Université de Montréal

Accélération de méthodes de résolution classiques par l'utilisation de stratégies de séparation locale comme outil d'hybridation

par

Walter Rei

Département d'informatique et de recherche opérationnelle

Faculté des arts et des sciences

Thèse présentée à la Faculté des études supérieures

en vue de l'obtention du grade de

Doctorat (Ph.D.)

en informatique option recherche opérationnelle

Novembre 2006

Université de Montréal

Faculté des études supérieures

Cette thèse intitulée:

Accélération de méthodes de résolution classiques par l'utilisation de stratégies de séparation locale comme outil d'hybridation

présenté par:

Walter Rei

a été évalué par un jury composé des personnes suivantes:

Jean-Yves Potvin

(président-rapporteur)

Michel Gendreau

(directeur de recherche)

Patrick Soriano

(co-directeur)

Jacques Ferland

(membre du jury)

François Louveaux

(examinateur externe)

Thèse acceptée le:

17 avril 2007

Sommaire

Cette thèse est consacrée au développement de méthodes de résolution efficaces adaptées à des problèmes de programmation stochastique en nombres entiers. La programmation stochastique permet de traiter le cas où de l'incertitude est présente dans les paramètres d'un problème d'optimisation. D'un point de vue pratique, les modèles de programmation stochastique en nombres entiers sont intéressants car ils permettent de mieux traduire des situations réelles d'optimisation. Par contre, puisque ces problèmes sont à la fois combinatoires et stochastiques, ils posent des défis importants.

L'approche méthodologique utilisée dans le cadre de ce travail est l'hybridation de méthodes. Plus précisément, la séparation locale est employée au sein d'approches classiques d'optimisation afin de rendre celles-ci plus performantes pour le cas considéré. L'une des principales approches utilisée en programmation stochastique est la décomposition de Benders. Par conséquent, la séparation locale sera premièrement utilisée de façon à accélérer la décomposition de Benders classique. En appliquant des phases de recherche de type séparation locale à partir des solutions aux problèmes maîtres relaxés, il est possible d'améliorer simultanément les bornes inférieure et supérieure générées par l'algorithme de Benders. Les résultats numériques obtenus sur une famille générale de problèmes de conception de réseaux offriront un exemple des bénéfices de cette méthode. La contribution principale de cette approche est d'ordre méthodologique, car les principes évoqués sont applicables à tous les problèmes pouvant être résolus par la décomposition de Benders.

Les résultats obtenus dans le contexte général de la décomposition de Benders nous

ont poussé à appliquer des idées similaires au cas de l'algorithme « L-shaped » binaire. L'algorithme « L-shaped » binaire est utilisé pour résoudre des problèmes de programmation stochastiques en nombres entiers pour lesquels un sous-ensemble des variables sont binaires. Dans ce cas, nous démontrons qu'une phase de recherche de type séparation locale permet de générer un système d'inégalités valides applicable aux problèmes relaxés résolus par l'algorithme. Ces inégalités servent à fournir de meilleures bornes inférieures pour la composante stochastique du problème. Ces résultats sont spécialisés au cas des problèmes de tournées d'un véhicule avec demandes stochastiques. Deux stratégies d'utilisation pour ces inégalités sont présentées. De plus, une procédure de génération de plans coupants, adaptée au problème considéré, est développée en vue d'une utilisation au sein de l'algorithme « L-shaped » binaire. L'expérimentation numérique, qui est effectuée sur un ensemble de 280 instances, montre qu'il y a un avantage significatif à utiliser ces nouvelles inégalités pour le problème traité. Dans ce cas, les contributions sont à la fois méthodologique et pratique. D'un point de vue méthodologique, ces inégalités sont générales et peuvent être adaptées à tout problème stochastique qui est résolu par l'algorithme « L-shaped » binaire. Au niveau pratique, la spécialisation de l'approche au cas des problèmes de tournées d'un véhicule avec demandes stochastiques permet d'améliorer de façon marquée l'approche de résolution exacte la plus efficace pour ces problèmes. Il est cependant important de préciser que les résultats obtenus nous ont également permis d'identifier des instances particulièrement difficiles à résoudre de façon exacte.

Ces dernières observations ont servi de motivation au développement d'une approche de résolution heuristique efficace pour les problèmes de tournées d'un véhicule avec demandes stochastiques. Pour ce faire, nous nous sommes intéressé à la simulation de Monte Carlo. La simulation de Monte Carlo est utilisée en programmation stochastique afin de contrôler la complexité associée à la composante aléatoire des problèmes. Dans le cas présent, il faut également tenir compte de la complexité combinatoire de la problématique considérée. L'heuristique proposée est une méthode hybride alliant des principes de recherche de type séparation locale avec les techniques d'approximation de la simulation Monte Carlo. Dans cette approche, la séparation locale sert à explorer

efficacement le domaine réalisable des problèmes tout en contrôlant la complexité combinatoire de ceux-ci. En utilisant les principes de la séparation locale, nous développons des précédures de diversification et d'intensification qui sont insérées au sein de l'heuristique qui fonctionne selon un schème de recherche de type multi-descentes. Les résultats numériques obtenus sur un sous-ensemble d'instances considérées comme étant difficiles à résoudre de façon exacte, montrent que l'heuristique est à la fois robuste, car elle converge vers des solutions optimales ou quasi-optimales, et efficace, car elle permet de trouver de bonnes solutions rapidement. La contribution principale de la méthode proposée est d'ordre pratique. L'heuristique a été développée en fonction de caractéristiques propres aux problèmes de tournées d'un véhicule avec demandes stochastiques. Par contre, les principes méthodologiques de recherche qui sont utilisés sont facilement généralisables à d'autres problématiques.

Mots-clés. Décomposition de Benders, séparation locale, conception de réseaux, programmation stochastique en nombres entiers, inégalités valides, tournées de véhicules avec demandes stochastiques, simulation de Monte Carlo.

Summary

The main objective pursued in this thesis was to enhance certain algorithms that are used to solve integer stochastic programming problems. Integer stochastic programming can be applied when uncertainty is present in the parameters of an optimization problem. From a modeling point of vue, integer stochastic programming is interesting, since one rarely has perfect information concerning the problem to be solved. However, both the combinatorial and stochastic complexities of these models make them notoriously hard to solve.

The research that we have done focuses on the use of local branching strategies in order to accelerate certain classical algorithms. One of the main solution strategies that is used in integer stochastic programming is Benders decomposition. Therefore, we first propose a way to accelerate the classical Benders decomposition algorithm using local branching. By applying local branching search phases throughout the Benders solution process, one can simultaneously improve both the upper and lower bounds generated by the algorithm. An important characteristic of the proposed method is that it can be applied to all problems solved using the Benders decomposition algorithm. Results obtained on a set of network design problems provide a numerical example of the benefits of this approach.

The favorable results obtained on the classical Benders decomposition method encouraged us to extend similar ideas to the case of the integer 0-1 L-shaped algorithm. The integer 0-1 L-shaped algorithm is an exact procedure that solves integer stochastic programming problems for which a subset of variables are binary. In this case, we

demonstrate that a local branching search phase can be used to produce a system of valid inequalities that provide better lower bounds for the stochastic component of the relaxed problems solved by the algorithm. The single vehicle routing problem with stochastic demands is used to test these new inequalities. Two different strategies are presented for generating these cuts. Furthermore, a separation procedure, adapted to this particular problem, is developped to search for violated cuts. Results obtained on a set of 280 instances show that there is a significant advantage in using these new inequalities for the case of the single vehicle routing problem with stochastic demands. However, these results also show that certain instances remain hard to solve optimally in a reasonable amount of time.

These last observations served as motivation for the development of a heuristic procedure adapted to the single vehicle routing problem with stochastic demands. The proposed heuristic is a hybrid method that uses both local branching and Monte Carlo simulation in a multi-descent search strategy. Monte Carlo simulation is used in stochastic programming to control the stochastic complexity associated with the problems to be solved. In the case of an integer stochastic programming problem, one must also consider the combinatorial complexity. Therefore, local branching is used to search effectively the feasible region of the problem while controlling the combinatorial complexity. Using principles taken from local branching, we present both a diversification and an intensification search strategy for the single vehicle routing problem with stochastic demands. Results obtained on a subset of particularly hard instances show that the heuristic is both robust, since it finds optimal or near optimal solutions, and effective, because it obtains very good solutions quickly. The solution method presented is adapted to the particular problem that is considered. However, it should be specified that the principles used are general and may be applied to a wider range of applications.

Keywords. Benders decomposition, local branching, network design, integer stochastic programming, valid inequalities, vehicle routing with stochastic demands, Monte Carlo simulation.

Table des matières

Liste des figures

Liste des tableaux

Remerciements

Je remercie très sincèrement mes directeurs de recherche, M. Michel Gendreau, directeur, et M. Patrick Soriano, co-directeur, pour leur soutien et leurs conseils tout au long de ce travail. J'ai eu la chance de pouvoir compter sur leurs connaissances et leurs nombreuses années d'expérience. J'aimerais également leur témoigner ma reconnaissance pour les nombreuses discussions scientifiques que nous avons eues de même que pour la complète liberté qu'ils m'ont donnée en ce qui a trait à ce travail.

J'aimerais aussi remercier M. Jean-François Cordeau, avec qui j'ai eu le plaisir de travailler sur un article scientifique. J'ai bénéficié de sa profonde compréhension du domaine étudié de même que de sa grande disponibilité. Sa collaboration à cet article m'a beaucoup aidé à un moment important dans la réalisation de cette thèse.

Je tiens à exprimer ma reconnaissance au personnel du Centre de recherche sur les transports, où j'ai eu le privilège d'effectuer tous mes travaux de recherche. Particulièrement, j'aimerais souligner l'aide précieuse que m'ont donnée MM. Serge Bisaillon et François Guertin en ce qui a trait au développement informatique. J'aimerais également remercier MM. Éric Springuel, Nikolaj Van Omme, Benoît Crevier et Jean-François Bérubé, ainsi que Mme. Nadia Larichi pour leur écoute, leurs conseils et pour avoir partagé les aléas de la recherche.

J'aimerais remercier mon épouse, Anne-Héloise Bédard, pour son support de tous les instants et pour sa grande compréhension tout au long de ce travail. Sans elle, ces dernières années auraient été bien plus difficiles. J'ai profité de ses conseils précieux et de ses suggestions pertinentes tout au long de mon cheminement aux études supérieures.

Finalement, j'aimerais remercier ma famille, mes amis et principalement, M. Michel Bédard et Mme. Sylvie Glen pour toute leur aide concernant la rédaction de ce travail.

Chapitre 1

Introduction

Dans le domaine de l'optimisation, il arrive souvent que des problèmes comportent des aspects aléatoires. Nous entendons par là que la résolution de certaines problématiques s'opère dans un contexte où les paramètres ne sont pas tous connus à l'avance. La programmation mathématique stochastique a été développée dans le but de traiter certaines de ces situations. Dans le cadre de ce travail, nous considérons le cas du problème de programmation stochastique linéaire entier avec recours fixe. Ce problème peut être défini de la façon suivante (nous utilisons ici la notation proposée par Birge et Louveaux [7]) :

$$Min \quad c^{\top}x + \mathcal{Q}(x) \tag{1.1}$$

$$s.l.c. \quad Ax = b \tag{1.2}$$

$$x \in X \tag{1.3}$$

$$\text{où } \mathcal{Q}(x) = \mathbf{E}_{\xi}[Q(x, \xi(\omega))] \tag{1.4}$$

$$\text{et } Q(x, \xi(\omega)) = Min_y\{q(\omega)^{\top}y \mid Wy = h(\omega) - T(\omega)x,\ y \in Y\}. \tag{1.5}$$

Ce type de modèle comporte deux niveaux de décisions. Ces niveaux de décisions dépendent du moment où les paramètres du problème deviennent disponibles. D'un côté, il y a les décisions dites de première étape, qui sont représentées par le vecteur x. Ces décisions doivent être prises a priori en considérant un ensemble de contraintes

((1.2) et (1.3)) et un objectif à optimiser ((1.1)). La matrice A (de dimension $m_1 \times n_1$), de même que le vecteur c (de dimension $n_1 \times 1$), sont des données connues à l'avance. Quant à lui, l'ensemble X contient toutes les autres contraintes impliquant les variables de première étape. Cela inclut les contraintes de non-négativité et d'intégralité selon les cas traités.

Une fois le vecteur x fixé, un événement aléatoire se réalise. Nous définissons Ω comme étant l'ensemble de toutes les réalisations possibles de cet événement. Le vecteur ξ est un vecteur aléatoire défini par rapport à Ω. Nous dénotons par $\xi(\omega)$, $\omega \in \Omega$, les valeurs du vecteur ξ pour une certaine réalisation ω de l'événement aléatoire. Les vecteurs $q(\omega)$ et $h(\omega)$ (de dimensions $n_2 \times 1$ et $m_2 \times 1$), de même que la matrice $T(\omega)$ (de dimension $m_2 \times n_1$), que l'on nomme matrice technologique, constituent les composantes de $\xi(\omega)$. Lorsque ces paramètres deviennent disponibles, une série de décisions, dites de deuxième étape, sont prises. Ces décisions, représentées par le vecteur y, définissent le recours possible étant données les décisions prises précédemment x et les valeurs observées du vecteur aléatoire $\xi(\omega)$. La fonction $Q : x \times \xi(\omega) \rightarrow \mathbb{R}_+$, qui est définie par le problème d'optimisation en (1.5), mesure la valeur optimale du recours pour x et $\xi(\omega)$ fixé. Par convention, lorsque le problème (1.5) est irréalisable, on pose $Q(x, \xi(\omega)) = +\infty$. Il est question de problèmes à recours fixe, puisque la matrice W (de dimension $m_2 \times n_2$), que l'on nomme matrice de recours, est constante $\forall\ \omega \in \Omega$. Encore une fois, l'ensemble Y contient toutes les autres contraintes impliquant les variables y. Selon les cas considérés, des contraintes de non-négativité et d'intégralité y sont présentes.

L'utilisation du problème (1.1)-(1.5) suppose qu'il est impossible de prévoir avec certitude laquelle des réalisations sera observée. L'objectif (1.1) comprend donc la fonction $\mathcal{Q} : x \rightarrow \mathbb{R}_+$, que l'on nomme fonction de recours. Cette fonction mesure la valeur espérée du recours associé à x pour l'ensemble des réalisations possibles de l'événement aléatoire. En résolvant (1.1)-(1.5), on obtient une solution $x^\star$, qui est optimale, en moyenne, lorsqu'on considère le vecteur aléatoire ξ. Les problèmes de programmation stochastiques sont intéressants, car ils permettent d'incorporer une certaine variabilité au sein des paramètres d'un modèle. Par contre, il est important de réaliser que la complexité associée a ce type de problème est très grande. Il est donc justifié de questionner

l'utilité de la solution optimale $x^\star$ au modèle (1.1)-(1.5).

Madansky [44] compare la solution $x^\star$ à celle que l'on obtient en résolvant le problème d'approximation déterministe associé à (1.1)-(1.5). Le problème d'approximation déterministe est obtenu en remplaçant le vecteur aléatoire ξ par son espérance $\overline{\xi} = \mathbf{E}[\xi]$. Ce faisant, le problème devient : $Min\ \{c^\top x + \overline{q}^\top y \quad | \quad Ax = b,\ Wy = \overline{h} - \overline{T}x,\ x \in X,\ y \in Y\}$, où $\overline{q} = \mathbf{E}[q(\omega)]$, $\overline{h} = \mathbf{E}[h(\omega)]$ et $\overline{T} = \mathbf{E}[T(\omega)]$. Dénotons par $\overline{x}$, la solution optimale à ce nouveau problème. Madansky [44] démontre l'inégalité suivante : $c^\top x^\star + \mathcal{Q}(x^\star) \leq c^\top \overline{x} + \mathcal{Q}(\overline{x})$. La valeur de la solution stochastique (VSS), qui est definie comme étant VSS $= c^\top \overline{x} + \mathcal{Q}(\overline{x}) - c^\top x^\star + \mathcal{Q}(x^\star)$, représente l'écart existant entre la valeur de la solution au modèle stochastique et la valeur de la solution à l'approximation déterministe. L'utilité de (1.1)-(1.5) est directement relié à la VSS. Lorsque la VSS est faible, la version déterministe est une bonne approximation de (1.1)-(1.5). Dans le cas limite où VSS $= 0$, la solution $\overline{x}$ est optimale pour le problème stochastique ($x^\star = \overline{x}$). Par contre, lorsque la VSS est élevée, l'approximation déterministe est mauvaise et le modèle (1.1)-(1.5) devient beaucoup plus intéressant. L'importance de l'écart défini par la VSS dépend du contexte d'optimisation considéré. Il est également important de réaliser que l'obtention de cette valeur suppose, à la base, la résolution du problème stochastique. Bien qu'elle puisse être difficile à estimer, la VSS doit être néanmoins considérée lorsqu'il est question de résoudre un problème particulier de programmation stochastique. Dans le cadre de cette thèse, la principale problématique étudiée est celle des tournées de véhicules avec demandes stochastiques. L'intérêt pour ces problèmes découle du fait qu'ils combinent à la fois l'optimisation combinatoire et la programmation stochastique, ce qui pose des défis de résolution très importants. De plus, tel que montré par Louveaux [42], l'utilisation des versions stochastiques est tout à fait justifiée dans ces contextes.

Les contributions des trois articles présentés dans la cadre de cette thèse se divisent en deux catégories, selon qu'elles sont méthodologiques ou pratiques. Nous nous sommes premièrement intéressé à la décomposition de Benders [4], qui est l'une des principales approches de résolution utilisée en programmation stochastique. L'approche générale de Benders comporte certaines faiblesses qui peuvent ralentir le pro-

cessus de résolution. Par conséquent, le premier article de cette thèse propose une technique générale d'amélioration de l'algorithme classique de Benders. Cette technique d'amélioration est basée sur l'utilisation successive de phases de recherche de type séparation locale [20]. Pour ce premier article, la principale contribution est d'ordre méthodologique, car ce qui est proposé est une approche hybride générale de résolution de type Benders pour les problèmes d'optimisation mixte. L'expérimentation numérique, effectuée sur une série de problèmes de conception de réseau, n'est utilisée que dans le but de valider l'approche de résolution générale proposée.

Dans le cadre du deuxième article, nous avons poursuivi les travaux de recherche entamés, en spécialisant l'approche hybride au cas de la programmation stochastique. Sur le plan méthodologique, les contributions se sont traduites par le développement de nouvelles inégalités valides générales qui s'appliquent à l'algorithme de séparation et plans coupants « L-shaped » pour le cas des problèmes de type (1.1)-(1.5) comportant des variables de décision binaires (algorithme présenté par Laporte et Louveaux [39]). Nous avons appliqué ces nouvelles inégalités au cas particulier des problèmes de tournées d'un véhicule avec demandes stochastiques. Cette application pratique constitue également une importante contribution de l'article puisqu'elle a permis d'améliorer significativement la meilleure approche de résolution exacte pour ce type de problème. L'expérimentation numérique effectuée pour ce deuxième article nous a permis de distinguer des instances particulièrement difficiles à résoudre exactement. Ces observations ont motivé les travaux de recherche réalisés pour l'article final présenté dans cette thèse.

Pour ce dernier article, nous proposons un algorithme heuristique spécialisé au cas des problèmes de tournées d'un véhicule avec demandes stochastiques. Cette heuristique est une méthode hybride fondée sur la combinaison de recherche par séparation locale et simulation Monte Carlo, et ce, au sein d'une approche itérative de type multi-descentes. La principale contribution de cet article est d'ordre pratique, car ce qui est proposé est une heuristique efficace et robuste pour la résolution des instances difficiles du problème considéré. Par contre, il est important de noter que, d'un point de vue méthodologique, l'approche développée repose sur des concepts généraux, qui peuvent être facilement adaptés à d'autres problèmes stochastiques.

Ce travail est divisé en six chapitres. Le prochain chapitre comprend une brève revue des travaux antérieurs. Il est important de préciser que ce chapitre sera consacré aux principales approches méthodologiques qui ont été développées afin de résoudre des problèmes de programmation stochastique en nombres entiers. L'objectif principal de cette thèse a été de développer des approches de résolution efficaces pour résoudre des problèmes de type (1.1)-(1.5). En présentant les principales approches méthodologiques développées dans le domaine de la programmation stochastique en nombres entiers, il sera possible de situer beaucoup plus facilement le travail effectué dans le cadre des articles scientifiques proposés. Par la suite, un chapitre est consacré à chacun des trois articles présentés. Finalement, nous concluons ce travail au chapitre six.

