar{u}$. Initialize Algorithm 6.1 with $u^1 = \bar{u}$. There can be no descent step and Theorem 8.2 establishes the existence of an optimal primal solution. □

9 Numerical illustrations

To get a feeling of the numerical merits of our approach, we benchmark it on 16 test-problems against two implementations of its direct concurrent: the standard bundle method, which we briefly recall now.

If no attention is paid to its smoothness, Φ can be approximated via the linearizations $\bar{\Phi}_j^s(u_j) := \Phi_j(u_j) - (u_j - u_j^s)^\top y_j(u_j^s)$, instead of the quadratic functions $\tilde{\Phi}_j(u_j)$ of (3.4). Then Φ can be approximated just as Π by a polyhedral function (call it $\hat{\Phi}$) instead of the quadratic function $\tilde{\Phi}$ of (3.4); then a bundle method maximizes the resulting polyhedral approximation $\hat{\Phi} + \hat{\Pi}$ of θ, stabilized by a quadratic term $\frac{1}{2t}\|u - \hat{u}\|^2$; $t > 0$ is a parameter. Standard bundle methods maximize this approximation, and then manage the stability center $\hat{u}$ just as in Algorithm 6.1 (except that no backtracking is necessary).

An implementation in the spirit of the present paper uses the *individual* approximations $\Phi_j(u_j) \leqslant \hat{\Phi}_j(u_j) := \min_s \bar{\Phi}_j^s(u_j)$, thus replacing (3.5) by

$$\left.\begin{array}{l} \max\limits_{h,\phi,\pi}\Big\{\sum\limits_{j=1}^n \phi_j + \sum\limits_{k=1}^K \pi^k - \dfrac{1}{2t^S}\|h\|^2\Big\} \\ \text{s.t. } \phi_j \leqslant \bar{\Phi}_j^s(\hat{u}_j + h_j),\ j = 1,\dots,n, \\ \qquad \pi^k \leqslant (\hat{u}+h)^\top x^k(u^s),\ k = 1,\dots,K, \\ \qquad h_j \geqslant \frac{1}{c_j} - \hat{u}_j,\ j = 1,\dots,n. \end{array}\right\} s = 1,\dots,S,$$

We will refer to this implementation as `bfull`, as it fully splits the approximation of θ. Now remember Remark 5: ignoring the summation in Φ, we can also use the *compound* (less accurate) polyhedral approximation $\Phi(u) \leqslant \min_s \sum_j \bar{\Phi}_j^s(u_j)$. In compensation, the resulting quadratic program simplifies to

$$\left.\begin{array}{l} \max\limits_{h,\phi,\pi}\Big\{\phi + \sum\limits_{k=1}^K \pi^k - \dfrac{1}{2t^S}\|h\|^2\Big\} \\ \text{s.t. } \phi \leqslant \sum\limits_{j=1}^n \bar{\Phi}_j^s(\hat{u}_j + h_j), \\ \qquad \pi^k \leqslant (\hat{u}+h)^\top x^k(u^s),\ k = 1,\dots,K, \\ \qquad h_j \geqslant \frac{1}{c_j} - \hat{u}_j,\ j = 1,\dots,n. \end{array}\right\} s = 1,\dots,S,$$

We also compare our method to this implementation, referred to as `bhalf`: it uses only a half of the splitting possibilities in θ.

With Algorithm 6.1 (referred to as `Ncp`), this makes three solvers, which have been implemented in C on a bi-processor Intel Xeon (30.06GHz, 1.5GB RAM) under Linux operating system. Both standard bundle variants are home-made implementations of [12] (in particular for the t-management); Dijkstra's algorithm is used for the shortest path problems and the various QP are solved by Cplex 10.0. We use $\kappa = \kappa' = 0.1$ in Algorithm 6.1; the stopping criterion is $\underline{\delta} = 10^{-6}$ for `Ncp` and $\varepsilon = 10^{-6}$ for `bfull` and `bhalf`. We group the commodities by source nodes so that each computation of Π calls at most m times Dijkstra's algorithm.

The results are summarized in Table 9.1.

Pb	m	n	K	iterations			total CPU			relative final θ-accuracy		
				`Ncp`	`bfull`	`bhalf`	`Ncp`	`bfull`	`bhalf`	`Ncp`	`bfull`	`bhalf`
1	14	22	23	12 (88)	19	645	0.03	0.20	286.2	0.	0.	10^{-6}
2	19	68	30	5 (6)	11	16	0.02	0.14	0.08	0.	10^{-7}	0.
3	60	280	100	7 (95)	14	93	0.20	1.47	8.01	0.	0.	0.
4	61	148	122	7 (8)	24	167	1.07	8.84	61.09	0.	0.	2×10^{-8}
5	20	64	133	7 (8)	16	156	0.35	2.67	70.59	0.	0.	5×10^{-8}
6	122	332	162	9 (90)	21	309	0.61	5.23	495.4	0.	0.	10^{-8}
7	100	600	200	7 (96)	17	190	0.78	6.91	250.6	0.	10^{-8}	2×10^{-8}
8	30	72	335	7 (104)	24	1000*	0.13	3.56	5982.1	0.	10^{-7}	3×10^{-6}
9	21	68	420	7 (8)	42	151	5.62	88.66	856.5	0.	0.	6×10^{-7}
10	100	800	500	10 (304)	16	274	9.60	26.32	2061	0.	4×10^{-8}	8×10^{-7}
11	67	170	761	8 (133)	32	158	4.84	64.26	409.	0.	3×10^{-8}	5×10^{-8}
12	34	160	946	5 (6)	14	285	1.03	12.20	1650.2	0.	5×10^{-8}	5×10^{-8}
13	300	2000	1000	11 (319)	22	569	73.37	322.51	30564.	0.	7×10^{-9}	2×10^{-7}
14	48	198	1583	9 (80)	22	803	13.09	68.45	5h45	0.	10^{-7}	4×10^{-7}
15	81	188	2310	3 (4)	19	1000*	2.44	308.51	28h	0.	3×10^{-7}	9×10^{-4}
16	122	342	2881	9 (73)	30	1000*	311.85	764.5	42h	0.	2×10^{-7}	2×10^{-2}

Table 9.1 Comparison of Algorithm 6.1 (`Ncp`) with two alternative standard bundle implementations (`bfull` and `bhalf`).

– The first group of 4 columns describes the 16 test problems, ranked by number K of commodities (recall that the number of dual variables is n). Problems 3,7,10 and 13 are the random networks already used in [20]. Problems 1 and 4 are nso22 and nso148, well known in the convex optimization community: see [2, 8]. The remaining test problems are based on actual networks.
– The next group of columns gives the number of QP solved; column `Ncp` gives also the number of oracle calls (given by the number of backtracking steps; for standard bundle, one iteration solves one QP and calls the oracle once).
– Computing times are in seconds; they are mainly indicative and would probably change substantially with the use of a specialized QP solver such as [11, 13, 4].
– The last three columns are constructed as follows: for each test-problem, the best θ-value obtained by the three methods is called optimal (note that `Ncp` is always the winner); then we record the final gap obtained by the other two methods.

This table is rather eloquent. In terms of accuracy, all methods are comparable, except three failures of `bhalf`; but `Ncp` is drastically faster. First, it always requires less iterations. Also, the work per iteration is definitely smaller for `bhalf`, which solves a much cheaper quadratic program. Nevertheless, its computing time is overwhelmed by the two others', even forgetting the three instances where the stopping criterion could not be reached.

If comments on the behaviour of `Ncp` should be ventured, we could observe that the number of QP resolutions is remarkably and consistently small. This probably means that the polyhedral part Π of θ is easily approximated by the polyhedral model $\hat{\Pi}$ (only few null-steps are performed). However there are relatively many bactrackings, which occur in the early iterations, when Newton's model approximates Φ poorly; more backtrackings are needed when the number n of arcs is larger. As for CPU, it is mostly spent by the QP solver; computing Π is marginal in comparison, even for example in Problem 16, which computes 73x2881 shortest paths.

Table 9.1 allows some rudimentary comparison with other existing methods. In fact, the projection method [2] and ACCPM [8] were tested in [18] on Problems 3, 7, 10 and 13. Combining the results reported in [18, Table 4] with ours in Table 9.1 gives Table 9.2. Only iteration numbers are recorded: no reliable comparison could be established concerning computing times, the machines being so different. Recall that `Ncp` has two iteration numbers: one for the QP and one for the oracle. For the other two methods, these two numbers are equal.

Problem	3	7	10	13
m-n-K	60-280-100	100-600-200	100-800-500	300-2000-1000
`Ncp`	7(15)	7(96)	10(304)	11(319)
PM [2]	36	988	92	9949
ACCPM [8]	12	15	13	15

Table 9.2 Comparison of `Ncp` with PM and ACCPM on four problems.