Chapitre 2

Revue des travaux antérieurs

Ce chapitre est consacré aux principales approches de résolution, qui ont été développées afin de résoudre les problèmes de programmation stochastique linéaire entiers. Klein-Haneveld et van der Vlerk [36], de même que Schultz [54] font une synthèse de ces méthodes. Dans cette section, nous concentrerons la présentation sur les méthodologies de résolution pour les cas généraux où des contraintes d'intégralité peuvent être présentes en première et/ou en deuxième étape du problème. Dans un premier temps, nous présenterons les algorithmes de coupes. Ce groupe est principalement composé de techniques résultant de l'application de l'approche proposée par Benders [4]. Par la suite, il sera question des algorithmes de type séparation et évaluation progressive (ou « branch and bound »). Finalement, nous conclurons ce chapitre en présentant les approches basées sur la théorie des ensembles test.

2.1 Algorithmes de coupes

En 1970, Geoffrion [26, 27] propose une classification des principales techniques de résolution des problèmes de programmation mathématique de grande taille. Selon cette classification, l'approche de Benders [4] est décrite comme étant un algorithme de type projection, linéarisation externe et relaxation. Cette approche a été développée dans

le but de traiter spécifiquement le cas des problèmes linéaires mixtes, c'est-à-dire les problèmes où certaines des variables sont entières et d'autres continues. L'idée proposée par Benders consiste à effectuer premièrement une projection du problème dans l'espace défini par les variables entières. Le terme projeté subit par la suite une linéarisation externe. Finalement, le problème résultant est résolu par une approche de relaxation.

En 1969, Van Slyke et Wets [66] reprennent l'idée de Benders et l'appliquent au problème de programmation stochastique linéaire avec recours fixe. Les auteurs nomment leur procédure « L-shaped », étant donné la structure en blocs de la matrice des contraintes du problème à résoudre. Nous commencerons cette section en présentant la décomposition de Benders classique [4] (cas des problèmes linéaires mixtes), telle qu'appliquée au cas des problèmes de programmation stochastique. Dans ce cas, les variables de première étape sont entières ($X = \{x \in \mathbb{Z}^{n_1} \mid x \geq 0\}$) et les variables de deuxième étape sont continues ($Y = \{y \in \mathbb{R}^{n_2} \mid y \geq 0\}$). Nous verrons par la suite comment ces idées peuvent être étendues au cas purement entier. Le problème à résoudre est défini de la façon suivante :

$$Min \quad c^\top x + Q(x) \tag{2.1}$$

$$s.l.c. \quad Ax = b \tag{2.2}$$

$$x \in X \tag{2.3}$$

$$\text{où } Q(x) = \mathbf{E}_\xi[Q(x, \xi(\omega))] \tag{2.4}$$

$$\text{et } Q(x, \xi(\omega)) = Min_y\{q(\omega)^\top y \mid Wy = h(\omega) - T(\omega)x,\ y \geq 0\}. \tag{2.5}$$

Afin d'appliquer l'approche, nous supposons que le vecteur aléatoire ξ a une distribution discrète et que l'ensemble Ω est fini. Nous dénotons par ξ_k, $k = 1, \ldots, K$ les valeurs possibles de ξ. Chaque ξ_k est composée des vecteurs q_k et h_k, de même que de la matrice T_k. Définissons un vecteur y_k pour représenter les décisions de deuxième étape associées à chacune des réalisations possibles de l'événement aléatoire. En fixant p_k tel que $Pr(\xi = \xi_k) = p_k$, il est possible de réécrire le modèle (2.1)-(2.5) de la façon

suivante :

$$Min \quad c^\top x + \sum_{k=1}^{K} p_k q_k^\top y_k \tag{2.6}$$

$$s.l.c. \quad Ax = b \tag{2.7}$$

$$T_k x + W y_k = h_k, \ k = 1, \ldots, K \tag{2.8}$$

$$x \in X, \ y_k \geq 0, \qquad k = 1, \ldots, K. \tag{2.9}$$

Pour ce qui suit, nous supposons que (2.6)-(2.9) est à la fois réalisable et fini.

Dans un premier temps, projetons le problème (2.6)-(2.9) sur l'espace défini par les variables de première étape x. Cela donne le problème suivant :

$$Min \quad c^\top x + \sum_{k=1}^{K} p_k \ \inf\{q_k^\top y_k \mid T_k x + W y_k = h_k, \ y_k \geq 0\} \tag{2.10}$$

$$s.l.c. \quad Ax = b \tag{2.11}$$

$$x \in X. \tag{2.12}$$

Les fonctions inf présentes dans (2.10) sont définies par les modèles suivants :

$$\begin{array}{llcll} (P_k) & \text{Min} & q_k^\top y_k & & (D_k) \quad \text{Max} \quad (h_k - T_k x)^\top \lambda_k \\ & s.l.c. & W y_k = h_k - T_k x & \Longleftrightarrow & s.l.c. \quad W^\top \lambda_k \leq q_k \\ & & y_k \geq 0. & & \end{array}$$

Ces modèles constituent tout simplement les formulations primales et duales des problèmes de deuxième étape associés aux vecteurs ξ_k $(k = 1, \ldots, K)$ étant donné des décisions de première étape fixées à x.

La prochaine étape à effectuer dans le cadre de cette décomposition est d'exprimer les modèles P_k, $k = 1, \ldots, K$, en fonction des points et rayons extrêmes de D_k. Selon le théorème de résolution de Goldman [38], en supposant que les polytopes associés aux problèmes D_k soient de plein rang, si l'on considère les ensembles $\Lambda_k = \{\lambda_k \mid W^\top \lambda_k \leq q_k\}$ $(k = 1, \ldots, K)$ et que l'on définit π_k^i, $i = 1, \ldots, n_k^{'}$, et σ_k^j, $j = 1, \ldots, n_k^{''}$, comme étant respectivement les points extrêmes de Λ_k et les rayons extrêmes du cône $\{\lambda_k \mid W^\top \lambda_k \leq 0\}$, alors il est possible d'exprimer Λ_k de la façon suivante : $\Lambda_k = \{\lambda_k = \sum_{i=1}^{n_k^{'}} \mu_i \pi_k^i + \sum_{j=1}^{n_k^{''}} \nu_j \sigma_k^j \mid \sum_{i=1}^{n_k^{'}} \mu_i = 1, \ \mu_i \geq 0, \nu_i \geq 0, \ i = 1, \ldots, n_k^{'} \ et \ j = 1, \ldots, n_k^{''}\}$.

Le théorème fondamental de la programmation mathématique linéaire affirme que, s'il existe une solution optimale à un problème linéaire, alors il en existe une, qui est un point extrême du polyèdre qui lui est associé. En invoquant ce résultat et en supposant pour l'instant que les problèmes D_k sont bornés étant donné x, pour $k = 1, \ldots, K$, il est possible d'exprimer D_k de la façon suivante :

$$(D_k) \quad Max_{i=1,\ldots,n'_k} \ (h_k - T_k x)^\top \pi_k^i. \tag{2.13}$$

Nous devons à présent nous assurer que les décisions de première étape x sont telles que pour $k = 1, \ldots, K$, alors $\{y \in \mathbb{R}^{n_2} \mid Wy_k = h_k - T_k x, \ y_k \geq 0\} \neq \emptyset$. Cette condition revient à dire qu'il faut s'assurer que les problèmes P_k sont réalisables. Pour ce faire, en se référant au théorème fort de la dualité, il suffit d'obliger les problèmes D_k à être bornés. Nous y arrivons en imposant les contraintes $(h_k - T_k x)^\top \sigma_k^j \leq 0$, $j = 1, \ldots, n''_k$, à chacun des problèmes D_k définis en (2.13). Nous obtenons donc, pour $k = 1, \ldots, K$, les problèmes suivants :

$$\begin{aligned} (D_k) \quad & Max_{i=1,\ldots,n'_k} \ (h_k - T_k x)^\top \pi_k^i \\ & s.l.c. \qquad\qquad (h_k - T_k x)^\top \sigma_k^j \leq 0, \ j = 1, \ldots, n''_k, \end{aligned}$$

qui sont équivalents à :

$$(D_k) \quad Min \ \Theta_k \tag{2.14}$$

$$s.l.c. \quad (h_k - T_k x)^\top \pi_k^i \leq \Theta_k, \ i = 1, \ldots, n'_k \tag{2.15}$$

$$(h_k - T_k x)^\top \sigma_k^j \leq 0, \quad j = 1, \ldots, n''_k. \tag{2.16}$$

Le théorème fort de la dualité, qui nous assure que P_k a une solution si et seulement si D_k en a une également et que les valeurs de celles-ci sont égales, nous permet de

remplacer P_k par (2.14)-(2.16) au sein du modèle (2.10)-(2.12). Ce faisant, on obtient :

$$Min \quad c^\top x + \sum_{k=1}^{K} p_k \Theta_k \tag{2.17}$$

$$s.l.c. \quad Ax = b \tag{2.18}$$

$$(h_k - T_k x)^\top \pi_k^i \leq \Theta_k, \ i = 1, \ldots, n_k^{'}, \ k = 1, \ldots, K \tag{2.19}$$

$$(h_k - T_k x)^\top \sigma_k^j \leq 0, \quad j = 1, \ldots, n_k^{''}, \ k = 1, \ldots, K \tag{2.20}$$

$$x \in X, \tag{2.21}$$

que l'on désigne comme étant le problème maître et qui est équivalent à (2.6)-(2.9). Les contraintes (2.19) représentent les coupes d'optimalité. Ces contraintes expriment les valeurs possibles de la fonction de recours en fonction des variables de première étape x et des points extrêmes associés aux polyèdres des problèmes de deuxième étape. Les contraintes (2.20) définissent quant à elles les coupes de réalisabilité. Ces inégalités forcent les valeurs de x à être telles que les problèmes P_k sont réalisables. En se référant à la classification de Geoffrion [26, 27], le modèle (2.19)-(2.20) constitue le résultat de la linéarisation externe du terme projeté en (2.10)-(2.12). L'approche qui est décrite ici est la version multi-coupes « L-shaped » présentée par Birge et Louveaux [6]. Selon cette formulation, des coupes d'optimalité et de réalisabilité sont comprises pour chacun des ξ_k, $k = 1, \ldots, K$. Il est important de noter qu'à l'origine, dans la version présentée par Van Slyke et Wets [66], les coupes d'optimalité sont agrégées en fonction des réalisations possibles de l'événement aléatoire. Dans les deux cas, le problème maître obtenu est équivalent au problème original (2.6)-(2.9). Il n'y a que l'ensemble des coupes d'optimalité (2.19) qui est exprimé différemment.

L'expression du modèle (2.17)-(2.21) nécessite la connaissance de tous les points et rayons extrêmes associés aux problèmes de deuxième étape. Puisqu'il est difficile, voire impossible, d'obtenir toute cette information a priori, l'approche employée afin de résoudre (2.17)-(2.21) est fondée sur une stratégie de relaxation. Nous présentons ici l'algorithme « L-shaped » multi-coupes de Birge et Louveaux [6], et pour ce faire, nous utilisons la même notation que les auteurs. Dans ce qui suit, W^i et T_k^i représentent respectivement les $i^{\text{ème}}$ lignes des matrices W et T_k, de même que h_k^i représente la $i^{\text{ème}}$

composante du vecteur h_k. Le modèle (2.22)-(2.26) est une version relaxée du problème maître où (2.24) et (2.25) constituent respectivement des sous-ensembles par rapport aux ensembles définis par les coupes de réalisabilité (2.20) et d'optimalité (2.19).

Algorithme multi-coupes L-Shaped :

Étape 0 (initialisation) :

$s = 0$,

$\nu = 0$,

$t_k = 0,\ k = 1, \ldots, K$;

Étape 1 (résoudre problème maître relaxé) :

$\nu = \nu + 1$

Résoudre le problème suivant :

$$Min \quad c^\top x + \sum_{k=1}^{K} \Theta_k \tag{2.22}$$

$$s.l.c. \quad Ax = b \tag{2.23}$$

$$D_l x \geq d_l,\ l = 1, \ldots, s \tag{2.24}$$

$$E_{l_k} x + \Theta_k \geq e_{l_k},\ l_k = 1, \ldots, t_k,\ k = 1, \ldots, K \tag{2.25}$$

$$x \in X. \tag{2.26}$$

Soit $(x^\nu,\ \Theta_1^\nu, \ldots, \Theta_K^\nu)$ la solution optimale au problème précédent.

Si aucune contrainte de type (2.25) n'est présente pour un certain k, on posera $\Theta_k^\nu = -\infty$ et on ne le considérera pas au niveau de la résolution de (2.22)-(2.26).

Étape 2 (coupes de réalisabilité) :

Pour $k = 1, \ldots, K$, soit le problème suivant :

$$Min \quad w^1 = \sum_{i=1}^{m_2} v_i^+ + \sum_{i=1}^{m_2} v_i^- \tag{2.27}$$

$$s.l.c. \quad W^i y + v_i^+ - v_i^- = h_k^i - T_k^i x^\nu,\ i = 1, \ldots, m_s \tag{2.28}$$

$$y \geq 0,\ v_i^+ \geq 0,\ v_i^- \geq 0,\ i = 1, \ldots, m_s. \tag{2.29}$$

Résoudre (2.27)-(2.29) jusqu'à obtenir un k tel que la valeur optimale soit $w^1 > 0$. Définissons les multiplicateurs du simplexe associés à ce problème comme étant σ^ν et posons :

$$D_{s+1} = \sigma^\nu T_k \tag{2.30}$$

$$d_{s+1} = \sigma^\nu h_k. \tag{2.31}$$

Ajouter une coupe de type (2.24) au problème maître relaxé (2.22)-(2.26), poser $s = s + 1$ et retourner à **Étape 1**.

Si pour $k = 1, \ldots, K$, $w^1 = 0$, aller à **Étape 3**.

Étape 3 (coupes d'optimalité) :

Pour $k = 1, \ldots, K$, résoudre le problème suivant :

$$Min \quad w^2 = q_k^\top y \tag{2.32}$$

$$s.l.c. \quad Wy = h_k - T_k x^\nu, \tag{2.33}$$

$$y \geq 0. \tag{2.34}$$

Définissons π_k^ν comme étant les multiplicateurs du simplexe associés à la solution optimale du problème k.

Si

$$\theta_k^\nu < p_k (h_k - T_k x^\nu)^\top \pi_k^\nu, \tag{2.35}$$

alors on définit :

$$E_{t_k+1} = p_k \pi_k^\nu T_k \tag{2.36}$$

$$e_{t_k+1} = p_k \pi_k^\nu h_k. \tag{2.37}$$

Ajouter une coupe de type (2.25) au problème maître relaxé (2.22)-(2.26), poser $t_k = t_k + 1$.

Si, pour $k = 1, \ldots, K$, (2.35) n'est pas vérifié, alors **STOP**, x^ν = solution optimale.

Sinon, retourner à **Étape 1**.

Certaines modifications s'imposent lorsque les variables de deuxième étape sont entières. Caroe et Tind [10] démontrent la convergence de l'algorithme « L-shaped » dans le cas où le recours est entier, c'est-à-dire $y_k \in Y = \{y \in \mathbb{Z}^{n_3} \mid y \geq 0\}$, $k = 1, \ldots, K$. De façon à présenter l'approche dans ce cas particulier, les auteurs apportent un certain nombre de précisions. Au sein de l'algorithme « L-shaped », les coupes de réalisabilité et d'optimalité sont générées à partir de l'information fournie par les problèmes duaux de deuxième étape. Étant donné les contraintes $y_k \in Y$, $k = 1, \ldots, K$, la formulation duale linéaire ne peut-être utilisée dans ce cas.

Nemhauser et Wolsey [48] présentent un concept de dualité s'applicant au cas des problèmes en nombres entiers. Nous utilisons ici la même notation que ces auteurs. Dans un premier temps, ils définissent le problème suivant :

$$(IP) \quad z_{IP} = Min\{c^\top x : \ x \in S\}, \ S = \{x \in \mathbb{Z}^n : Ax \geq b, x \geq 0\}$$

de même que l'ensemble des problèmes :

$$z(d) = Min\{c^\top x : \ x \in S(d)\}, \ S(d) = \{x \in \mathbb{Z}^n : Ax \geq d, x \geq 0\} \text{ pour } d \in D$$

où la matrice A et le vecteur c sont fixes, tandis que d est un paramètre tel que $d \in D \subseteq \mathbb{R}^m$. Nemhauser et Wolsey démontrent que la fonction $z(d)$, qu'ils désignent comme étant la fonction des valeurs de IP, est non décroissante sur le domaine $\mathbb{R}^m$, de même que sous-additive sur $D = \{d \in \mathbb{R}^m : S(d) \neq \emptyset\}$. Rappelons qu'une fonction $g : D \subseteq \mathbb{R}^m \rightarrow \mathbb{R}^1$ est sous-additive si $g(d_1) + g(d_2) \geq g(d_1 + d_2)$, $\forall d_1, \ d_2, \ d_1 + d_2 \in D$. De façon à simplifier la présentation des résultats sur la dualité, les auteurs font l'hypothèse que $z(0) = 0$. Tel que démontré en [48], cette hypothèse permet de s'assurer que : $z(d) < \infty$ pour tout $d \in \mathbb{R}^m$.

Toute fonction $g(d) : \mathbb{R}^m \rightarrow \mathbb{R}^1$ tel que $g(d) \leq z(d)$, $\forall d \in \mathbb{R}^m$, constitue une borne inférieure par rapport à z_{IP}. Par conséquent, il est possible d'obtenir une formulation duale pour le problème IP de la façon suivante :

$$Max\{g(b) : g(d) \leq z(d) \text{ pour } d \in \mathbb{R}^m, \ g(d) : \mathbb{R}^m \rightarrow \mathbb{R}^1\},$$

ce qui est équivalent à :

$$Max\{g(b) : g(d) \leq c^\top x \text{ pour } x \in S(d) \text{ et } d \in \mathbb{R}^m\}. \tag{2.38}$$

Puisque la fonction $z(d)$ est non décroissante sur $\mathbb{R}^m$, il est normal de s'attendre à ce que les fonctions candidates à être des bornes inférieures par rapport à z_{IP} le soient aussi. En imposant cette condition aux fonctions $g(d)$, nous obtenons le résultat suivant :

Proposition 2.1.1. (Nemhauser et Wolsey [48]) *Si $g(d)$ est non décroissante sur $d \in \mathbb{R}^m$, alors $\forall d \in \mathbb{R}^m$, $g(d) \leq c^\top x$ pour $x \in S(d) \iff g(Ax) \leq c^\top x$ pour $x \in \mathbb{Z}^n_+$.*

Démonstration. Si g est non décroissante alors :
(1) $g(d) \leq c^\top x$ pour $x \in S(d) \Rightarrow g(Ax) \leq c^\top x$ pour $x \in \mathbb{Z}^n_+$.
Pour $x \in \mathbb{Z}^n_+$, posons $Ax = \tilde{d}$.
Puisque $x \in S(\tilde{d})$ alors $g(Ax) = g(\tilde{d}) \leq c^\top x$.
Cet argument est vrai $\forall x \in \mathbb{Z}^n_+$.
(2) $g(Ax) \leq c^\top x$ pour $x \in \mathbb{Z}^n_+ \Rightarrow g(d) \leq c^\top x$ pour $x \in S(d)$.
$\forall x \in S(d)$, puisque $Ax \geq d$ alors $g(Ax) \geq g(d)$, car g est une fonction non décroissante sur $d \in \mathbb{R}^m$.
Puisque $g(Ax) \leq c^\top x$, alors $g(d) \leq c^\top x$. □

En utilisant la proposition précédente, il est possible de réécrire le problème (2.38) comme étant :

$$Max \quad g(b) \tag{2.39}$$

$$s.l.c. \quad g(Ax) \leq c^\top x \text{ pour } x \in \mathbb{Z}^n_+ \tag{2.40}$$

$$g \text{ non décroissante.} \tag{2.41}$$

À présent, si on impose la condition supplémentaire dictant les fonctions $g(d)$ à être sous-additives puisque $z(d)$ l'est également, il est possible de démontrer la proposition suivante :

Proposition 2.1.2. (Nemhauser et Wolsey [48]) *Si les fonctions g sont sous-additives, alors $g(Ax) \leq c^\top x$ pour $x \in \mathbb{Z}^n_+ \iff g(a_j) \leq c_j$ pour $j \in N$, où a_j et c_j correspondent respectivement à la $j^{ième}$ colonne de la matrice A et à la $j^{ième}$ composante du vecteur c, où $j \in N = \{1, \ldots, n\}$.*

En employant ce résultat, le problème dual (2.39)-(2.41) peut être écrit de la façon suivante :

$$
\begin{aligned}
(SDP)\ w = Max \quad & g(b) \\
s.l.c. \quad & g(a_j) \leq c_j \text{ pour } j \in N \\
& g(0) = 0 \\
& g : \mathbb{R}^m \rightarrow \mathbb{R}^1, \text{ non décroissante et sous-additive}
\end{aligned}
$$

La contrainte $g(0) = 0$ est imposée au problème SDP, étant donné l'hypothèse $z(0) = 0$. À présent, à partir de IP et SDP, il est possible de démontrer le théorème de la dualité forte (la démonstration de ce résultat est faite au chapitre II.3 du livre de Nemhauser et Wolsey [48]).