10 Putting the method in perspective

We conclude with a discussion on some general aspects of this work.

(i) Field of applicability. Our method is of course not limited to the Kleinrock delay function, not even to the format (1.1). We can have for example primal problems of the type

$$\min f(y)+h(x)\,,\quad Ax-By=d\,,\quad x\in P\,. \tag{10.1}$$

Minimizing the Lagrangian

$$L(x,y,u)=f(y)-u^\top By+h(x)+u^\top Ax-d^\top u$$

results in a dual function of the type

$$\theta(u)=-f^*\big(B^\top u\big)+\Pi(u)-d^\top u$$

where $\Pi(u):=\min_{x\in P}h(x)+u^\top Ax$. No particular assumption is needed on P and h, except that $\Pi(u)$ must be computable for given u. Besides, polyhedral P and h are preferred (see (iv) below).

Our approach is relevant whenever f^* is twice differentiable. According to [9, Corollary X.4.2.10], this essentially requires an f which is twice differentiable, with a positive definite Hessian. Indeed, the Lagrangian has then a unique minimum $y(u)$ with respect to y, given by the system of equations

$$\nabla f(y)-B^\top u=0\,.$$

From the implicit function theorem, $y(u)$ is differentiable and our calculations in §2 can be reproduced.

More generally, we have here a method to maximize a sum $\theta(u)=\Phi(u)+\Pi(u)$ of two concave functions, where Φ is twice differentiable. Three informations are needed for each u: a supergradient of Π, as well as the differential elements $\nabla\Phi(u)$ and $\nabla^2\Phi(u)$. Note that, if Φ is known only through its gradient, the method can still work via the approximation of $\nabla^2\Phi$ by a quasi-Newton strategy.

(ii) Other variants. Our algorithm is based on the so-called proximal bundle method. Since several other forms exist (let us cite level bundle [15], dual bundle [17], bundle-trust [21], see also [5]), a relevant question is whether they could be considered as well. The above comments suggest that they are actually less adapted to the present problem. It does make a lot of sense to approximate Φ [resp. Π] by its second-order development $\tilde\Phi$ [resp. polyhedral $\hat\Pi$], and add these two approximations to obtain $\tilde\Phi+\hat\Pi\simeq\Phi+\Pi$.

Similarly, line-search is not the only possible strategy for backtracking. An alternative is for example the "curved search" of [16], in which u^+ solves

$$\max\tilde\Phi(u)+\hat\Pi(u)-\frac{1}{2t}\|u-\hat u\|^2\,,\quad u\geqslant 1/c\,. \tag{10.2}$$

Interpreting $1/2t$ as a Lagrange multiplier, this is essentially equivalent to the trust-region technique, familiar in nonlinear programming:

$$\max \tilde{\Phi}(u) + \hat{\Pi}(u)\,, \quad \|u-\hat{u}\| \leqslant \Delta\,, \quad u \geqslant 1/c$$

(see [19] for a review). This approach is motivated by badly conditioned Hessians $\nabla^2\Phi$: it annihilates the second-order term in $\tilde{\Phi}$ when $t \searrow 0$ in (10.2) (or equivalently $\Delta \searrow 0$). On the other hand, it requires one more resolution of the quadratic master after each backtrack.

(iii) Convergence theory. Our results of §7 are limited to a rather particular situation:

– Compactness automatically holds (Lemma 7.2); the stability centers are bounded, and this makes life much easier to establish convergence.
– The quadratic term in the master problem (3.5) behaves itself: the ℓ and L of Lemma 7.1 are appropriately bounded, which allows an easy proof of Theorem 7.3. We do not know if this proof would be preserved with a more nasty Φ.
– Algorithm 6.1 keeps all the answers from the oracle (2.2) to make up $\hat{\Pi}$. Yet, as alluded to in Remark 6.2, it is standard practice to clean the bundle when necessary, to spare memory and ease the QP solver; this would kill the proof of Theorem 7.5. This proof can probably be generalized but still, the work has to be done.

The smooth aspect of the method also presents some interest, in particular the interaction between bundling and backtracking. In summary, an improved and more thorough convergence theory deserves study.

(iv) Numerical efficiency. A question then naturally arises: does our variant really deserve attention? Does it really improve standard implementations of the existing optimization methods? Table 9.1 suggests a definite yes but what is the generality of our experiments?