Théorème de la dualité forte (cas entier).
(1) Si IP est réalisable alors SDP l'est aussi et $w = z(b)$.
(2) Si IP n'est pas réalisable alors SDP n'est pas borné.

Caroe et Tind [10] utilisent ces résultats au sein de l'algorithme « L-shaped ». Ces auteurs considèrent le cas des problèmes de type (1.1)-(1.5), où les variables de première étape sont continues et celles de deuxième étape sont entières. Dans la version qui est traitée, le vecteur q, présent dans la fonction objectif des problèmes de deuxième étape, est constant pour $k = 1, \ldots, K$. Par conséquent, le problème à résoudre s'énonce comme suit :

$$
\begin{aligned}
Min \quad & c^\top x + \sum_{k=1}^{K} p_k q^\top y_k \\
s.l.c. \quad & Ax \geq b \\
& Wy_k \geq h_k - T_k x \\
& x \geq 0,\ y_k \in \mathbb{Z}_+^{n_2},\ k = 1, \ldots, K.
\end{aligned}
$$

Afin de s'assurer que les problèmes de deuxième étape sont toujours réalisables, et ce, peu importe le terme de droite, il est supposé que : $\exists u \in \mathbb{R}_+^{m_2}$ tel que $W^\top u \leq q$.

En utilisant la formulation duale exposée précédemment, il est possible d'exprimer

les problèmes de deuxième étape de la façon suivante :

$$(P_k)\quad \begin{array}{ll} \text{Min} & q^\top y_k \\ s.l.c. & Wy_k \geq h_k - T_k x \\ & y_k \in \mathbb{Z}_+^{n_2} \end{array} \quad \Longleftrightarrow \quad (D_k)\quad \begin{array}{ll} \text{Max} & F(h_k - T_k x) \\ s.l.c. & F(Wy_k) \leq q^\top y_k, \forall y_k \in \mathbb{Z}_+^{n_2} \\ & F(0) = 0 \\ & F \text{ non décroissante.} \end{array}$$

En posant $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty\} \cup \{+\infty\}$, les auteurs définissent $\overline{\mathcal{F}}$ comme étant l'ensemble des fonctions $F : \mathbb{R}^{m_2} \rightarrow \overline{\mathbb{R}}$ tel que $F(0) = 0$ et F est non décroissante. Par conséquent, le problème D_k devient :

$$\begin{array}{ll} Max & F(h_k - T_k x) \\ s.l.c. & F(Wy_k) \leq q^\top y_k, \forall y_k \in \mathbb{Z}_+^{n_2} \\ & F \in \overline{\mathcal{F}}. \end{array}$$

Si l'on considère une solution réalisable $\widehat{x}$ et que l'on pose $\widehat{F}^k$ (pour $k = 1, \ldots, K$) comme étant les solutions optimales aux problèmes D_k lorsque les décisions de première étape sont fixées à $\widehat{x}$, alors il est possible de démontrer que $\Theta \geq \sum_{k=1}^{K} p_k \widehat{F}^k(h_k - T_k x)$ constitue une coupe d'optimalité pour le problème maître relaxé. Des principes similaires s'appliquent lorsqu'il est question de vérifier si une solution $\widehat{x} \in \{x \in \mathbb{R}_+^{n_1} \mid Ax \geq b\}$ est réalisable en deuxième étape. Pour ce faire, il faudra résoudre pour $k = 1, \ldots, K$:

$$\begin{array}{ll} Min & e^\top t \\ s.l.c. & Wy_k + It \geq h_k - T_k \widehat{x} \\ & y_k \in \mathbb{Z}_+^{n_2},\ t \in \mathbb{Z}_+^{m_2}, \end{array}$$

où $e = [1, \ldots, 1]$. Si pour un k donné, on obtient une solution optimale t^* telle que $e^\top t^* \geq 0$, en posant $\widehat{G}$ comme étant la solution optimale au problème dual $\text{Max}\{G(h_k - T_k\widehat{x}) \mid G(Wy_k + It) \leq e^\top t, \forall (y_k, t) \in \mathbb{Z}_+^{n_2} \times \mathbb{Z}_+^{m_2}, G \in \overline{\mathcal{F}}\}$, alors il est possible de démontrer que $0 \geq \widehat{G}(h_k - T_k x)$ est une coupe de réalisabilité pour le problème maître relaxé.

Pour terminer, en fixant $\mathcal{F}$ tel que $\mathcal{F} \subseteq \overline{\mathcal{F}}$, Caroe et Tind définissent le $\mathcal{F} - dual$

de P_k comme étant :

$$Max\ F(h_k - T_k x) \tag{2.42}$$

$$s.l.c.\quad F(W y_k) \leq q^\top y_k \text{ pour tout } y_k \in \mathbb{Z}_+^{n_2} \tag{2.43}$$

$$F \in \mathcal{F}. \tag{2.44}$$

Il est possible d'obtenir le résultat de la dualité forte pour les problèmes P_k et (2.42)-(2.44) lorsque le sous-ensemble $\mathcal{F}$ est suffisamment grand, c'est-à-dire lorsqu'il permet d'obtenir un écart de dualité nul. Le sous-ensemble $\mathcal{F}$ dépend de l'approche algorithmique employée pour résoudre les problèmes entiers de deuxième étape. Deux approches de résolution sont considérées par Caroe et Tind en [10] : une méthode de coupes et une approche par séparation et évaluation progressive. En utilisant ces deux approches, les auteurs démontrent comment il est possible de construire les coupes d'optimalité et de réalisabilité générées par l'algorithme « L-shaped ». Ceci conclut notre présentation générale de la décomposition de Benders telle qu'elle s'applique au cas de la programmation stochastique en nombres entiers (algorithm « L-shaped »). Nous allons à présent décrire certaines des approches méthodologiques générales, reposant sur la génération de coupes, qui ont été développées dans le but de résoudre des problèmes de type (1.1)-(1.5).

Wollmer [69] présente une approche de résolution de type « L-shaped », s'appliquant au cas où les variables de première étape sont binaires et celles de deuxième étape sont continues. Dans cette approche, le problème maître relaxé n'est pas résolu directement, l'algorithme proposé effectue plutôt une recherche exhaustive des solutions possibles de première étape. Toute solution partielle identifiée subit un test permettant de vérifier s'il s'agit d'une solution réalisable en considérant les contraintes du problème maître relaxé. Dans le cas où l'une des contraintes n'est pas respectée, l'algorithme élimine l'ensemble des solutions complètes pouvant découler de celle-ci. S'il s'agit d'une solution réalisable au problème maître relaxé, alors, selon le cas, la procédure génère une coupe de réalisabilité ou d'optimalité, qui est par la suite ajoutée au modèle. L'efficacité de cette approche dépend directement de sa capacité d'éliminer des zones de recherche du domaine réalisable de première étape. Une série de tests est donc présentée afin

d'améliorer de façon significative le processus d'élimination.

Averbakh [3] propose également un algorithme de type « L-shaped » pour résoudre des problèmes de programmation stochastique en nombres entiers. Les problèmes traités sont tels que la matrice technologique est constante et celle du recours varie en fonction de l'événement aléatoire. L'auteur suppose également que l'événement aléatoire suit une distribution continue. Dans ce cas, le problème à résoudre est formulé de la façon suivante (nous utiliserons ici la même notation que Haneveld et van der Vlerk [36]) :

$$\begin{aligned} Max \quad & G(x) + E_\omega[C(\omega, y(\omega))] \\ s.l.c. \quad & E_\omega[A_i(\omega, y(\omega))] \leq B_i(x), \ i = 1, \ldots, m_2 \\ & y(\omega) \in Y(\omega) \subset \mathbb{R}^{n_2} \\ & x \in X. \end{aligned}$$

Il est important de noter que, pour les problèmes considérés, les fonctions $C(\omega, y(\omega))$ et $A_i(\omega, y(\omega))$ ne sont pas tenues d'être convexes. De plus, des contraintes d'intégralité peuvent être imposées au niveau des variables de deuxième étape. Averbakh démontre la convexité des problèmes de deuxième étape, en effectuant quelques suppositions sur $C(\omega, y(\omega))$ et $A_i(\omega, y(\omega))$, $i = 1, \ldots, m_2$. Une formulation duale pour les problèmes de deuxième étape est définie par la suite, ce qui permet d'effectuer une décomposition du type Benders.

L'algorithme « L-shaped » est spécialisé par Laporte et Louveaux [39] au cas où les variables de première étape sont binaires et où le recours est entier. Dans les cas traités, $\{x \mid Ax = b, x \in X\} \subseteq \{x \mid \mathcal{Q}(x) < \infty\}$ (propriété du recours relativement complet) et il est supposé que la fonction de recours $\mathcal{Q}$ est calculable à partir d'une solution x donnée. De plus, les problèmes à résoudre sont tels qu'il est possible d'obtenir une borne inférieure L telle que $L \leq min_x\{\mathcal{Q}(x) \mid Ax = b, \ x \in \{0,1\}^{n_1}\}$. Les auteurs définissent un ensemble de t coupes d'optimalité comme étant valide si $\forall x \in X$, $(x, \Theta) \in \{(x, \Theta) \mid E_k x + \Theta \geq e_k, \ k = 1, \ldots, t\} \Rightarrow \Theta \geq \mathcal{Q}(x)$. Puisque les problèmes considérés impliquent des variables de première étape binaires, alors il existe un nombre fini de solutions x réalisables. Posons $r = 1, \ldots, R$, comme étant les indices représentant ces solutions. Il

est démontré en [39] que :

$$\Theta \geq (\Theta_r - L)\Big(\sum_{i \in S_r} x_i - \sum_{i \notin S_r} x_i\Big) - (\Theta_r - L)(|S_r| - 1) + L, \text{ pour } r = 1, \ldots, R \quad (2.45)$$

où la $r^{\text{ième}}$ solution réalisable est dénotée par $x_i = 1,\ i \in S_r$ et $x_i = 0,\ i \notin S_r$ et Θ_r est la valeur de la fonction de recours pour cette solution, constituent un ensemble valide de coupes d'optimalité. Des principes similaires sont utilisés afin de définir un ensemble valide de coupes de réalisabilité. Si $X = \overline{X} \cap \{0,1\}^{n_1}$, alors un ensemble de coupes de réalisabilité est valide s'il existe un nombre fini s tel que $x \in \overline{X}$ si et seulement si $\{D_k x \geq d_k,\ k = 1, \ldots, s\}$. L'algorithme « L-shaped » est implanté selon un schème de type séparation et plans coupants où des coupes de réalisabilité et d'optimalité sont ajoutées, selon les besoins, au problème maître relaxé. Il est démontré que l'algorithme converge en un nombre fini d'itérations pour tout problème où il existe des ensembles valides de coupes d'optimalité et de réalisabilité.

Il est possible d'améliorer la qualité du problème maître, en y ajoutant des inégalités valides permettant de borner plus efficacement $\mathcal{Q}(x)$. Des inégalités valides peuvent être obtenues à partir de la relaxation linéaire des problèmes de deuxième étape. Plus précisément, en posant $R(x, \xi(\omega)) = min_y\{q(\omega)^\top y \mid Wy = h(\omega) - T(\omega)x,\ y \geq 0\}$ et $R(x) = \mathbf{E}_\xi[R(x, \xi(\omega))]$, les auteurs démontrent que $\Theta \geq R(x^k) + \partial R(x^k)(x - x^k)$ où x^k est une solution réalisable de première étape et $\partial R(x^k)$ est un sous-gradient de R à cette solution, constitue une inégalité valide pour le problème maître relaxé. Les auteurs montrent également comment il est possible d'améliorer les coupes d'optimalité en identifiant de meilleures bornes inférieures pour des solutions voisines de celle qui est utilisée pour générer une inégalité de type (2.45).

Des approches de résolution utilisant la programmation disjonctive ont également été développées dans le but de résoudre des problèmes de type (1.1)-(1.5) (voir [9], [61], [58] et [59]). Supposons que les problèmes de deuxième étape sont définis comme suit : $Q(x, \xi(\omega)) = Min_y\{q^\top y \mid Wy \geq h(\omega) - T(\omega)x,\ y \in Y\}$, où $Y = \{y = [y^1, \ldots, y^{n_2}]^\top \mid y^i \in \mathbb{R}_+,\ i = 1, \ldots, p,\ y^j \in \{0,1\},\ j = p+1, \ldots, n_2\}$. Puisqu'un sous-ensemble des variables de deuxième étape est binaire, tel qu'indiqué précédemment, les principes de la dualité linéaire ne peuvent être appliqués directement au sein de l'algorithme

de Benders. Définissons $X = \{x \mid Ax \geq b,\ 0 \leq x \leq 1\}$, comme étant l'ensemble comportant les contraintes linéaires de première étape du problème (1.1)-(1.5). Posons également $x = \hat{x} \in X$, en supposant que $Q(\hat{x}, \xi(\omega)) < +\infty$, $\forall \omega \in \Omega$. Il est démontré par Sen et Higle [58] que pour $\hat{\omega} \in \Omega$, à partir de $Q(\hat{x}, \xi(\hat{\omega})) = Min_y\{q^\top y \mid Wy \geq h(\hat{\omega}) - T(\hat{\omega})\hat{x},\ y \in Y\}$, en utilisant des disjonctions sur les variables binaires du vecteur y, il est possible de trouver des inégalités de la forme suivante :

$$\eta^\top y \geq \eta_0(x, \omega), \tag{2.46}$$

qui sont valides $\forall (x, \omega) \in X \times \Omega$. Par conséquent, pour un nombre fini L de ces inégalités, nous obtenons le problème suivant :

$$Q_0^L(x, \xi(\omega)) = Min \quad q^\top y \tag{2.47}$$

$$s.l.c. \qquad Wy \geq h(\omega) - T(\omega)x \tag{2.48}$$

$$\eta_l^\top y \geq \eta_0^l(x, \omega),\ l = 1, \ldots, L \tag{2.49}$$

$$y^j \geq 0,\ j = 1, \ldots, p \tag{2.50}$$

$$0 \leq y^j \leq 1,\ j = p+1, \ldots, n_2 \tag{2.51}$$

qui est tel que $Q(x, \xi(\omega)) \geq Q_0^L(x, \xi(\omega))$, $\forall (x, \omega) \in X \times \Omega$. Le problème (2.47)-(2.51) comporte plusieurs avantages. Premièrement, puisque le terme de droite des contraintes (2.49) ne dépend que de x et ω ($\eta_0^l : X \times \Omega \rightarrow \mathbb{R}$, pour $l = 1, \ldots, L$), en mesurant les valeurs $\eta_0^l(x, \omega)$ pour un x et ω donné, le problème (2.47)-(2.51) définit une borne inférieure valide pour $Q(x, \xi(\omega))$. Un algorithme de type génération de plans coupants peut être utilisé par la suite afin d'ajouter de nouvelles inégalités de type (2.46). Ce faisant, il est possible d'améliorer la borne inférieure utilisée (voir Sen et Higle [58]), ou alors, de résoudre à l'optimalité le problème de deuxième étape considéré (voir Sherali et Fraticelli [61]). Un autre grand avantage de cette approche est que le modèle (2.47)-(2.51) n'implique que des variables continues. Par conséquent, dans le contexte de la décomposition de Benders, il est possible de générer des coupes en utilisant la solution optimale duale associée à (2.47)-(2.51). Lorsqu'un algorithme de type séparation et plans coupants est utilisé pour obtenir $Q(x, \xi(\omega))$, Sen et Sherali [59] démontrent qu'il est possible de générer des inégalités valides sur l'espace défini par X, en utilisant l'information fournie par l'arbre d'énumération obtenu par l'algorithme. Ce résultat

découle directement de l'observation que tout arbre d'énumération définit une disjonction. Finalement, il est important de préciser que, si des contraintes d'intégralité sont présentes au niveau des variables de première étape, il est toujours possible d'explorer l'ensemble X en utilisant une approche de type séparation et évaluation progressive.

Nous concluons cette section en présentant brièvement l'article de van der Vlerk [64]. L'auteur démontre comment il est possible de trouver des bornes inférieures pour la fonction de recours $\mathcal{Q}(x)$ en modifiant quelque peu la fonction de distribution associée à l'événement aléatoire. Les bornes ainsi trouvées améliorent strictement l'information générée par rapport à celle obtenue de la relaxation linéaire. De plus, pour les problèmes où la matrice W est totalement unimodulaire (ce qui est le cas des problèmes de flots), alors l'écart entre la borne et la valeur optimale est nul. Par conséquent, il est possible d'utiliser ces résultats au sein d'algorithmes de coupes (et autres), de façon à améliorer la qualité des modèles traités lors du processus de résolution. Cependant, il est important de noter que, dans les cas considérés, le vecteur aléatoire ne comprend que le terme de droite associé aux contraintes de deuxième étape (soit $\xi(\omega) = [h(\omega)^\top]$).

2.2 Algorithmes de type séparation et évaluation progressive

Nous commençons cette section en présentant l'article de Caroe et Schultz [8]. Dans le cadre de cet article, les auteurs cherchent à résoudre des problèmes où des contraintes d'intégralité sont imposées aux variables de première et deuxième étape. Pour ce faire, ils proposent une approche de type séparation et évaluation progressive, où des bornes sont obtenues en appliquant la relaxation lagrangienne aux sous-problèmes traités par l'algorithme. L'idée principale derrière ces bornes est tirée de la décomposition utilisée au sein du « progressive hedging algorithm » de Rockafellar et Wets [52].

Le principe de décomposition proposé par Rockafellar et Wets consiste à séparer le problème stochastique en fonction des réalisations possibles de l'événement aléatoire. Pour ce faire, définissons premièrement un vecteur de variables de première étape pour

chacune des réalisations possibles de ξ (soit x_k, $k = 1, \ldots, K$). Puisqu'il est impossible d'anticiper l'événement aléatoire, ces variables doivent être identiques au sein du nouveau modèle (c'est-à-dire $x_1 = x_2 = \ldots = x_K$). Ces contraintes supplémentaires, que l'on nomme contraintes de non-anticipativité, nous assurent que les étapes de décision sont bien respectées. À présent, le problème stochastique peut-être reformulé de la façon suivante (nous utilisons ici la notation de Caroe et Schultz [8]) :

$$Min \quad \sum_{k=1}^{K} p_k(c^\top x_k + q_k^\top y_k) \tag{2.52}$$

$$s.l.c. \quad Ax_k = b, \qquad k = 1, \ldots, K \tag{2.53}$$

$$T_k x_k + W y_k = h_k, \; k = 1, \ldots, K \tag{2.54}$$

$$\sum_{k=1}^{K} H_k x_k = 0 \tag{2.55}$$

$$x_k \in X, \; y_k \in Y, \quad k = 1, \ldots, K, \tag{2.56}$$

où (2.55) constitue une expression générale des restrictions de non-anticipativité (les matrices H_k étant de dimension $l \times n_1$).