We believe that the whole issue is whether Π is well approximated by $\hat{\Pi}$, i.e. whether Π looks like a polyhedral function (with respect to (i) above, so is the case if P and h of (10.1) are polyhedral, with moderately many corners). Then bundling will not be crucial, a few pieces in $\hat{\Pi}$ will suffice. Newton will take care of approximating Φ (very efficiently, as is well-known). As a result, convergence will be fast (namely comparable to Newton). In terms of (1.1), the property $\Pi \simeq \hat{\Pi}$ means that the optimal flows look like paths: they split only at few nodes, and in only few branches.

In fact, stabilization of a cutting-plane algorithm can be viewed as follows: we do know that $\hat{\Pi}$ overestimates the actual Π, sometimes drastically; the bundle technique subtracts a (Euclidean) term from $\hat{\Pi}$, hopefully improving the approximation. Here we subtract nothing. If $\hat{\Pi}$ were really a bad approximation of Π, it would perhaps be a better idea to introduce a stabilizing parameter $t > 0$ and solve the quadratic master (10.2).

Note that the role of t is here different from that in (ii): the additionnal Euclidean term is here supposed to improve the approximation $\hat{\Pi} \simeq \Pi$ and all possible values of $t > 0$ are a priori suitable. In (ii), this Euclidean term was supposed to improve the approximation $\tilde{\Phi} \simeq \Phi$ and successive trials with decreasing values of

$t \in]0,1]$ were natural. As usual with the bundle approach, an appropriate management of t would be delicate here; especially if it interferes with its backtracking role of (ii).

Our experiments, confirmed by those of [1], suggest that this stabilization is useless, at least with the instances that we have tested: they are probably close to multicommodity shortest *path* problems, i.e. their optimal flows ar close to true paths. This might be due to the fact that capacities are "comfortable", in terms of the traffic they have to accommodate. For stiffer instances, a refinement as suggested in (10.2) might improve convergence if necessary. This technique might also become useful if $\hat{\Pi}$ is replaced by the compound approximation mentioned in Remark 6.2.

(v) Implementation questions. Needless to say, our numerical experiments in §9 are only preliminary; their ambition is limited to checking the viability of the method and the role of the Newton term, as compared with a standard proximal term. More intensive experiments should in particular involve serious comparisons with competitors such as projection [2] or ACCPM [8,1], on a set of common test-problems. To be conclusive, however, these comparisons should involve large instances. The quadratic master problem (3.5) then becomes large-scale and hard to solve by our general-purpose CPLEX software: a more refined QP solver is then needed. Beyond [11,13,4] already mentioned, such a solver could follow the ideas of [2].

Let us elaborate on this last point. Neglecting for simplicity the constraint $u \geqslant 1/c$ (remember footnote 1, page 7), (3.5) is easy to dualize. As in (2.4), (3.4), call $\hat{y} := (\nabla f)^{-1}(\hat{u})$ (a primal stabilizer) the unique point minimizing $f(y) - \hat{u}^\top y$ and form the quadratic approximation of f around $\hat{y}$; say

$$\tilde{f}(y) := \nabla f(\hat{y})^\top (y-\hat{y}) + \frac{1}{2}(y-\hat{y})^\top \nabla^2 f(\hat{y})(y-\hat{y})\,.$$

Then the dual of the simplified (3.5) consists in minimizing a quadratic function over a product of simplices, namely

$$\min \tilde{f}\Big(\sum_{k=1}^{K} x^k\Big)\,, \quad x^k \in \operatorname{conv}\big\{x^k(u^1),\ldots,x^k(u^S)\big\}\,,\ k=1,\ldots,K\,.$$

This problem is very similar to the one considered in [2]; a similar resolution method could then be considered. Alternatively, we can say that our Newton-cutting plane method is very similar to that of [2], with a special rule to update the quadratic approximation (i.e. the primal center $\hat{y}$); this rule takes care in particular of the capacity constraints $y < c$.

The constraint $u \geqslant 1/c$ that we negleced in the above development is really troublesome: (3.5) would be a lot easier to solve if u were free in $\mathbb{R}^n$. This is so true that [1] developed a whole machinery (using "compound congestion functions") to somehow anticipate the really active part of u.