À partir du problème (2.52)-(2.56), en ayant recours à la relaxation lagrangienne, il est possible de traduire les contraintes de non-anticipativité dans l'objectif à optimiser. Ce faisant, pour tout $\lambda \in \mathbb{R}^l$, on obtient la fonction lagrangienne suivante :

$$D(\lambda) = \; Min \quad \sum_{k=1}^{K} p_k(c^\top x_k + q_k^\top y_k) + \lambda^\top H_k x_k \tag{2.57}$$

$$s.l.c. \quad Ax_k = b, \qquad k = 1, \ldots, K \tag{2.58}$$

$$T_k x_k + W y_k = h_k, \; k = 1, \ldots, K \tag{2.59}$$

$$x_k \in X, \; y_k \in Y, \quad k = 1, \ldots, K. \tag{2.60}$$

En posant $D_k(\lambda) = \; Min\{p_k(c^\top x_k + q_k^\top y_k) + \lambda^\top H_k x_k : \; Ax_k = b, \; T_k x_k + W y_k = h_k, \; x_k \geq 0, \; y_k \geq 0, \; k = 1, \ldots, K\}$, (2.57)-(2.60) devient : $D(\lambda) = \sum_{k=1}^{K} D_k(\lambda)$. En fixant λ, nous constatons que $D(\lambda)$ se décompose en fonction des scénarios de l'événement aléatoire. Finalement, le dual lagrangien associé à $D(\lambda)$, qui s'énonce de la façon suivante : $Z_{LD} = Max \; \{D(\lambda) : \; \lambda \in \mathbb{R}^l\}$, permet d'obtenir une borne inférieure par rapport à (2.52)-(2.56).

Caroe et Schultz [8] appliquent une stratégie de résolution de type séparation et évaluation progressive au problème (2.52)-(2.56). L'idée est d'effectuer une recherche exhaustive du domaine réalisable de première étape, en utilisant Z_{LD} afin d'obtenir des bornes inférieures pour les sous-problèmes générés lors de l'exploration. Afin d'accélérer le processus de recherche, une heuristique permettant de construire des solutions réalisables à partir de points non-réalisables est également présentée.

Le prochain article présenté dans cette section est celui de Norkin et *al.* [49]. Dans le cadre de cette recherche, les auteurs proposent un algorithme de séparation et évaluation progressive de type probabiliste. Certains problèmes de programmation stochastique impliquent des événements aléatoires dont le nombre de réalisations peut être très grand, voire infini. Dans ce cas, il est bien souvent impossible de mesurer exactement la fonction de recours. Les auteurs suggèrent de contourner cette difficulté en effectuant une recherche du domaine réalisable, qui est orientée en calculant progressivement, à l'aide d'estimateurs aléatoires, des bornes pour les sous-problèmes traités. L'algorithme ainsi défini converge en probabilité vers la solution optimale.

Considérons le problème stochastique (1.1)-(1.5), où X comprend des contraintes d'intégralité (Y pouvant en contenir ou non). Dans un premier temps, les auteurs définissent $\mathcal{P}$ comme étant une partition de l'ensemble X. Pour tout $X^p \in \mathcal{P}$, la fonction $F^*(X^p)$ est définie comme étant :

$$F^*(X^p) = Min \quad c^\top x + \mathcal{Q}(x) \tag{2.61}$$

$$s.l.c. \quad Ax = b \tag{2.62}$$

$$x \in X^p. \tag{2.63}$$

Étant donné (2.61)-(2.63), le problème original (1.1)-(1.5) peut-être réexprimé comme suit : $F^*(X) = \min_{X^p \in \mathcal{P}} F^*(X^p)$. De façon à pouvoir appliquer l'approche proposée, les auteurs considèrent premièrement qu'il existe des fonctions L et U telles que pour tout $X^p \subset X$ alors $L(X^p) \leq F^*(X^p) \leq U(X^p)$ et lorsque X^p ne contient qu'un seul élément, alors $L(X^p) = F^*(X^p) = U(X^p)$. Les auteurs supposent également que pour tout $X^p \subset X$, il est possible d'estimer aléatoirement $L(X^p)$ et $U(X^p)$. Par conséquent, il existe des fonctions $\phi^n(X^p, \omega)$, $n = 1, 2, \ldots$ et $\psi^m(X^p, \omega)$, $m = 1, 2, \ldots$, telles que

$\lim_{n\to\infty} \phi^n(X^p,\omega) = L(X^p)$ et $\lim_{m\to\infty} \psi^m(X^p,\omega) = U(X^p)$.

En ce qui concerne la définition de $U(X^p)$, il suffit d'identifier une solution réalisable au problème, disons x', tel que $x' \in X^p$. En posant $U(X^p) = c^\top x' + \mathcal{Q}(x')$, on obtient une borne supérieure par rapport à $F^*(X^p)$. Puisque dans le cas considéré, il est impossible d'évaluer exactement la fonction de recours pour une solution donnée, les auteurs proposent d'utiliser la simulation de Monte Carlo afin d'estimer aléatoirement $\mathcal{Q}(x')$. On obtient donc la fonction suivante : $\psi^m(X^p,\omega) = c^\top x' + \frac{1}{m}\sum_{i=1}^{m} Q(x,\xi(\omega)_i)$, où $\xi(\omega)_i$ $(i = 1,\ldots,m)$ est un échantillon tiré aléatoirement par rapport à ξ.

Afin d'obtenir des bornes inférieures pour $F^*(X^p)$, les auteurs utilisent premièrement la relaxation lagrangienne. L'idée proposée est de relaxer les contraintes linéaires associées aux problèmes de deuxième étape du modèle (2.61)-(2.63). Par la suite, la fonction $L(X^p)$ est définie comme étant la valeur optimale au problème dual lagrangien résultant. Des bornes inférieures peuvent également être obtenues, en interchangeant l'espérance mathématique et l'objectif à optimiser au sein de (2.61)-(2.63). Ceci revient à mesurer la valeur de la borne « wait and see » définie par Madansky [44]. En posant $WS = \mathbf{E}_\xi\Big[\min_{x\in X\cap\{x|Ax=b\}} c^\top x + Q(x,\xi(\omega))\Big]$, Madansky démontre que $WS \leq c^\top x^\star + \mathcal{Q}(x^\star)$, où $x^\star$ est la solution optimale au problème stochastique (1.1)-(1.5). La borne WS mesure la valeur du recours dans le cas où il serait possible d'anticiper l'avenir. En combinant ce résultat avec les bornes lagrangiennes précédentes, les auteurs définissent une deuxième borne inférieure pour $F^*(X^p)$. Une fois de plus, puisque le nombre de scénarios ne permet pas de mesurer exactement l'espérance mathématique, les auteurs utilisent la simulation de Monte Carlo afin d'obtenir $\phi^n(X^p,\omega)$.

Nous concluons cette section en présentant l'article d'Ahmed et *al.* [2]. Dans cet article, les auteurs proposent un algorithme de type séparation et évaluation progressive pour résoudre des problèmes stochastiques où les variables de première étape sont continues et où celles des problèmes de deuxième étape sont entières. Dans la problématique traitée, la matrice technologique est déterministe ($T(\omega) = T$, $\forall\omega \in \Omega$), tandis que celle du recours est stochastique ($W(\omega)$, $\omega \in \Omega$). Le problème à résoudre est énoncé de la

façon suivante :

$$Min \quad c^\top x + \sum_{k=1}^{K} p_k Q_k(x) \tag{2.64}$$

$$s.l.c. \quad x \in X \tag{2.65}$$

$$où \quad Q_k(x) = Min_y\{q_k^\top y \mid W_k y \geq h_k + Tx,\ y \in Y \cap \mathbb{Z}^{n_2}\}. \tag{2.66}$$

L'esemble X est considéré comme étant non-vide et compact. Les problèmes de deuxième étape ont la propriété du recours complet ($\forall x \in \mathbb{R}^{n_1}$, $\mathcal{Q}(x) < \infty$) et il est supposé que pour $k = 1, \ldots, K$, $\exists u_k \in \mathbb{R}_+^{m_2}$ tel que $u_k W_k \leq q_k$. Finalement, les coefficients des matrices W_k sont entiers. En posant $\chi = Tx$ et $\mathcal{X} = \{\chi \in \mathbb{R}^{m_2} \mid \chi = Tx,\ x \in X\}$, le modèle (2.64)-(2.66) peut-être exprimé comme étant :

$$Min \quad \phi(\chi) + \overline{\psi}(\chi) \tag{2.67}$$

$$s.l.c. \quad \chi \in \mathcal{X} \tag{2.68}$$

$$où \quad \phi(\chi) = Min\{c^\top x \mid \chi = Tx,\ x \in X\} \tag{2.69}$$

$$\overline{\psi}(\chi) = \sum_{k=1}^{K} p_k \psi^k(\chi) \tag{2.70}$$

$$\psi^k(\chi) = Min\{q_k^\top y \mid W_k y \geq h_k + \chi,\ y \in Y \cap \mathbb{Z}^{n_2}\}. \tag{2.71}$$

De façon à obtenir une procédure de recherche exhaustive, les auteurs démontrent dans un premier temps qu'il est possible de séparer le domaine réalisable des variables χ (soit $\mathcal{X}$) en un nombre fini de sous-ensembles pour lesquels la fonction de recours est constante *. Puisque les coefficients des matrices W_k (pour $k = 1, \ldots, K$) sont entiers, alors $\psi^k(\chi) = Min\{q_k^\top y \mid W_k y \geq h_k + \chi,\ y \in Y \cap \mathbb{Z}^{n_2}\} = Min\{q_k^\top y \mid W_k y \geq \lceil h_k + \chi \rceil,\ y \in Y \cap \mathbb{Z}^{n_2}\}$ et ce, $\forall \chi \in \mathcal{X}$. Il est également important de noter que les fonctions ψ^k ($k = 1, \ldots, K$) sont discontinues lorsque le terme de droite des contraintes qui leur sont associées est entier. À partir de ces constats et en posant $\psi_i^k(\chi_i)$ comme étant $\psi^k(\chi)$ défini en fonction de la $i^{\text{ème}}$ composante de χ ($i = 1, \ldots, m_2$), il est démontré en [2] que : pour tout $s_k^i \in \mathbb{Z}$, la valeur de $\psi_i^k(\chi_i)$ est constante pour $\chi_i \in (s_k^i - h_k^i - 1, s_k^i - h_k^i]$ et ce pour $k = 1, \ldots, K$ et $i = 1, \ldots, m_2$. À présent, pour $\mathbf{s} = (s_1^1, \ldots, s_k^i, \ldots, s_K^{m_2}) \in \mathbb{Z}^{Km_2}$,

*Ce résultat fut également démontré par Schultz et *al.* [55].

en définissant $\mathcal{C}(\mathbf{s}) = \{\chi \in \mathbb{R}^{m_2} \mid \chi \in \cap_{k=1}^{K} \prod_{i=1}^{m_2} (s_k^i - h_k^i - 1, s_k^i - h_k^i]\}$, il est également prouvé que l'ensemble $\{\mathcal{C}(\mathbf{s}) \mid \mathbf{s} \in \mathbb{Z}^{Km_2}\}$ constitue une partition de $\mathbb{R}^{m_2}$ et, si $\mathcal{C}(\mathbf{s}) \neq \emptyset$, alors la fonction $\overline{\psi}(\chi)$ est constante sur $\mathcal{C}(\mathbf{s})$. Finalement, l'ensemble $\mathcal{S} = \{\mathbf{s} \in \mathbb{Z}^{Km_2} \mid \mathcal{C}(\mathbf{s}) \cap \mathcal{X} \neq \emptyset\}$ est tel que $|\mathcal{S}| < \infty$ lorsque $\mathcal{X}$ est compact.

Le processus de séparation proposé par les auteurs consiste à diviser le domaine réalisable $\mathcal{X}$ en régions définies comme suit : $\chi \in \prod_{i=1}^{m_2} (l^i, u^i] \cap \mathcal{X}$, où la fonction de recours est discontinue aux points l^i. Pour identifier ces points, il suffit de trouver des valeurs χ_i pour lesquelles $h_k^i + \chi_i$ est une valeur entière pour au moins un scénario k. En séparant le domaine en fonction de ces valeurs, on obtiendra éventuellement des sous-problèmes pour lesquels la fonction de recours est constante. Puisque dans le cas présent, le nombre de ces sous-problèmes est fini, cela veut dire que la partition ainsi obtenue l'est également.

Afin d'obtenir une borne inférieure pour un sous-problème donné, les auteurs proposent de résoudre le problème suivant :

$$Min \quad \phi(\chi) + \theta \tag{2.72}$$

$$s.l.c. \quad L_j \leq \chi \leq U_j \tag{2.73}$$

$$\theta \geq \sum_{k=1}^{K} p_k \psi^k (L_j + \epsilon) \tag{2.74}$$

$$\chi \in \mathcal{X}. \tag{2.75}$$

Les contraintes (2.73) définissent le sous-problème considéré. Le vecteur $\epsilon \in \mathbb{R}^{m_2}$ est fixé de façon à ce que les fonctions ψ^k, $k = 1, \ldots, K$, soient constantes sur l'intervalle $(L_j, L_j + \epsilon]$. Il est montré en [2] comment trouver un tel vecteur. Puisque, pour $k = 1, \ldots, K$, $\psi^k(\chi) \geq Min\{q_k^\top y \mid W_k y \geq L_j + \epsilon,\ y \in Y \cap \mathbb{Z}^{n_2}\}$ lorsque $\chi \in \mathcal{X}$ et $L_j \leq \chi \leq U_j$, alors (2.74) permet d'obtenir une borne inférieure sur la valeur de la fonction de recours pour le domaine réalisable associé au sous-problème. Il est important de noter que la solution optimale au modèle (2.72)-(2.75), soit χ^j, est une solution réalisable au problème (2.67)-(2.71). Par conséquent, $\phi(\chi^j) + \overline{\psi}(\chi^j)$ constitue une borne supérieure possible pour le problème à résoudre. Une borne supérieure générale peut-être obtenue en conservant la valeur de la meilleure solution identifiée.

2.3 Approches par les ensembles test

Dans cette section nous débutons en présentant ce qu'est un ensemble test. Par la suite, nous verrons comment ces concepts peuvent s'appliquer à la programmation stochastique en nombres entiers. Pour ce qui suit, nous utilisons la même notation que Weismantel [68], qui fait une présentation générale des principaux résultats reliés aux ensembles test.

Considérons le problème à résoudre suivant :

$$Min \;\; c^\top x : \; x \in X = \{x \in \mathbb{Z}^n : \; Ax = b, \; l \leq x \leq u\}, \tag{2.76}$$

où les paramètres sont tels que : $c \in \mathbb{Z}^n$, $b \in \mathbb{Z}^m$, $A \in \mathbb{Z}^{m \times n}$, $l \in (\mathbb{Z} \cup \{-\infty\})^n$ et $u \in (\mathbb{Z} \cup \{\infty\})^n$. Définissons également $IP(b, c)$ comme étant l'ensemble des problèmes du type (2.76) où les paramètres $l = (0)^n$, $u \in (\mathbb{Z} \cup \{\infty\})^n$ et $A \in \mathbb{Z}^{m \times n}$ sont fixés, tandis que $c \in \mathbb{Z}^n$ et $b \in \mathbb{Z}^m$ peuvent varier. De façon similaire, posons $IP(b)$ comme étant l'ensemble des problèmes (2.76), où tous les paramètres sont fixés à l'exception de $b \in \mathbb{Z}^m$ qui est variable.

L'ensemble test associé au problème (2.76) (lorsque $l = (0)^n$), est défini comme étant un sous-ensemble $B \subseteq \mathbb{Z}^n$ tel que : $c^\top v < 0, \forall v \in B$, et pour toute solution réalisable, mais non optimale, au problème (2.76), soit $x \in X$, alors $\exists v \in B$ tel que $x+v$ est réalisable ($x+v \in X$). À partir de B, il est simple de concevoir une approche de résolution pour le problème en nombres entiers (2.76). Il suffit d'identifier une solution réalisable $x \in X$ et de trouver par la suite un vecteur $v \in B$ pour lequel $x + v \in X$. Puisque $c^\top v < 0$, cette nouvelle solution est telle que $c^\top (x+v) < c^\top x$. Afin de résoudre (2.76), il suffit de répéter cette opération jusqu'au moment où il n'est plus possible d'identifier un vecteur $v \in B$ permettant d'améliorer la solution considérée.

En 1975, Graver [28] introduit la notion d'ensemble test associé à la famille de problèmes $IP(b, c)$, lorsque $u = (\infty)^n$. Afin de bien présenter la définition d'un ensemble test de Graver, il faut tout d'abord définir ce qu'est une base de Hilbert associée à un cône polyhédral. Un cône est polyédral lorsqu'il est possible de le générer de façon finie, c'est-à-dire, soit C un cône, celui-ci est polyédral si $\exists V = \{v^1, \ldots, v^k\} \subseteq \mathbb{R}^n$ tel que

$C = \{y : \ y = \sum_{i=1}^{k} \lambda_i v^i, \ \lambda_1, \ldots, \lambda_k \geq 0\}$. Un cône polyédral C est rationnel s'il existe une matrice $A \in \mathbb{Q}^{m \times n}$ telle que $C = \{x \in \mathbb{R}^n : \ Ax = 0\}$. À présent, une base de Hilbert définie sur un cône rationnel C est un ensemble fini $H = \{h^1, \ldots, h^t\} \subseteq C \cap \mathbb{Z}^n$ tel que tout élément $z \in C \cap \mathbb{Z}^n$ s'écrit comme comme étant : $z = \sum_{i=1}^{t} \lambda_i h^i$ pour des valeurs $\lambda_1, \ldots, \lambda_t \geq 0$. Par conséquent, il est possible d'exprimer tous les vecteurs entiers du cône C par une combinaison linéaire non-négative des éléments contenus dans la base de Hilbert qui lui est associée.

Weismantel [68] rapporte que tout cône rationnel possède une base de Hilbert. De plus, si ce cône est pointé, alors il n'existe qu'une seule de ces bases qui est minimale pour ce qui est de l'inclusion. Graver [28] démontre que : en définissant O_j comme étant le $j^{\text{ème}}$ orthant de $\mathbb{R}^n$, et en posant $C_j = O_j \cap \{x \in \mathbb{R}^n : \ Ax = 0\}$, de même que H_j comme étant la base de Hilbert minimale (en ce qui concerne l'inclusion) associée à C_j, alors $H = \bigcup_j H_j \backslash \{0\}$ constitue un ensemble test fini associé à la famille de problèmes $IP(b, c)$. L'ensemble H contient donc un ensemble test pour chacun des problèmes inclus dans $IP(b, c)$. Encore une fois, lorsque H est connu, il est possible de résoudre tout problème de $IP(b, c)$ par l'approche d'amélioration itérative présentée précédemment.

Une autre façon de générer un ensemble test associé à une famille de problèmes en nombres entiers est d'utiliser les bases de Gröbner. Une base de Gröbner est définie sur l'ensemble $IP(b)$, où l'on suppose que la matrice A est de plein rang et où $u = (\infty)^n$. Supposons également que les problèmes de type (2.76) pour tout $b \in \mathbb{Z}^m$ sont bornés. Pour toute solution $x \in \mathbb{N}^n$, nous définissons $b(x)$ comme étant le vecteur unique tel que $Ax = b(x)$. Nous définissons également $y(x) \in \mathbb{N}^n$ comme étant la solution optimale au problème (2.76) lorsque $b = b(x)$, de même que l'ensemble $S = \{x \in \mathbb{N}^n : \ x \neq y(x)\}$. L'ensemble S comprend toutes les solutions réalisables et non réalisables associées aux problèmes de $IP(b)$ à l'exception de $y(x)$. Weismantel démontre qu'il existe un ensemble fini de vecteurs $x^1, \ldots, x^t \in \mathbb{N}^n$ tel que $S = \bigcup_{i=1}^{t} (x^i + \mathbb{N}^n)$ où $x^i + \mathbb{N}^n = \{w \in \mathbb{N}^n : \ w \geq x^i\}$.

Si l'on pose pour $i \in \{1, \ldots, t\}$ le vecteur $y^i \in \mathbb{N}^n$ comme étant la solution optimale à (2.76) lorsque $b = b(x^i)$, alors l'ensemble $G_c = \{y^i - x^i : \ i = 1, \ldots, t\}$ forme la

base de Gröbner réduite associée à $IP(b)$. La base de Gröbner réduite comprend un ensemble test pour chacun des problèmes inclus dans $IP(b)$. De façon à bien comprendre ce résultat, nous allons voir comment il est possible de trouver, pour toute solution réalisable non-optimale à un problème inclus dans $IP(b)$, un élément de G_c représentant un déplacement réalisable de descente. Supposons que : $x \in \mathbb{N}^n$ constitue une solution réalisable mais non-optimale pour le problème (2.76) défini pour un certain vecteur $b \in \mathbb{Z}^m$. Étant donné le résultat démontré par Weismantel, il existe un indice $i \in \{1, \ldots, t\}$ et un vecteur $v \in \mathbb{N}^n$ tels que $x = x^i + v$. En définissant $x' = x + (y^i - x^i)$, on s'aperçoit que x' constitue une solution réalisable au problème considéré puisque $Ay^i = Ax^i = b(x^i)$. Étant donné que $c^\top y^i < c^\top x^i$, alors la nouvelle solution x' est telle que $c^\top x' < c^\top x$. Le vecteur $y^i - x^i \in G_c$ constitue donc un déplacement réalisable de descente pour la solution x. Pour terminer, Weismantel démontre que la base de Gröbner universelle associée à la matrice A, qui est définie comme étant : $G = \bigcup_{c \in \mathbb{Z}^n} G_c$, est un sous-ensemble de l'ensemble test de Graver. Par conséquent, les bases de Gröbner sont des ensembles finis.

L'utilité des ensembles test dans le cas de la programmation stochastique en nombres entiers s'explique par le fait que ces ensembles induisent une approche rapide de résolution pour des familles de problèmes. En supposant que les problèmes de deuxième étape s'énoncent de la façon suivante : $Min_y\{q(\omega)^\top y \mid Wy = h(\omega) - T(\omega)x,\ y \in \mathbb{Z}_+^{n_2}\}$ (où $W \in \mathbb{Z}^{m_2 \times n_2}$), à partir de l'ensemble test de Graver qui leur est associé, il est possible de résoudre par l'approche d'amélioration itérative tout problème inclus dans $IP(q(\omega), h(\omega) - T(\omega)x)$ (où $q(\omega) \in \mathbb{Z}^{n_2}$ et $h(\omega) - T(\omega)x \in \mathbb{Z}^{m_2}$). Ceci peut s'avérer fort intéressant si l'événement aléatoire implique un très grand nombre de réalisations. De façon similaire, il est possible d'utiliser la base de Gröbner réduite lorsque l'événement aléatoire n'implique que le terme de droite des contraintes des problèmes de deuxième étape.

Le premier article qui propose d'utiliser les ensembles test dans le cas de la programmation stochastique est celui de Schultz et *al.* [55]. Dans le cadre de cet article, les auteurs cherchent à résoudre des problèmes où les variables de première étape sont continues, celles de deuxième étape sont entières et où l'événement aléatoire comprend

uniquement le vecteur $h(\omega)$. Dans ce cas, les problèmes de deuxième étape s'énoncent de la façon suivante : $Q(x,\xi_k) = Min_y\{q^\top y \mid Wy \geq h_k - Tx,\ y \in \mathbb{Z}_+^{n_2}\}$, où $q \in \mathbb{Z}^{n_2}$ et $W \in \mathbb{Z}^{m_2 \times n_2}$, pour $k = 1,\ldots,K$. Afin de résoudre efficacement ces problèmes, les auteurs proposent de trouver la base de Gröbner réduite (soit G_q) qui leur est associée. L'ensemble G_q est construit en utilisant l'algorithme de Buchberger tel que présenté par Thomas [62]. De façon à résoudre le problème stochastique, une procédure de recherche exhaustive, équivalente à celle présentée par Ahmed et *al.* [2], est premièrement utilisée afin d'énumérer une série de solutions réalisables de première étape à partir desquelles il est possible de créer une partition du domaine réalisable. La base de Gröbner réduite G_q est par la suite utilisée afin d'évaluer la fonction de recours pour chacune des solutions de première étape identifiées.

Hemmecke et Schultz [30] ont également proposé une approche de résolution basée sur les ensembles test pour les problèmes de programmation stochastique en nombres entiers. Les auteurs considèrent le cas où les variables de première et deuxième étapes sont entières. Encore une fois, l'événement aléatoire n'implique que le vecteur $h(\omega)$. Le problème à résoudre s'énonce donc de la façon suivante :

$$\begin{aligned} Min \quad & c^\top x + \sum_{k=1}^{K} p_k q^\top y_k \\ s.l.c. \quad & Ax = b \\ & Tx + Wy_k = h_k, \quad k = 1,\ldots,K \\ & x \in \mathbb{Z}_+^{n_1},\ y_k \in \mathbb{Z}_+^{n_2},\ \ k = 1,\ldots,K. \end{aligned}$$

En posant $h^\top = (c, p_1 q, \ldots, p_K q)^\top$, $d^\top = (b, h_{1,\ldots,h_K})^\top$, $z^\top = (x, y_1, \ldots, y_K)^\top$ et la matrice A_K comme suit :

$$A_K = \begin{pmatrix} A & 0 & 0 & \cdots & 0 \\ T & W & 0 & \cdots & 0 \\ T & 0 & W & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ T & 0 & 0 & \cdots & W \end{pmatrix}$$

il est possible de réexprimer le problème précédent comme étant :

$$Min\ \{h^{\top}z:\ A_K z = d,\ z \in \mathbb{Z}_+^N\} \text{ où } N = n_1 + Kn_2. \tag{2.77}$$

Hemmecke et Schultz proposent de construire l'ensemble test de Graver (soit H) associé aux problèmes du type (2.77), afin de résoudre le problème stochastique en nombres entiers. Étant donné la taille de la matrice A_K, il peut être difficile de calculer directement H. Afin de contourner ce problème, les auteurs démontrent qu'étant donné la structure en blocs de A_K, il est possible de construire les vecteurs compris dans H à partir d'un ensemble plus restreint d'éléments. Un algorithme fini permettant de trouver ces éléments est présenté en [30]. Une fois ces éléments obtenus, l'approche d'amélioration itérative définie précédemment peut alors être utilisée afin de résoudre le problème stochastique.

Chapitre 3

Accelerating Benders Decomposition by Local Branching

Tel qu'indiqué précédemment, afin de résoudre des problèmes de type (1.1)-(1.5), plusieurs des algorithmes de génération de coupes qui ont été développés utilisent la décomposition de Benders [4] afin de contourner la complexité associée à la fonction de recours. Considérons le cas où les variables de première étape sont binaires et les variables de deuxième étape sont continues. En appliquant la décomposition de Benders, le problème original stochastique (1.1)-(1.5) est séparé en deux, en fonction des étapes de décision du modèle. Nous obtenons ainsi un problème maître impliquant les variables de première étape x, où $\mathcal{Q}(x)$ est approximé à l'aide de coupes d'optimalité et où les coupes de réalisabilité sont utilisées de façon à obtenir des solutions réalisables en deuxième étape ($x \in \{x \mid \mathcal{Q}(x) < +\infty\}$). Les problèmes de deuxième étape sont utilisés, quant à eux, afin d'obtenir l'information nécessaire à la génération des coupes d'optimalité et de réalisabilité. Bien que la décomposition de Benders permette de traiter la complexité stochastique associée à (1.1)-(1.5), il n'en demeure pas moins que, d'un point de vue algorithmique, cette approche comporte certaines faiblesses.

Notons premièrement que les problèmes maîtres relaxés impliquent des variables entières. Par conséquent, selon l'approche classique de Benders, à chaque fois qu'une coupe est générée, un problème en nombres entiers doit être résolu. Ces problèmes deviennent également plus difficiles à résoudre au fur et à mesure que des coupes y sont ajoutées. À présent, puisqu'à chaque itération une nouvelle coupe est ajoutée au problème maître relaxé, nous constatons que la borne inférieure, qui est définie comme étant la valeur de la solution au problème maître relaxé, est non-décroissante. Par contre, aucune garantie n'existe pour ce qui est de la borne supérieure. La borne supérieure est définie comme étant la valeur de la meilleure solution réalisable trouvée par l'algorithme. La qualité de cette borne dépend donc de l'ordre dans lequel les solutions réalisables sont obtenues. Si une bonne solution est obtenue dès le départ, le processus de résolution sera plus rapide. En revanche, étant donné que le problème maître relaxé ne fournit qu'une description partielle du problème original, il arrive fréquemment que les bonnes solutions ne soient obtenues que lorsqu'un nombre suffisant de coupes ont été ajoutées. Si l'algorithme ne peut trouver une bonne solution réalisable rapidement, le processus de résolution sera plus long. Il est important de noter que ces faiblesses découlent directement de l'approche générale de Benders et ne sont pas spécifiques au contexte stochastique. Par conséquent, cet article propose une amélioration de l'algorithme général de Benders, fondée sur l'utilisation de la séparation locale [20].

Ce qui est proposé est d'utiliser les solutions obtenues pour les problèmes maîtres relaxés comme points de départ pour effectuer des phases de recherche de type séparation locale. Ce faisant, il est possible d'améliorer simultanément les bornes inférieure et supérieure générées par l'algorithme de Benders. Ces phases de recherche permettent de trouver de nouvelles solutions réalisables en explorant des voisinages différents autour de la solution du problème maître relaxé. Il est donc possible d'obtenir plus rapidement de meilleures solutions réalisables au problème original. La séparation locale permet également d'identifier rapidement des solutions réalisables différentes. Ce faisant, puisqu'il est possible de générer une coupe d'optimalité pour chacune des solutions réalisables au problème original, la séparation locale permet également de trou-

ver de nouvelles coupes à ajouter au problème maître relaxé. Finalement, l'exploration des voisinages définis par la séparation locale permet, lorsque certaines conditions sont vérifiées, de générer des inégalités valides délimitant les régions réalisables du problème. Ces inégalités peuvent être utilisées afin de remplacer ou complémenter les coupes de réalisabilité trouvées par l'algorithme. Ces améliorations sont applicables à tout contexte où la décomposition de Benders est utilisée. Des tests numériques sont effectués sur une famille générale de problèmes de conception de réseaux. Ces résultats permettent d'illustrer les avantages de cette nouvelle approche.

Accelerating Benders Decomposition by Local Branching

Walter Rei
Université de Montréal

Jean-François Cordeau
HEC Montréal

Michel Gendreau
Université de Montréal

Patrick Soriano
HEC Montréal

December 2005

Abstract

This paper explains how local branching can be used to accelerate the classical Benders decomposition algorithm. By applying local branching throughout the solution process, one can simultaneously improve both the lower and upper bounds. We also show how Benders feasibility cuts can be strenghtened or replaced with local branching constraints. To assess the performance of the different algorithmic ideas, computational experiments were performed on a series of network design problems and results illustrate the benefits of this approach.

Keywords. Benders decomposition, local branching, network design.

3.1 Introduction

Benders decomposition [4] was introduced in the early sixties as a solution strategy for mixed-integer problems. As shown by Geoffrion [26, 27], this method can be classified as a pattern of projection, outer linearization and relaxation. Let us consider the following general class of mixed 0-1 linear programming problems :

$$\text{Min} \quad c_1^\top x + c_2^\top y \tag{3.1}$$

$$\text{s.t.} \quad A_1 x = b_1 \tag{3.2}$$

$$A_2 x + E y = b_2 \tag{3.3}$$

$$x \in \{0,1\}^{n_1} \tag{3.4}$$

$$y \in \mathbb{R}_+^{n_2}, \tag{3.5}$$

where $c_1 \in \mathbb{R}^{n_1}$, $c_2 \in \mathbb{R}^{n_2}$, $b_1 \in \mathbb{R}^{m_1}$, $b_2 \in \mathbb{R}^{m_2}$, $A_i \in \mathbb{R}^{n_1 \times m_i}$ $(i = 1, 2)$ and $E \in \mathbb{R}^{n_2 \times m_2}$. It should be noted that all ideas proposed in this paper can be easily generalized to the case where a subset of the x variables are general integers. However, in order to keep the presentation simple, only problem (3.1)-(3.5) will be used.

By projecting problem (3.1)-(3.5) onto the space defined by the binary variables x, we obtain the following problem :

$$\begin{aligned} \text{Min} \quad & c_1^\top x + \inf_y \{c_2^\top y \mid Ey = b_2 - A_2 x,\ y \in \mathbb{R}_+^{n_2}\} \\ \text{s.t.} \quad & A_1 x = b_1 \\ & x \in \{0,1\}^{n_1}. \end{aligned}$$

Define $\Pi = \{\pi \in \mathbb{R}^{m_2} \mid E^\top \pi \leq c_2\}$ and let Λ be the set of extreme points associated with Π, and Φ be the set of extreme rays of cone $C = \{\pi \in \mathbb{R}^{m_2} \mid E^\top \pi \leq 0\}$. Applying an outer linearization to function $\inf_y \{c_2^\top y \mid Ey = b_2 - A_2 x,\ y \in \mathbb{R}_+^{n_2}\}$, we can restate

the original problem as the following equivalent master problem :

$$\text{Min} \quad c_1^\top x + \Theta \tag{3.6}$$

$$\text{s.t.} \quad A_1 x = b_1 \tag{3.7}$$

$$(b_2 - A_2 x)^\top \lambda_i \leq \Theta, \ \forall \lambda_i \in \Lambda \tag{3.8}$$

$$(b_2 - A_2 x)^\top \phi_j \leq 0, \ \forall \phi_j \in \Phi \tag{3.9}$$

$$x \in \{0,1\}^{n_1}. \tag{3.10}$$

Constraints (3.8) are called optimality cuts since they define the objective value associated with feasible values of x. Constraints (3.9) are called feasibility cuts since they eliminate values of x for which function $\inf_y \{c_2^\top y \mid Ey = b_2 - A_2 x, \ y \in \mathbb{R}_+^{n_2}\} = +\infty$.

Since the number of constraints in sets (3.8) and (3.9) may be very large, Benders proposed a relaxation algorithm in order to solve problem (3.6)-(3.10). This algorithm can be stated as follows :

Relaxation Algorithm

Step 0 (Initialization)

$\nu = 0$, $t = 0$, $s = 0$,

$\underline{z} = -\infty$,

$\overline{z} = +\infty$.

Step 1 (Solve the relaxed master problem)

$\nu = \nu + 1$

Solve

$$\text{Min} \quad c_1^\top x + \Theta \tag{3.11}$$

$$\text{s.t.} \quad A_1 x = b_1 \tag{3.12}$$

$$(b_2 - A_2 x)^\top \lambda_i \leq \Theta, \ i = 1, \ldots, t \tag{3.13}$$

$$(b_2 - A_2 x)^\top \phi_j \leq 0, \ j = 1, \ldots, s \tag{3.14}$$

$$x \in \{0,1\}^{n_1}. \tag{3.15}$$

Let (x^ν, Θ^ν) be an optimal solution to problem (3.11)-(3.15).

Set $\underline{z} = c_1^\top x^\nu + \Theta^\nu$.

Step 2 (Solve the subproblem)

Solve

$$v^\star = \text{Max } (b_2 - A_2 x^\nu)^\top \pi \tag{3.16}$$

$$\text{s.t. } E^\top \pi \leq c_2, \tag{3.17}$$

$$\pi \in \mathbb{R}^{m_2}. \tag{3.18}$$

If (3.16)-(3.18) is feasible :

- Set $\overline{z} = \min\{\overline{z},\ c_1^\top x^\nu + v^\star\}$,

- **If** $\frac{\overline{z}-\underline{z}}{\overline{z}} < \epsilon$, **Stop.**

- One obtains an extreme point λ_{t+1} violating one of the inequalities (3.8) which can be added to (3.11)-(3.15),

- set $t = t + 1$ and go to Step 1.

Else :

- One obtains an extreme ray ϕ_{s+1} violating one of the inequalities (3.9) which can be added to (3.11)-(3.15),

- set $s = s + 1$ and go to Step 1.

There are certain observations to be made when analyzing the Benders decomposition approach. One must first consider the fact that each time Step 1 of the relaxation algorithm is performed, an integer problem must be solved. Furthermore, this process becomes more difficult each time a new cut is added. Another important observation concerns the bounds generated by the algorithm. Since each iteration of the algorithm adds a new cut to the relaxed master problem, the lower bound $\underline{z}$ is therefore non-decreasing. However, there is no guarantee that the upper bound $\overline{z}$ is decreasing. On many instances, the evolution of the objective function value associated with the feasible solutions obtained is quite erratic.

In order to circumvent these difficulties, we propose to use phases of local branching (Fischetti and Lodi [20]) throughout the solution process. The aim of the local branching phase is to find better upper bounds as well as multiple optimality cuts at each iteration

of the relaxation algorithm. By working simultaneously on improving the lower bound $\underline{z}$ and the upper bound $\overline{z}$, one can hope to reduce the number of relaxed master problems that need to be solved.

The remainder of this article is organized as follows. Section 2 contains a brief review of the different techniques that have been proposed in order to improve the classical Benders decomposition approach. In Section 3 we describe the local branching strategy and how this strategy can be used to improve Benders decomposition. This is followed by computational experiments in Section 4. Finally we conclude in Section 5 with a discussion of the performance of our approach as well as ideas for future research.

3.2 Related Work

Over the years a series of techniques have been proposed to speed-up the classical Benders decomposition approach. Research has mainly focused on either reducing the number of integer relaxed master problems being solved or on accelerating the solution of the relaxed master problem. In 1977, McDaniel and Devine [47] proposed to generate cuts from solutions obtained by solving the linear relaxation of (3.11)-(3.15). Since any extreme point (or extreme ray) of the dual subproblem generates a valid optimality (or feasibility) cut for the integer master problem, one can hope to generate useful information for the integer case by adding the cuts derived from the continuous relaxation, therefore reducing the number of integer master problems that need to be solved. In 1984, Côté and Laughton [14] also observed that one does not have to solve the relaxed master problem to optimality in Step 1 of the relaxation algorithm. One only needs to find an integer solution in order to generate an optimality (or feasibility) cut. Therefore, any heuristic that is adapted for the specific problem to be solved can be used to generate different cuts. The main drawback of such a strategy is that by generating only the cuts associated with the solutions obtained using the heuristic, one can fail to generate cuts that are necessary to ensure convergence.

In 1981, Magnanti and Wong [45] proposed to accelerate the convergence of the

Benders algorithm by adding Pareto-optimal cuts. A Pareto-optimal cut is defined as follows : consider problem (3.1)-(3.5) and let $X = \{x \mid A_1x = b_1,\ x \in \{0,1\}^{n_1}\}$. A cut $(b_2 - A_2x)^\top\lambda_1 \leq \Theta$ is said to dominate another cut $(b_2 - A_2x)^\top\lambda_2 \leq \Theta$ if $(b_2 - A_2x)^\top\lambda_1 \geq (b_2 - A_2x)^\top\lambda_2$ for all $x \in X$ with a strict inequality for at least one point in X ; a cut is said to be Pareto-optimal if no other cut dominates it. Let us now consider the case where there are multiple optimal solutions to the subproblem in Step 2. In such a case, Magnanti and Wong demonstrated that one can obtain a Pareto-optimal cut by evaluating for each of the dual solutions the associated cut at a core point of set X and then choosing the one that gives the maximum value.

The addition of non-dominated optimality cuts to the relaxed master problems can greatly improve the quality of the lower bound obtained in the Benders relaxation algorithm. In turn, this will accelerate the convergence by reducing the total number of iterations needed to obtain an optimal solution.

In 1983, Van Roy [65] proposed a new type of decomposition (cross decomposition) in order to solve hard integer problems. Cross decomposition can also be seen as a way to improve the classical Benders decomposition approach. The main idea proposed by Van Roy was to use simultaneously primal decomposition (Benders decomposition) and dual decomposition (Lagrangean relaxation). The author showed that a solution to the Lagrangean subproblem can act as a possible solution to the Benders master problem and vice versa : a solution to the Benders subproblem can act as a possible solution to the Lagrangean problem. Therefore, one can alternately solve the two subproblems in order to generate useful information for the master problems. Unfortunately, in order to maintain convergence, one still has to solve at times the Benders master problem. However, by reducing the number of such resolutions and replacing them by calls to the Lagrangean subproblem, cross decomposition can serve as a feature to speed-up the Benders decomposition algorithm.