Let us finally say a word about the choice of parameters. Numerical methods have often some parameters hard to tune, which may perceivably (or critically) influence convergence; this may have a bad influence on the robustness of the method. Here there are only two such parameters: κ and κ' of §4 (barring the

above-mentioned t). They play the role of the Armijo-Goldstein parameters in classical (smooth) optimization, which are known to have marginal influence on convergence. The same insensitivity should be observed here; indeed, we have not even bothered to try other values than $\kappa = \kappa' = 0.1$, which could be considered as "default" values.

Acknowledgment. We are indebted to the referees, whose thorough readings and insightful comments were decisive to improve an earlier version of this paper.

References

1. F. Babonneau and J.P. Vial. Proximal-accpm with a nonlinear constraint and active set strategy to solve nonlinear multicommodity flow problems. *Mathematical Programming, forthcoming*, 2006.
2. D. Bertsekas and E.M. Gafni. Two-metric projection methods for constrained optimization. *SIAM J. Control and Optimization*, 22:936–964, 1983.
3. E. Cheney and A. Goldstein. Newton's method for convex programming and Tchebycheff approximations. *Numerische Mathematik*, 1:253–268, 1959.
4. A. Frangioni. Solving semidefinite quadratic problems within nonsmooth optimization algorithms. *Computational Operational Research*, 23(11):1099–1118, 1996.
5. A. Frangioni. Generalized bundle methods. *SIAM J. on Optimization*, 13(1):117–156, 2003.
6. M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistic Quarterly*, 3:95–110, 1956.
7. M. Gerla, L. Fratta, and L. Kleinrock. The flow deviation method: an approach to store-and-forward communication network design. *Networks*, 3:97–133, 1984.
8. J.-L. Goffin, J. Gondzio, R. Sarkissian, and J.-Ph Vial. Solving nonlinear multicommodity flow problems by the analytic center cutting plane method. *Mathematical Programming*, 76:131–154, 1996.
9. J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms*. Springer Verlag, Heidelberg, 1993. Two volumes.
10. J. E. Kelley. The cutting plane method for solving convex programs. *J. of the SIAM*, 8:703–712, 1960.
11. K. C. Kiwiel. A dual method for certain positive semidefinite quadratic programming problems. *SIAM J. on Scientific and Statistical Computing*, 10(1):175–186, 1989.
12. K. C. Kiwiel. Proximity control in bundle methods for convex nondifferentiable minimization. *Mathematical Programming*, 46(1):105–122, 1990.
13. K.C. Kiwiel. A Cholesky dual method for proximal piecewise linear programming. *Numerische Mathematik*, 68:325–340, 1994.
14. C. Lemaréchal. Lagrangian relaxation. In M. Jünger and D. Naddef, editors, *Computational Combinatorial Optimization*, pages 112–156. Springer Verlag, Heidelberg, 2001.
15. C. Lemaréchal, A.S. Nemirovskii, and Yu.E. Nesterov. New variants of bundle methods. *Mathematical Programming*, 69:111–148, 1995.
16. C. Lemaréchal and C. Sagastizábal. Variable metric bundle methods: from conceptual to implementable forms. *Mathematical Programming*, 76(3):393–410, 1997.
17. C. Lemaréchal, J.-J. Strodiot, and A. Bihain. On a bundle method for nonsmooth optimization. In O.L. Mangasarian, R.R. Meyer, and S.M. Robinson, editors, *Nonlinear Programming 4*, pages 245–282. Academic Press, 1981.
18. P. Mahey, A. Ouorou, L. LeBlanc, and J. Chifflet. A new proximal decomposition algorithm for routing in telecommunication networks. *Networks*, 31:227–238, 1998.
19. J.J. Moré. Recent developments in algorithms and software for trust region m ethods. In A. Bachem, M. Grötschel, and B. Korte, editors, *Mathematical Programming, the State of the Art*, pages 258–287. Springer-Verlag, Berlin, 1983.
20. A. Ouorou, Ph. Mahey, and J-Ph. Vial. A survey of algorithms for convex multicommodity flow problems. *Management Science*, 47(1):126–147, 2000.
21. H. Schramm and J. Zowe. A version of the bundle idea for minimizing a nonsmooth function: conceptual idea, convergence analysis, numerical results. *SIAM J. on Optimization*, 2(1):121–152, 1992.
22. Ph. Wolfe. Convergence theory in nonlinear programming. In J. Abadie, editor, *Integer and Nonlinear Programming*, pages 1–36. North Holland, 1970.