As for techniques to reduce the solution time of the relaxed master problem, research has been mostly oriented toward the use of Lagrangean relaxation. Côté and Laughton [14] made an observation concerning the case where set $X = \{x \mid A_1x =$

$b_1,\ x \in \{0,1\}^{n_1}\}$ has a special structure wich can be exploited by solution algorithms. In this case, constraints (3.13) and (3.14) would prevent the use of specifically adapted methods for solving model (3.11)-(3.15). By using Lagrangean relaxation on these constraints, one can regain the special structure of the problem. For specific values of the Lagrange multipliers, the problem can be solved efficiently with a method that exploits the structure of X. However, the integer solution obtained may not be feasible in model (3.11)-(3.15). The authors propose to use subgradient optimization in order to modify the Lagrange multipliers and resolve the problem. Unfortunately, at the end of this process one may still not find an optimal solution to the master problem due to the duality gap that may exist. For such a case, a branching strategy must be used in the master problem to bridge the gap.

Lagrangean relaxation may appear as a good way to circumvent the difficulty of solving the master problems. However, Holmberg [34] made an in-depth analysis on the use of such a technique. This author showed that the lower bound defined by the use of Lagrangean relaxation on the master problem cannot be better than the bound obtained from the Lagrangean dual of the original problem (the dual subproblem used in cross decomposition). This is always the case, even when all cuts are added to the Benders master problem. Therefore, one can never hope to obtain better bounds by applying Lagrangean relaxation on the master problem (even when a large number of cuts have been added) compared to using Lagrangean relaxation on the original problem. Also, when using Lagrangean relaxation on the master problem, a lack of controllability on the integer solution obtained may prevent the approach from converging (i.e., generating necessary cuts).

Recently, Codato and Fischetti [12] proposed to use combinatorial inequalities in the case of Benders Decomposition. When considering problem (3.1)-(3.5), for the cases where $c_1 \neq [0]^\top$ and $c_2 = [0]^\top$ or $c_1 = [0]^\top$ and $c_2 \neq [0]^\top$, the purpose of subproblem (3.16)-(3.18) can be reduced to testing the feasibility of the integer solutions and only feasibility cuts are added to the master problems. For these cases, using the principles evoked by Hooker [35], the authors make the observation that whenever the solution to the master problem x^ν is infeasible, one may find at least one $C \subseteq \{1,\ldots,n_1\}$ such that

x_j^ν, $j \in C$, is a minimal (or irreducible) infeasible subsystem of (3.16)-(3.18). Therefore, one needs to change at least one binary value of x_j^ν, $j \in C$, in order to eliminate the infeasibility associated with C. In order to impose this condition, Codato and Fichetti define the following combinatorial Benders cut : $\sum_{i \in C: x^\nu_{j(i)}=1} (1 - x_j) + \sum_{i \in C: x^\nu_{j(i)}=0} x_j \geq 1$. In [12], the authors present a separation algorithm in order to find minimal infeasible subsystems. Combinatorial Benders cuts are then added to the master problems in a general branch and cut framework. Computational experiments were performed on two classes of mixed-integer problems. Results seem to demonstrate that by adding combinatorial Benders cuts, one can considerably improve the quality of the bounds obtained for the LP relaxation of the master problems.

3.3 Local Branching in Benders Decomposition

We begin this section by giving a brief description of the local branching strategy [20]. The main idea behind local branching is to divide the feasible region of a problem into smaller subregions and then use a generic solver (e.g., CPLEX) to find the best solution (or at least a good feasible solution) in each of the subregions. By doing so, one can take advantage of the fact that generic optimizers can efficiently solve small instances of a problem and then use this characteristic in a general branching strategy for solving large-scale problems.

Let us consider problem (3.1)-(3.5) and (x^0, y^0) a feasible solution to (3.1)-(3.5). By using the Hamming distance function $\Delta(x, x^0) = \sum_{j \in S_0} (1 - x_j) + \sum_{j \in N_1 \setminus S_0} x_j$ (where $N_1 = \{1, \ldots, n_1\}$ and $S_0 = \{j \in N_1 \mid x_j^0 = 1\}$), one can divide the feasible region of problem (3.1)-(3.5) by considering on one hand the values of x for which $\Delta(x, x^0) \leq \kappa$ and on the other hand the values of x for which $\Delta(x, x^0) \geq \kappa + 1$ (where κ is a positive integer). By imposing an adequate value κ, one can solve efficiently problem (3.1)-(3.5) in the subregion defined by $\Delta(x, x^0) \leq \kappa$ using the generic optimizer. Subregion $\Delta(x, x^0) \geq \kappa + 1$ is left for further exploration.

Let us now consider the local branching algorithm at iteration t, where (x^t, y^t) is

the current feasible solution to (3.1)-(3.5). Let x^j, $j \in J^t$, be a series of values such that $\inf_y \{c_2^\top y \mid Ey = b_2 - A_2 x^j,\ y \in \mathbb{R}_+^{n_2}\} \neq +\infty$ (i.e., previous feasible solutions). Let us also define a set I^t, such that $I^t \subseteq J^t$. By using function $\Delta(x, x^t)$ as well as sets J^t and I^t, one can divide the unexplored feasible region of problem (3.1)-(3.5) into the following two subproblems (thus creating a left and right branch) :

$$
\begin{array}{llll}
(P_t) & \text{Min} & c_1^\top x + c_2^\top y & \\
& \text{s.t} & A_1 x = b_1 & \\
& & A_2 x + Ey = b_2 & \\
& & \Delta(x, x^j) \geq 1,\ j \in J^t & \\
& & \Delta(x, x^i) \geq \kappa_i,\ i \in I^t & \\
& & \Delta(x, x^t) \leq \kappa & \\
& & x \in \{0,1\}^{n_1} & \\
& & y \in \mathbb{R}_+^{n_2}. &
\end{array}
\qquad
\begin{array}{llll}
(\overline{P}_t) & \text{Min} & c_1^\top x + c_2^\top y & \\
& \text{s.t} & A_1 x = b_1 & \\
& & A_2 x + Ey = b_2 & \\
& & \Delta(x, x^j) \geq 1,\ j \in J^t & \\
& & \Delta(x, x^i) \geq \kappa_i,\ i \in I^t & \\
& & \Delta(x, x^t) \geq \kappa + 1 & \\
& & x \in \{0,1\}^{n_1} & \\
& & y \in \mathbb{R}_+^{n_2}. &
\end{array}
$$

The feasible region of P_t is limited to all feasible values of (x, y) such that $x \in \{x \in \{0,1\}^{n_1} \mid \Delta(x, x^j) \geq 1,\ j \in J^t,\ \Delta(x, x^i) \geq \kappa_i,\ i \in I^t\}$ and for which the Hamming distance between x and x^t is less than or equal to κ.

Subproblem P_t is solved using the generic optimizer. Let (x^{t+1}, y^{t+1}) be the optimal solution to P_t. If $c_1^\top x^{t+1} + c_2^\top y^{t+1} < c_1^\top x^t + c_2^\top y^t$ then one can set $\kappa_t = \kappa + 1$, replace constraint $\Delta(x, x^t) \leq \kappa$ with $\Delta(x, x^t) \geq \kappa_t$, set $I^{t+1} = I^t \cup \{t\}$ (which gives us subproblem $\overline{P}_t$), and then divide the feasible region of $\overline{P}_t$ as before, using function $\Delta(x, x^{t+1})$ (creating subproblems P_{t+1} and $\overline{P}_{t+1}$). One can then proceed by solving the new left branch created (subproblem P_{t+1}). However, if $c_1^\top x^{t+1} + c_2^\top y^{t+1} \geq c_1^\top x^t + c_2^\top y^t$ or P_t is infeasible, a diversification procedure is applied. The diversification procedure consists of replacing $\Delta(x, x^t) \leq \kappa$ with $\Delta(x, x^t) \leq \kappa + 1$ in subproblem P_t and then adding constraint $\Delta(x, x^t) \geq 1$ (with $J^{t+1} = J^t \cup \{t\}$). Therefore, one increases the size of the feasible region of P_t and, by adding $\Delta(x, x^t) \geq 1$, eliminates solution (x^t, y^t). Subproblem P_t is solved anew, and the procedure continues by testing the new solution obtained. For the diversification procedure, it should be noted that Fischetti and Lodi [20] proposed to replace $\Delta(x, x^t) \leq \kappa$ with $\Delta(x, x^t) \leq \kappa + \lceil \frac{\kappa}{2} \rceil$. However, for the purpose of this paper, in an effort to limit the increase in size of the active subproblem,

an augmentation of one is used.

An important observation to be made concerns the solution of the active subproblem. Since the size of the neighborhood defined by the Hamming distance may become large, it may be impossible to solve the subproblem to optimality. Fischetti and Lodi [20] proposed a series of techniques to circumvent this difficulty. These techniques include imposing a time limit on the solution of the active subproblem as well as a series of diversification mechanisms derived from local search metaheuristics. A complete description of these techniques is provided in [20]. For the purpose of this paper, it should be specified that every active subproblem will be solved until an $\bar{\epsilon} - opt$ solution is found or a predetermined time limit is reached.

Let us now examine the usefulness of local branching in the context of Benders decomposition. By fixing a time limit and a starting feasible solution (x^0, y^0), the local branching algorithm searches for a better upper bound in a neighbourhood around (x^0, y^0). Let $P_0, P_1, \ldots, P_L$ be the series of subproblems solved by the algorithm. Without any loss of generality, let us suppose that the first L' $(L' \leq L)$ of these subproblems are feasible and let (x^1, y^1), (x^2, y^2), $\ldots$, $(x^{L'}, y^{L'})$ be the solutions obtained for each of these subproblems. Since the algorithm adds a constraint $\Delta(x, x^t) \geq 1$ each time the value of the solution obtained is not better than the previous one, then at least $\left\lceil \frac{L'+1}{2} \right\rceil$ of the solutions (x^0, y^0), (x^1, y^1), $\ldots$, $(x^{L'}, y^{L'})$ will be different.

In the context of Benders decomposition, let us now suppose that at iteration ν of the relaxation algorithm, one uses the solution to the relaxed master problem x^ν as a starting point for a phase of local branching. It is important to observe that x^ν does not have to be a feasible solution (i.e., induce a feasible subproblem (3.16)-(3.18)) in order to be used as a starting point for the local branching algorithm. One can start from an infeasible point and search its neighbourhood for feasible solutions. At the end of the local branching phase, one obtains an upper bound $\hat{z} = \min_{l=1,\ldots,L'}\{c_1^\top x^l + c_2^\top y^l\}$ on (3.1)-(3.5) and each feasible solution identified can be used in order to generate an optimality cut. Therefore, with at least $\left\lceil \frac{L'}{2} \right\rceil$ (or $\left\lceil \frac{L'+1}{2} \right\rceil$ if x^ν is feasible) different solutions one can create a pool of possible cuts to be added to the relaxed master

problem (3.11)-(3.15). At iteration ν, solution x^ν provides a point in the set X that must be considered in the Benders solution process. By searching in the neighbourhood of this point, one may hope to find useful information that will help the Benders algorithm in eliminating a larger subregion of X.

Another point to be made concerns the branching strategy used in the local branching algorithm as well as the size of the subregions explored (parameter κ). In the classical local branching algorithm the branching decision is applied whenever the algorithm finds a better feasible solution. However, one should keep in mind the two purposes of the local branching search in the context of Benders decomposition : finding better upper bounds and generating different cuts in order to obtain better lower bounds. Local branching offers a general framework that makes it easy to pursue both objectives. By keeping a relatively low parameter κ and by applying the branching decision often, one is sure to explore rapidly different parts of the feasible region. In this case, the emphasis is placed on finding different feasible solutions (i.e., optimality cuts). Since the main difficulty related to problems of type (3.1)-(3.5) concerns the quality of the lower bounds obtained, by generating a wide pool of optimality cuts at each iteration of the relaxation algorithm, one can expect to accelerate the search process.

Not every cut identified after a local branching phase may be worth adding to problem (3.11)-(3.15). One must choose cuts that permit what Holmberg [33] defined as cut-improvement. At iteration ν of the relaxation algorithm, an optimality cut that gives cut-improvement is a new cut (i.e., a cut not yet present in (3.11)-(3.15)) that may be active in an optimal solution of problem (3.1)-(3.5). In cross decomposition Van Roy [65] uses the solution to the dual suproblem in order to generate a cut in the Benders relaxed master problem. Since there is no guarantee that this cut will be useful, the author uses this next result in order to verify cut-improvement. Let the relaxed master

problem at iteration ν be :

$$\begin{aligned} Min \quad & c_1^\top x + \Theta \\ s.t \quad & A_1 x = b_1 \\ & (b_2 - A_2 x)^\top \lambda_i \leq \Theta, \ i = 1, \ldots, t \\ & (b_2 - A_2 x)^\top \phi_j \leq 0, \ j = 1, \ldots, s \\ & x \in \{0,1\}^{n_1}. \end{aligned}$$

Let $\tilde{x}$ be a feasible solution and $\overline{z}$ the value of the best feasible solution used in order to generate an optimality cut in the previous problem. If $c_1^\top \tilde{x} + (b_2 - A_2 \tilde{x})^\top \lambda_i < \overline{z}$, $i = 1, \ldots, t$, then $\tilde{x}$ gives cut-improvement in the relaxed master problem.

This result may also be used to identify which of the different feasible solutions obtained after the local branching phase allow cut-improvement. However, since each added cut will make the relaxed master problem harder to solve, one may want to add only the deepest cut available (i.e., the cut that will eliminate the biggest part of the feasible region of the relaxed master problem). Defining a measure for the deepness of a cut is not easily done. Similarly to Magnanti and Wong [45], what we propose is to use a core point in order to simulate the direction to the optimal solution of problem (3.1)-(3.5). By identifying a point $\hat{x}^0$ in the interior of the feasible region of (3.1)-(3.5) and then evaluating the cuts at $\hat{x}^0$, one can choose the cut that will eliminate the most solutions in that direction. Using this criterion, the deepest cut is

$$\lambda \in \text{argmax}\{(b_2 - A_2 \hat{x}^0)^\top \lambda_l \mid l = 0, \ldots, L'\}, \tag{3.19}$$

where λ_l, $l = 0, \ldots, L'$, are the optimal solutions to the dual of subproblem (3.16)-(3.18) when $x = x^l$. The effectiveness of this measure will depend on the core point that is chosen. Therefore, one can either fix this point at the beginning or adjust it iteratively throughout the solution process.

There is another important point to be made when using local branching in a Benders decomposition process. This point concerns the subproblems that are found to be infeasible. In the classical Benders decomposition approach, at iteration ν, when the solution to the relaxed master problem (x^ν) is infeasible, a feasibility cut is added to

(3.11)-(3.15) using ϕ_{s+1}. This cut will eliminate from further consideration x^ν as well as all values of $x \in X$ for which $(b_2 - A_2 x)^\top \phi_{s+1} > 0$. When using local branching, one will sometimes find subproblems that are infeasible. Suppose that one executes a local branching phase around x^ν. Let us consider subproblem P_l which is defined by the local branching constraints $\Delta(x, x^j) \geq 1,\ j \in J^l$, $\Delta(x, x^i) \geq \kappa_i,\ i \in I^l$ and $\Delta(x, x^l) \leq \kappa$. If P_l is found to be infeasible, then one has found in the neighbourhood of x^ν a subregion which contains no feasible solution. Since the feasible solutions $x^j, j \in J^l$, have already been considered by the algorithm, one can use the information given by P_l by adding constraints $\Delta(x, x^i) \geq \kappa_i,\ i \in I^l$ and $\Delta(x, x^l) \geq \kappa + 1$ to (3.11)-(3.15), thus eliminating from further consideration the subregion defined by $\Delta(x, x^i) \geq \kappa_i,\ i \in I^l$ and $\Delta(x, x^l) \leq \kappa$. In this case, the local branching constraints are used in a similar way as the combinatorial Benders cuts presented by Codato and Fischetti [12]. They can either be used as a complement to the feasibility cuts or, if a local branching phase is used each time the relaxed master problem is solved and all integer variables in the original problem are binary, as a way to replace constraints (3.14).

Finally, it should be noted that during a local branching phase, the series of subproblems $P_0, P_1, \ldots, P_L$ that are solved retain the same structure as the original problem (3.1)-(3.5). Therefore, if Benders decomposition can be specialized to solve problem (3.1)-(3.5), then it can also be used to solve $P_0, P_1, \ldots, P_L$. Benders decomposition has the added advantage that the cuts that are obtained when solving a subproblem can be reused for the solution of future subproblems. This is an important observation to be made : by using Benders decomposition as the optimizer in local branching, the ideas proposed in this paper can be extended to all settings where Benders decomposition is applied or could be applied.

3.4 Computational Experiments

We chose the multicommodity capacitated fixed-charge network design problem (MCFND) to evaluate the performance of the algorithmic ideas proposed in this paper. As is apparent in the recent survey of Costa [13], network design problems have often

been solved successfully using Benders decomposition. What motivated the choice of the MCFND is the fact that it offers a simple formulation that represents well this general class of design problems. It is worth specifying that because the idea of using local branching in Benders decompostion is proposed in a general setting as a means of accelerating the general method, no effort was made to adapt the algorithms to the specific problem considered here. Therefore, all results reported in this section were only used to measure the tradeoffs in using local branching within the Benders relaxation algorithm. In no way, should this strategy be viewed as the optimal way of solving the MCFND.

One should specify that the MCFND will be defined using the classical notation used in the case of flow problems. Therefore, vector y will be used for the integer variables and vector x will refer to the continuous variables. Let $G(N, A, K)$ be a directed graph where N, A and K are respectively the sets of nodes, arcs and commodities that have to be routed over the network. For each arc $(i, j) \in A$, let us associate a boolean variable y_{ij} taking value 1 if (i, j) is used, and 0 otherwise. For each $(i, j) \in A$ and $k \in K$ let variable x_{ij}^k denote the flow of commodity k on arc (i, j). Let us also define parameters f_{ij}, c_{ij}^k, u_{ij} and d_k as being, respectively the fixed cost of using arc (i, j), the routing cost of one unit of commodity k through arc (i, j), the total capacity of arc (i, j) and the demand for commodity k, $\forall (i, j) \in A$ and $\forall k \in K$. The MCFND problem can then be formulated as :

$$\text{Min} \quad \sum_{(i,j)\in A} f_{ij} y_{ij} + \sum_{(i,j)\in A} \sum_{k\in K} c_{ij}^k x_{ij}^k \tag{3.20}$$

$$\text{s.t} \quad \sum_{j\in N_i^+} x_{ij}^k - \sum_{j\in N_i^-} x_{ji}^k = \begin{cases} d_k, \ i = O(k), \\ 0, \ i \notin \{O(k), D(k)\}, \ \forall i \in N, \ \forall k \in K, \\ -d_k, \ i = D(k), \end{cases} \tag{3.21}$$

$$\sum_{k\in K} x_{ij}^k \leq u_{ij} y_{ij}, \ \forall (i, j) \in A, \tag{3.22}$$

$$x_{ij}^k \geq 0, \ \forall (i, j) \in A, \ \forall k \in K \tag{3.23}$$

$$y_{ij} \in \{0, 1\}, \ \forall (i, j) \in A, \tag{3.24}$$

where $N_i^+ = \{j \mid (i,j) \in A\}$, $N_i^- = \{j \mid (j,i) \in A\}$ and the pair $\{O(k), D(k)\}$ corresponds to the origin and destination for the transportation of commodity $k \in K$. The MCFND consists of minimizing the fixed and routing costs (3.20) subject to the flow conservation constraints (3.21) and capacity constraints (3.22).

Applying the Benders decomposition scheme on problem (3.20)-(3.24), one obtains the following master problem :

$$Min \quad \sum_{(i,j)\in A} f_{ij}y_{ij} + \Theta \tag{3.25}$$

$$s.t \quad \sum_{(i,j)\in A} u_{ij}\pi_{ij}y_{ij} + \sum_{k\in K} d_k\alpha^k_{O(k)} - d_k\alpha^k_{D(k)} \leq \Theta, \ \forall(\pi,\alpha) \in \Lambda, \tag{3.26}$$

$$\sum_{(i,j)\in A} u_{ij}\pi_{ij}y_{ij} + \sum_{k\in K} d_k\alpha^k_{O(k)} - d_k\alpha^k_{D(k)} \leq 0, \ \forall(\pi,\alpha) \in \Phi, \tag{3.27}$$

$$y_{ij} \in \{0,1\}, \ \forall(i,j) \in A, \tag{3.28}$$

where sets Λ and Φ are defined as before.

All experiments were performed with a standard implementation of the Benders decomposition algorithm, including the features proposed by McDaniel and Devine [47] as well as Magnanti and Wong [45]. This implementation will be refered to as BD. It should be noted that during the LP phase, the linear relaxation of problem (3.20)-(3.24) is solved completely. Furthermore, all optimality cuts added during the LP and integer phases are Pareto-optimal cuts. As was already noted, no effort was made in order to specialize the BD implementation to the MCFND problem. The only exception is the addition of the following well-known cutset inequalities : $\sum_{j\in N_i^+} u_{ij}y_{ij} \geq \sum_{k\in K|i=O(k)} d^k, \ \forall i \in N$ and $\sum_{j\in N_i^-} u_{ij}y_{ij} \geq \sum_{k\in K|i=D(k)} d^k, \ \forall i \in N$. These inequalities state that for each node i the total capacity associated with the arcs leaving (resp. entering) node i must be at least equal to the sum of demands for which node i is an origin (resp. a destination).

Since the purpose of this paper is to assess how local branching can accelerate Benders decomposition, BD will be compared to the same implementation of Benders decomposition but in which a phase of local branching is performed after solving each

master problem in the integer phase of the solution process. The local branching procedure uses as starting points the solutions to the master problems in the integer phase. This enhanced algorithm will be referenced in the following as BD-LB.

Since the local branching search can be performed many times, a limit (Time-LB) is imposed on the time spent each time this procedure is called. The optimizer used in the local branching phase is the Benders algorithm itself. It is important to specify that during the LP phase, all cuts added to the relaxed master problem are also added to the relaxed master problem used for the optimizer. These cuts will form a basic pool to be used for all calls to the local branching procedure. Furthermore, all cuts obtained during a local branching search are kept in order to solve the series of subproblems. However, at the end of the local branching phase, these cuts are deleted. An optimality gap of 1% ($\bar{\epsilon} = 1\%$) was used for the solution of the subproblems. Since the objective is to quickly identify several feasible solutions, the branching scheme is applied each time a new solution is identified, wheter the objective function value associated with this new solution is better than that of the previous one or not. The diversification procedure is applied in the case where the subproblem obtained is infeasible. As for parameter κ, it starts with value one for all runs of the local branching procedure.

Another remark to be made concerns the core point $\hat{y}^0$ that is used in the evaluation of the deepest cut : point $\hat{y}^0$ is fixed to $\hat{y}^0_{ij} = 0.5$, $\forall (i,j) \in A$, at the beginning of the solution process and does not change. In order to test the validity of the deepest cut measure, two implementations of BD-LB will be tested. In BD-LB-All, all solutions found in the local branching phases will be used to generate cuts that will be added to the master problem, while in BD-LB-One only the solution that is identified as generating the deepest cut will be added at each iteration. One should note that in both cases, cuts are added in addition to the one obtained for the solution to the master problem. Also, in both implementations, all neighbourhoods found to be infeasible are eliminated from the feasible region of the master problem using the local branching constraints.

All instances of the MCFND problem were obtained using the *mulgen* generator

described in Crainic and *al.* [15]. A first set of instances are based on networks of 10 nodes, with 50, 60, 70 and 80 arcs, and 10 and 15 commodities. For each network size, 5 instances were generated randomly. For example, p1-10-50-10 will refer to the first instance whose network contains 10 nodes, 50 arcs and 10 commodities. This first set will be used to test the main algorithmic ideas proposed in this paper. A second set of instances, based on networks of 20 nodes, with 70 arcs and 10 or 15 commodities will then be used to validate the results. In this case, only 1 instance was generated for each network size. The data was generated so as to obtain instances for which the relative difference between the fixed and variable costs is either high compared to that of the arc capacities and demands (this will be refered to as set parameter 1), or low (wich will be set parameter 2). Therefore, the first set of problems contains 80 instances and the second set, four instances.

All experiments were performed on a 2.4 GHz AMD Opteron 64 bit processor. A maximum time of 10000 seconds was imposed on all runs for the first set of instances and the optimality gap considered was $\epsilon = 1\%$. As for the parameter Time-LB for both implementations using local branching, a first set of values was fixed to 1, 3, 5 and 7 seconds respectively, for the instances based on networks of 50, 60, 70 and 80 arcs. These times were then divided and multiplied by two in order to obtain two more sets of values that will provide a better understanding of how the results are affected when the effort devoted to the local branching phases varies. It should be noted that BD-LB-All-j and BD-LB-One-j for j= 1, 2 and 3, refer to the runs for each implementation with Time-LB fixed to the smallest, base and largest values.

In order to analyze the tradeoffs in using local branching, one should first compare BD to BD-LB-All, since BD-LB-All uses all information obtained by the local branching search. In Tables 3.1 and 3.2, solution times (in seconds) for the first set of problems are reported for the BD implementation and compared to BD-LB-All when Time-LB is fixed to the base values (BD-LB-All-2). Whenever an algorithm was unable to solve an instance in less than 10000 seconds, the solution time reported is > 10000 and the gap obtained is also given. Whenever the gap is not specified explicitly, one can consider that the algorithm has found an $\epsilon - opt$ solution.

Upon analyzing Tables 3.1 and 3.2, one may classify the instances as being easily solved (taking less than 1000 seconds), moderately hard to solve (taking between 1000 and 10000 seconds) and very hard to solve (more than 10000 seconds) for the BD implementation. By using such a classification, one distinguishes 35 easy instances, 19 moderate instances and 26 hard instances. Out of the 26 hard instances for BD, BD-LB-All-2 was able to solve 10 instances in less than 10000 seconds. One can also observe that BD-LB-All-2 is faster than BD for 70 out of the 80 instances. All instances for wich BD-LB-All-2 is slower than BD are considered easy instances. On these 10 cases, BD is, on average, about 5 % faster than BD-LB-All-2. Therefore, on all instances solved by both implementations before the time limit is reached (a total of 54 instances), one observes that BD-LB-All-2 is either faster (for 44 instances) or a little slower (for 10 instances) than BD.

To further analyze these results, let us consider Table 3.3. The first part of the table reports the average (Ave.), variance (Var.) and standard deviation (S.D.) for the solution times obtained by BD and BD-LB-All-2 on the easy and moderate instances. As for the hard instances, we report the average gap obtained on the 16 (out of the 26) instances that both BD and BD-LB-All-2 failed to solve in the maximum time allowed. One observes that on the easy instances, on average, BD-LD-All-2 is almost twice as fast as BD (84.21 sec. versus 163.68 sec.). On the moderate instances, the difference is even more significant, since BD-LB-All-2 is more than 4 times faster, on average, than BD (696.82 sec. versus 3021.13 sec.). For the hard instances, as was already noted, 10 out of the 26 instances were solved by BD-LB-All-2 before the time limit was reached. On some of these instances, the difference was quite large. For example, in the the case of p3-10-60-10.1, BD-LB-All-2 took 1239.2 seconds to complete the search, while BD was stopped with a gap of 9.14% after 10000 seconds. When both BD and BD-LB-All-2 were unable to complete the search within the time limit, on average, BD obtained a gap of 10.66% compared to the 5.71% gap obtained by BD-LB-All-2. This represents a relative decrease of 46.44%. What all these results seem to demonstrate is that, on average, one obtains significantly better results when using local branching in the context of the Benders decomposition algorithm.

Instances	BD	BD-LB-All-2
p1-10-50-10.1	24.8	**17.9**
p1-10-50-10.2	85.94	**40.03**
p1-10-50-15.1	3417.31	**467.7**
p1-10-50-15.2	1706.55	**344.98**
p2-10-50-10.1	6321.1	**1008.84**
p2-10-50-10.2	70.66	**36.67**
p2-10-50-15.1	927.44	**127.64**
p2-10-50-15.2	19.58	**6.45**
p3-10-50-10.1	**13.34**	53.92
p3-10-50-10.2	**5.01**	20.72
p3-10-50-15.1	73.51	**21.74**
p3-10-50-15.2	57.91	**35.88**
p4-10-50-10.1	480.94	**16.84**
p4-10-50-10.2	1249.43	**19.1**
p4-10-50-15.1	> 10000, 11.11%	> 10000, **3.56%**
p4-10-50-15.2	1455.82	**332.21**
p5-10-50-10.1	**8.05**	10.17
p5-10-50-10.2	**12.94**	54.64
p5-10-50-15.1	60.76	**5.48**
p5-10-50-15.2	25.75	**5.6**
p1-10-60-10.1	106.84	**103.82**
p1-10-60-10.2	**96.33**	158.02
p1-10-60-15.1	20.94	**8.35**
p1-10-60-15.2	3321.01	**450.14**
p2-10-60-10.1	6095.9	**1002.86**
p2-10-60-10.2	659.71	**50.51**
p2-10-60-15.1	> 10000, 10.10 %	**9548.79**
p2-10-60-15.2	407.48	**118.94**
p3-10-60-10.1	> 10000, 9.14 %	**1239.2**
p3-10-60-10.2	1.9	**1.3**
p3-10-60-15.1	> 10000, 7.94 %	> 10000, **4.92** %
p3-10-60-15.2	2974.62	**966.4**
p4-10-60-10.1	**40.09**	151.25
p4-10-60-10.2	1.28	**1.24**
p4-10-60-15.1	> 10000, 9.83 %	> 10000, **2.33** %
p4-10-60-15.2	> 10000, 3.74%	> 10000, **1.88** %
p5-10-60-10.1	104.29	**5.25**
p5-10-60-10.2	**4.55**	6.77
p5-10-60-15.1	> 10000, 1.20%	**2473.26**
p5-10-60-15.2	233.49	**3.82**

TAB. 3.1 – BD vs. BD-LB-All-2

Instances	BD	BD-LB-All-2
p1-10-70-10.1	> 10000, 12.70%	**7187.6**
p1-10-70-10.2	> 10000, 8.80%	**6560.24**
p1-10-70-15.1	**13.83**	40.61
p1-10-70-15.2	5281.74	**424.94**
p2-10-70-10.1	1773.31	**404.77**
p2-10-70-10.2	3.16	**1.41**
p2-10-70-15.1	> 10000, 23.40%	> 10000, **8.15 %**
p2-10-70-15.2	> 10000, 1.57%	**1051.98**
p3-10-70-10.1	1080.95	**316.23**
p3-10-70-10.2	1167.13	**338.17**
p3-10-70-15.1	> 10000, 11.37%	> 10000, **4.83 %**
p3-10-70-15.2	> 10000, 18.13%	> 10000, **13.64 %**
p4-10-70-10.1	1126.25	**192.65**
p4-10-70-10.2	310.38	**171.33**
p4-10-70-15.1	2453.59	**1017.95**
p4-10-70-15.2	165.55	**114.5**
p5-10-70-10.1	31.26	**18.03**
p5-10-70-10.2	5.67	**2.34**
p5-10-70-15.1	> 10000, 6.19%	> 10000, **5.44 %**
p5-10-70-15.2	1187.8	**685.88**
p1-10-80-10.1	502.08	**39.77**
p1-10-80-10.2	**160.04**	343.57
p1-10-80-15.1	> 10000, 11.39%	> 10000, **8.92 %**
p1-10-80-15.2	> 10000, 7.20%	> 10000, **2.55 %**
p2-10-80-10.1	> 10000, 2.80%	**2089.08**
p2-10-80-10.2	635.03	**605.22**
p2-10-80-15.1	> 10000, 22.41%	> 10000, **17.37 %**
p2-10-80-15.2	3725.01	**798.71**
p3-10-80-10.1	> 10000, 1.19%	**6520.39**
p3-10-80-10.2	**358.41**	547.62
p3-10-80-15.1	> 10000, 5.70%	> 10000, **3.3 %**
p3-10-80-15.2	> 10000, 1.006%	**2350.7**
p4-10-80-10.1	2773.39	**329.15**
p4-10-80-10.2	3479.37	**2368.79**
p4-10-80-15.1	> 10000, 18.21%	> 10000, **8.28 %**
p4-10-80-15.2	> 10000, 3.39%	> 10000, **2.10 %**
p5-10-80-10.1	> 10000, 5.47%	**7011.97**
p5-10-80-10.2	> 10000, 4.52%	> 10000, **1.37 %**
p5-10-80-15.1	> 10000, 6.09%	> 10000, **2.65 %**
p5-10-80-15.2	6811.16	**1770.05**

TAB. 3.2 – BD vs. BD-LB-All-2

Another important observation to be made concerns the standard deviations obtained by both implementations. For all three categories of instances, BD-LB-All-2 obtains smaller standard deviations than BD for the results considered in Table 3.3. In the case of the easy instances, one obtains a relative decrease of 38.36% in the standard deviation of the solution time when using BD-LB-All-2 compared to BD (142.01 sec. versus 230.37 sec.). For the moderate instances, the relative decrease is 69.49% (576.2 sec. versus 1888.33 sec.). As for the hard instances, one observes a relative decrease of 30.21% in the standard deviation of the gap obtained (4.55% versus 6.52%). By adding local branching searches throughout the integer phase of the Benders solution process, one makes the overall algorithm much more robust.

	BD			BD-LB-All-2		
Measures	easy	moderate	hard	easy	moderate	hard
Ave.	163.68 sec.	3021.13 sec.	10.66%	84.21 sec.	696.82 sec.	5.71%
Var.	53071.32	3565790.57	42.56	20166.66	332003.6	20.73
S.D.	230.37 sec.	1888.33 sec.	6.52%	142.01 sec.	576.2 sec.	4.55%
Fea.	21.54	82.53	-	8.66	32.26	-
Opt.	65.91	272.47	-	37.14	131.53	-
Nb. It.	102.03	356	-	19.69	78.47	-

TAB. 3.3 – BD vs. BD-LB-All-2

All these differences may be explained by analyzing the number of cuts added as well as the number of iterations executed by both implementations which are reported in the second part of Table 3.3. Fea., Opt. and Nb. It. refer respectively, to the average number of feasibility and optimality cuts added as well as the average number of iterations performed during the integer phase of the solution process. These results are not reported for the hard instances because the BD implementation failed to solve them. One should notice that for each iteration in the integer phase an integer master problem must be solved. When considering the instances that BD is able to solve in the maximum time allowed, one observes that the total number of cuts added by BD-LB-All-2 is, on average, much lower than BD. Since BD-LB-All-2 possibly adds multiple cuts at each iteration compared to only one in the case of BD, each additionnal cut in BD corresponds to at least one additionnal master problem to be solved. Indeed, when

comparing the total number of iterations needed in the integer phase (Nb. It.) by both implementations, one can see that, on average, BD solves about 5 times more integer master problems in the case of the easy and moderate instances. In turn, this explains the increased total solution times for BD with respect to BD-LB-All-2.

The comparison between BD and BD-LB-All seems to demonstrate that with a relatively small effort given to the local branching phases, one can obtain significantly better results when using Benders decomposition. This is particularly true for the cases where BD is less efficient. For moderate or hard instances, local branching can either considerably reduce the solution times or help to obtain better quality results.

	BD-LB-All			BD-LB-One		
Instances	1	2	3	1	2	3
Easy (sec.)	51.11	84.21	142.65	54.31	79.48	168.25
Moderate (sec.)	643.36	696.82	1162.75	691.11	831.96	1304.1
Hard - sol. (sec.)	2788.45	3662.1	4784.63	3011.77	3139.48	3822.59
Hard - not sol. (%)	6.61	6.2	6.64	6.27	6.83	6.68

TAB. 3.4 – BD-LB-All vs. BD-LB-One

One should now analyse how the results are affected when Time-LB varies as well as the tradeoff between using all feasible solutions to generate cuts (BD-LB-All) compared to using only the solution that is identified as generating the deepest cut (BD-LB-One). In Table 3.4 are reported the average results obtained by the BD-LB-All and BD-LB-One implementations. Concerning the hard instances, one should mention that Hard - sol. represents those instances that were solved to optimality in the maximum time allowed by all runs of BD-LB-All and BD-LB-One. As for Hard - not sol., it represents those that were not solved by all runs of BD-LB-All and BD-LB-One. When considering all instances that were solved, both implementations obtain the best average results when Time-LB is fixed to the smallest values. As for those hard instances that were not solved completely, the difference in the average gaps is quite small. Therefore, it seems that one only needs to invest a very limited effort in the local branching phases to obtain the best results when using either BD-LB-All or BD-LB-One.

When directly comparing the average times obtained by BD-LB-All and BD-LB-One, one can observe a slight advantage in using BD-LB-All. In order to better understand the solution process of implementations BD, BD-LB-All-1 and BD-LB-One-1 in the case of a hard instance (p3-10-80-10.1), let us consider Figures 3.1 to 3.3. In Figure 3.1 the evolution of the optimal gap is given. As was expected, BD-LB-All-1 and BD-LB-One-1 have a much faster convergence compared to BD. Figure 3.2 shows how the lower bound increases through the iterations in the integer phase. One can clearly see that the use of local branching gives a steeper ascent to the lower bound. When considering the evolution of the upper bound, illustrated in Figure 3.3, one also observes a favorable decrease when local branching is used. In the case of instance p3-10-80-10.1, BD-LB-All-1 is faster than BD-LB-One-1 and the difference is due to the evolution of the lower bound.When using BD-LB-All, one seems to hedge against the risk of not using a necessary cut by adding all cuts identified. This aspect of the BD-LB-All implementation constitutes an advantage on certain hard instances. However, one should note that the average differences between BD-LB-All and BD-LB-One are relatively small. Therefore, criteria (3.19), wich measures the deepness of each cut, does seem to work fairly well.

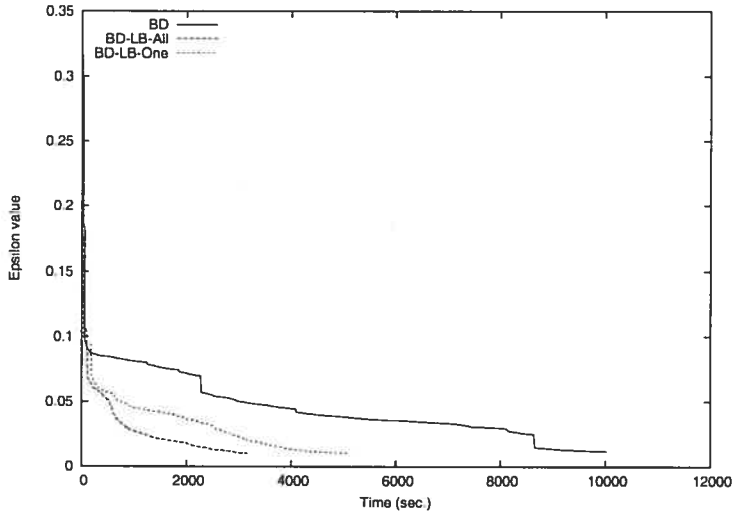

FIG. 3.1 – Time vs. Gap for p3-10-80-10.1

In order to validate the observations made so far, a second set of tests was obtained

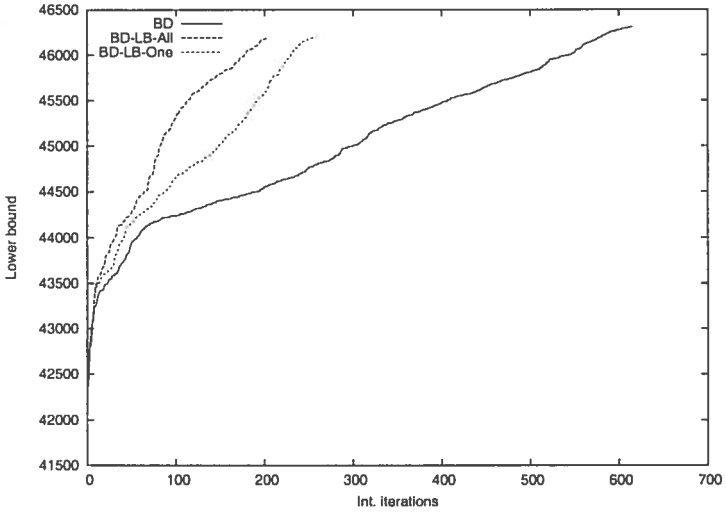

FIG. 3.2 – Int. iterations vs. Lower bound for p3-10-80-10.1

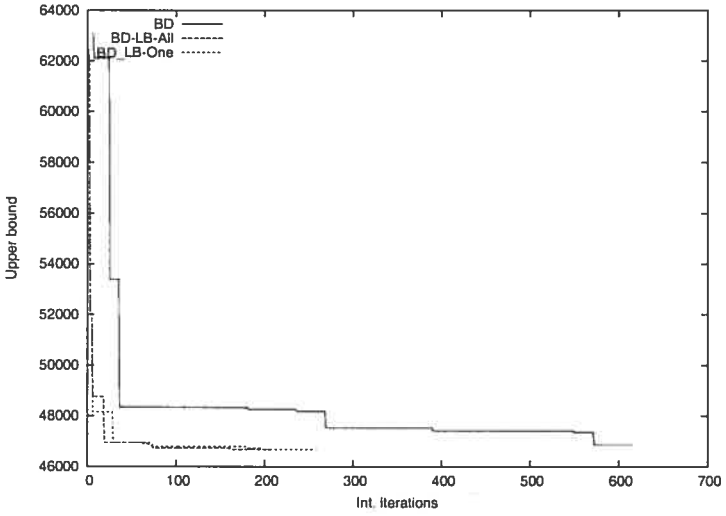

FIG. 3.3 – Int. iterations vs. Upper bound for p3-10-80-10.1

on instances based on larger networks (Table 3.5). For this second set of tests, the maximum time allowed was increased to 30,000 seconds. Also, since the previous tests seemed to demonstrate that, when using local branching in the Benders algorithm, one usually obtained the best results when Time-LB was fixed to the smallest values, Time-LB was set to 2.5 seconds for all runs in Table 3.5. These new results confirm previous observations. Once again, the use of local branching gives an added advantage when compared to BD. On these new instances, BD-LB-All also seems to be better compared to BD-LB-One. In the case of p20-70-10.2, BD-LB-All is about 2 times faster than BD-LB-One. As for p20-70-15.1, BD-LB-All obtains a relative reduction in solution time of 21.18% when compared to BD-LB-One.

	BD		BD-LB-One-1		BD-LB-All-1	
Instances	Time (sec.)	Gap	Time (sec.)	Gap	Time (sec.)	Gap
p20-70-10.1	$> 30,000$	12.37%	$> 30,000$	8.23%	$> 30,000$	11.22%
p20-70-10.2	1411.71	$< 1\%$	685.20	$< 1\%$	364.60	$< 1\%$
p20-70-15.1	$> 30,000$	6.97%	25177.90	$< 1\%$	19844.35	$< 1\%$
p20-70-15.2	$> 30,000$	7.31%	$> 30,000$	3.20%	$> 30,000$	2.43%

TAB. 3.5 – 2^{nd} Phase Tests

3.5 Conclusion

This paper proposes a novel way to accelerate the Benders decomposition algorithm by using local branching. By doing a local branching search after each master problem solved in the integer phase of the Benders algorithm, one can explore the neighborhood of the solution obtained for the master problem, which may or may not be feasible, in order to find different feasible solutions. By doing so, one can obtain better upper bounds at each iteration and since each different feasible solution can be used to generate an optimality cut, one can also simultaneously obtain better lower bounds. Furthermore, the subproblems that are found to be infeasible during the local branching phases can be used to produce inequalities that complement or replace the feasibility cuts added during the solution process. Therefore, all information obtained through the

local branching strategy can be used in the context of Benders decomposition.

Finally, since local branching only requires the use of a solver, if one can apply the Benders algorithm to the problem at hand then the same Benders algorithm can be used as this optimizer. This makes the use of local branching possible for all 0-1 integer problems that can be solved by Benders decomposition. Numerical tests in this paper have shown how local branching can greatly improve the performance of the standard Benders algorithm in the case of an integer linear problem (the MCFND problem). Recently, applications of Benders decomposition have focussed on non-linear or non-convex optimization problems. Clearly one interesting avenue of research would therefore be to verify that the acceleration technique proposed here remains as efficient in those settings.

Chapitre 4

Local Branching Cuts for the 0-1 Integer L-Shaped Algorithm

Au chapitre précédent, nous avons présenté une nouvelle approche permettant d'améliorer l'algorithme général de Benders [4]. À présent, nous verrons comment des idées similaires peuvent être spécialisées au cas de l'algorithme « L-shaped » pour les problèmes de type (1.1)-(1.5), où les variables de premières étape sont binaires et où le recours est entier (voir [39]). Notons que certaines des faiblesses évoquées précédemment pour la décomposition de Benders s'appliquent également dans le cas de l'algorithme « L-shaped » binaire. C'est notamment le cas pour la borne supérieure qui, encore une fois, n'offre aucune garantie de comportement monotone. Pour ce qui est de la borne inférieure, il est important de réaliser qu'une coupe d'optimalité du type (2.45) ne borne le recours que par rapport à la solution réalisable qui est utilisée pour la créer. L'information fournie par ces coupes est donc très locale. Si le recours n'est exprimé que par les inégalités (2.45), alors la qualité des bornes inférieures associées aux sous-problémes traités par l'algorithme de séparation et plans coupants sera faible. Dans ce cas, l'algorithme tendra à énumérer les solutions réalisables et le processus de résolution sera très long. Par conséquent, des phases de recherche par séparation locale peuvent de nouveau être utilisées afin d'accélérer la convergence de l'algorithme.

D'un point de vue méthodologique, nous verrons comment il est possible de générer un systéme d'inégalités valides à partir de l'information recueillie à la suite d'une phase de recherche par séparation locale. Ces inégalités valides sont utilisées afin de borner le recours dans les voisinages explorés. Plusieures motivations justifient cette approche. Dans un premier temps, il est généralement beaucoup plus simple et plus rapide d'obtenir de meilleures bornes inférieures dans des voisinages restreints du domaine réalisable des variables de première étape. Si de meilleures bornes inférieures sont obtenues localement, comparativement à la borne inférieure générale L, alors les inégalités proposées améliorent de façon stricte la description du recours dans les voisinages explorés. De plus, la séparation locale permet d'examiner des voisinages différents. Par conséquent, les inégalités développées dans cet article bornent le recours dans une region plus large du domaine réalisable comparativement aux coupes de type (2.45). Finalement, puisque ces phases de recherche peuvent également identifier de nouvelles solutions réalisables, alors il est possible d'améliorer plus rapidement la borne supérieure de l'algorithme.

Les inégalités proposées dans cet article sont applicables aux différents problèmes pouvant être résolus par l'algorithme « L-shaped » binaire. Puisqu'il est possible de les définir à partir de solutions réalisables ou non, elles offrent de nombreuses possibilités quant aux stratégies d'utilisation. Cependant, il est important de noter qu'elles nécessitent l'obtention de bornes inférieures dans les voisinages explorés lors des phases de séparation locale. De façon à appliquer ces inégalités, il est donc nécessaire de les spécialiser au cas traité.

Afin de tester ces nouvelles inégalités, nous avons choisi les problèmes de tournées d'un véhicule avec demandes stochastiques. Deux stratégies (locale et globale) concernant la génération de ces coupes seront présentées, de même qu'une technique permettant d'obtenir des bornes inférieures pour les voisinages explorés. Finalement, nous verrons comment les inégalités proposées peuvent être utilisées dans le cadre d'une procédure de génération de plans coupants adaptée aux problèmes de tournées d'un véhicule avec demandes stochastiques. Les résultats numériques obtenus montrent que l'approche proposée améliore de façon significative le meilleur algorithme exact existant pour le problème considéré.

Local Branching Cuts for the 0-1 Integer L-Shaped Algorithm

Walter Rei
Université de Montréal

Michel Gendreau
Université de Montréal

Patrick Soriano
HEC Montréal

November 2006

Abstract

Local branching has been presented as a new solution strategy for hard to solve mixed integer problems. Research has recently been done on using local branching as a way to generate optimality cuts in the case of Benders decomposition. In this paper, local branching is used to produce a new type of valid inequalities for the case of the 0-1 integer L-shaped algorithm. Numerical tests were conducted on a series of stochastic routing problems. Results prove the usefulness of these new inequalities.

Keywords. Stochastic programming, local branching, valid inequalities, vehicle routing problems with stochastic demands.

4.1 Introduction

Integer stochastic problems with fixed recourse can be defined as follows :

$$\text{Min} \quad c^\top x + \mathcal{Q}(x) \tag{4.1}$$

$$\text{s.t.} \quad Ax = b \tag{4.2}$$

$$x \in X, \tag{4.3}$$

$$\text{where } \mathcal{Q}(x) = \mathbf{E}_\xi[Q(x, \xi(\omega))] \tag{4.4}$$

$$\text{and } Q(x, \xi(\omega)) = Min_y\{q(\omega)^\top y \mid Wy = h(\omega) - T(\omega)x,\ y \in Y\}. \tag{4.5}$$

Problem (4.1)-(4.5) has two decision stages. Vector x represents the first stage. These decisions must be made before a series of stochastic parameters, represented by ξ, become known. In the second stage, one observes a certain random event, $\omega \in \Omega$, and ξ is fixed to $\xi(\omega)$. At this point, parameters $q(\omega)$, $h(\omega)$ and $T(\omega)$ that make up $\xi(\omega)$, become known. Problem (4.1)-(4.5) is said to have fixed recourse because matrix W is not dependent on the random event. In the second stage, vector y represents the recourse that one can take given that the first stage decisions were fixed at x. Function $Q : x \times \xi(\omega) \rightarrow \mathbb{R}_+$ measures the objective value of the recourse decisions. Therefore, function $\mathcal{Q} : x \rightarrow \mathbb{R}_+$, called the recourse function, gives the average value of recourse for x over all $\omega \in \Omega$. Problem (4.1)-(4.5) consists of finding a solution x that is minimal, on average, when considering ξ. Integrality constraints may be present in sets X and Y. As for the probability distribution of ξ, it may be either discrete or continuous.

Integer stochastic models have been successfully applied to a variety of contexts [7]. The algorithms that have been used to solve problems of type (4.1)-(4.5) have either been based on cutting plane strategies ([69], [3], [39], [9], [10], [61], [58] and [59]), branch-and-bound procedures ([49], [8] and [2]) or more recently on the use of test sets ([55] and [30]).

In this paper, we propose to use local branching as a way to generate a new type of cuts that can be introduced in the case of the 0-1 integer L-shaped method. This follows from the research initiated in Rei and *al.* [50] for the case of general Benders decomposition that showed how local branching could be used to simultaneously improve

the upper and lower bounds in the Benders algorithm. Similar ideas can be applied to the relaxation of the recourse function $\mathcal{Q}(x)$ in the case of the L-shaped branch-and-cut algorithm. The principles evoked here are applicable to all problems of type (4.1)-(4.5) having a subset of binary x variables. In this paper, however, they are illustrated and validated in the context of the single vehicle routing problem with stochastic demands (see [23], [32] and [40]), for which they lead to a significantly more efficient algorithm. The effectiveness is clearly demonstrated by computational experiments performed on a large set of instances whose difficulty ranges from easy to extremely difficult.

The remainder of this article is organized as follows. In section 2, a description of the 0-1 integer L-shaped method is given. Section 3 follows by providing an explanation of how a local branching search strategy may be applied to the case of the 0-1 integer L-shaped algorithm as well as a presentation of the valid inequalities that one can derive from it. In section 4, a presentation of the one vehicle routing problem is given as well as a description of the various cuts that can be applied to this problem. This is followed by computational results in section 5. Finally, this article is concluded in section 6.

4.2 The 0-1 integer L-shaped algorithm

The L-shaped algorithm was first introduced in 1969 by Van Slyke and Wets [66] to solve continuous stochastic programming problems with fixed recourse, in which random events are represented by a finite set of discrete scenarios. This algorithm can be interpreted as the application of Benders decomposition to the extensive form of the stochastic problem (see [7]). Indeed, when integrality constraints are introduced in set X, (4.1)-(4.5) becomes a mixed integer program and the L-shaped algorithm is then equivalent to the original Benders decomposition algorithm [4]. The problem becomes significantly harder when integrality constraints are introduced in set Y (integer recourse). In the Benders decomposition algorithm, cuts are created using the dual information provided by the solution of the subproblem. In the stochastic case, this subproblem is defined as follows : $Min_y\{q(\omega)^\top y \mid Wy = h(\omega) - T(\omega)x,\ y \in Y\}$ for all random events. One must realize that when integrality constraints are present

in Y, one can no longer apply standard continuous duality. Caroe and Tind [10] show how the L-shaped algorithm can be generalized to the case of integer recourse by using general duality theory. Convergence is maintained when either cutting plane algorithms or branch-and-bound procedures are used to solve the stochastic subproblems.

The 0-1 integer L-shaped method, which tackles problems with binary first stage variables and integer recourse, was introduced by Laporte and Louveaux [39]. In the following, we use the notation defined in their paper. Let us assume that vector x is of size n_1 and that set X is defined as follows : $X = \overline{X} \cap \{0,1\}^{n_1}$. Then let the following problem be the *current problem* :

$$\text{Min} \quad c^\top x + \Theta \tag{4.6}$$

$$\text{s.t.} \quad Ax = b \tag{4.7}$$

$$D_k x \geq d_k, \ k = 1, \ldots, s, \tag{4.8}$$

$$E_l x + \Theta \geq e_l, \ l = 1, \ldots, t, \tag{4.9}$$

$$0 \leq x \leq 1, \ \Theta \in \mathbb{R}. \tag{4.10}$$

Constraints of type (4.8) are called feasibility cuts, since they are used to induce feasible values of x. The authors define a set of feasibility cuts to be valid if there is a finite value s such that $x \in \overline{X}$ if and only if $\{D_k x \geq d_k, \ k = 1, \ldots, s\}$. Constraints of type (4.9) are said to be optimality cuts since they express possible feasible values of the recourse function. As before, a set of t optimality cuts is valid if the following condition is verified : $\forall x \in X, \ (x, \Theta) \in \{(x, \Theta) \mid E_l x + \Theta \geq e_l, \ l = 1, \ldots, t\}$ implies $\Theta \geq Q(x)$.

One can now define the 0-1 integer L-shaped algorithm as follows :

0-1 Integer L-shaped Algorithm

Step 0 (Initialization)

Set $\nu = 0$, $t = 0$, $s = 0$,

$\overline{z} = +\infty$,

$\Theta = -\infty$ or any valid general lower bound L.

Define the first pendant node as the initial current problem.

Step 1 (Selection)

Using a selection criterion, select a pendant node,

if there is none **STOP**.

Step 2 (Separation)

(2.1) $\nu = \nu + 1$.

(2.2) Solve the current problem,

if the current problem is infeasible **then**

fathom the node and go to **Step 1**,

else

let (x^ν, Θ^ν) be the optimal solution to the problem.

end if

(2.3) Search for violated constraints of type (4.8),

if one is found **then**

add it to the current problem, set $s = s + 1$ go to **(2.2)**,

else if $c^\top x^\nu + \Theta^\nu > \overline{z}$ **then**

fathom the node and go to **Step 1**.

end if

(2.4) Search for violated integrality constraints,

if one is found **then**

go to **Step 3**,

else

solution x^ν is feasible.

end if

(2.5) Compute $\mathcal{Q}(x^\nu)$, $z^\nu = c^\top x^\nu + \mathcal{Q}(x^\nu)$, $\overline{z} = Min\{\overline{z}, z^\nu\}$,

if $\Theta^\nu \geq \mathcal{Q}(x^\nu)$ **then**

fathom the node and go to **Step 1**,

else

add an optimality cut (4.9), set $t = t + 1$ go to **(2.2)**.

end if

Step 3 (Branching)

Using a branching criterion, create two new nodes, append them to the list of active nodes and go to **Step 1**.

The 0-1 integer L-shaped algorithm converges in a finite number of steps whenever a valid set of optimality and feasibility cuts exists. A general lower bound $\underline{z}$ can be obtained, at any point of the solution process, by considering the minimum of the lower bounds associated with the currently active nodes. This allows stopping the search procedure when an $\epsilon - opt$ solution is found.

Optimality cuts of type (4.9) can be expressed as follows : let us first define the $r - th$ feasible solution generated by the method as being $x_i = 1$, $i \in S_r$ and $x_i = 0$, $i \notin S_r$ and let Θ_r be the recourse value associated with this feasible solution. Laporte and Louveaux [39] show that the set of cuts defined for all feasible solutions r :

$$\Theta \geq (\Theta_r - L)\Big(\sum_{i \in S_r} x_i - \sum_{i \notin S_r} x_i\Big) - (\Theta_r - L)(|S_r| - 1) + L \tag{4.11}$$

where value L is a general lower bound on the value of recourse, is a valid set of optimality cuts. The proof of validity is evident when one considers that $\sum_{i \in S_r} x_i - \sum_{i \notin S_r} x_i$ is always less than or equal to $|S_r|$. The information provided by cut (4.11) is that whenever the solution considered is the $r-th$ feasible solution then the recourse is equal to Θ_r, else the value of recourse is less than L. One can also express the optimality cuts in the following way :

$$\sum_{i \in S_r} x_i \leq |S_r| - 1. \tag{4.12}$$

In the form of equation (4.12), the optimality cut is only used to eliminate from further consideration the $r - th$ feasible solution. In this case it provides no information on the value of recourse. However, it has the advantage of being composed of coefficients equal to one, which makes cut (4.12) more numerically stable than (4.11). As for the feasibility cuts of type (4.8), one can only say that they are problem dependent.

The main difficulty associated with the 0-1 integer L-shaped algorithm resides in the approximation of the recourse function. The information provided by optimality cuts of either type (4.11) or (4.12) is very local. Constraints (4.11) only bound the value of recourse associated with the feasible solutions that were used to create them. As for (4.12), they only eliminate the feasible solutions that were encountered by the algorithm. Therefore, in problem (4.6)-(4.10), the value of recourse will only be determined by the subset of optimality constraints present and the value of the general bound L. If one uses exclusively (4.12), then only the general bound L approximates the recourse function. If the quality of bound L is poor, then the L-shaped algorithm will tend to enumerate feasible solutions. The reason for this is that in the separation step (more precisely in **(2.3)**), the quality of bound $c^\top x^\nu + \Theta^\nu$ will be poor whenever x^ν is infeasible, resulting in an increase of the number of active subproblems. To solve this problem, one can either provide better general lower bounds L or use other lower bounding functionals to better approximate the recourse function. In [39], an improved optimality cut is presented where if a better lower bound exists in a close neighbourhood of the feasible solution considered, then (4.11) is strenghtened by this information. In the next section, a series of general valid inequalities will be presented to extend this idea of using better lower bounds around solutions to the *current problem*. These valid inequalities will help to provide a better description of the recourse values for different neighbourhoods of the first stage feasible region.

4.3 Local branching cuts for the 0-1 integer L-shaped algorithm

Local branching was introduced in 2003 by Fischetti and Lodi [20]. The idea behind this method is to take advantage of the efficiency of generic solvers, such as CPLEX, for solving small integer 0-1 problems. Therefore, one can divide the feasible space of a problem into a series of smaller subregions and then use a generic solver to explore efficiently each of the subregions thus created.

In the case of 0-1 integer problems, the function used to divide the feasible region is the Hamming distance defined from a particular integer point. Let us consider the general stochastic problem (4.1)-(4.5), for which $X = \overline{X} \cap \{0,1\}^{n_1}$. Let x^0 be a vector of 0-1 values, then the Hamming distance function relative to x^0 is : $\Delta(x, x^0) = \sum_{j \in S_0} (1 - x_j) + \sum_{j \in N_1 \setminus S_0} x_j$ (where $N_1 = \{1, \ldots, n_1\}$ and $S_0 = \{j \in N_1 \mid x_j^0 = 1\}$). Using function $\Delta(x, x^0)$, for any integer κ, one can divide the feasible region of (4.1)-(4.5) by creating two subproblems, one in which the constraint $\Delta(x, x^0) \leq \kappa$ is added, and the other in which $\Delta(x, x^0) \geq \kappa + 1$ is added. Constraint $\Delta(x, x^0) \leq \kappa$ can considerably reduce the size of the feasible region of problem (4.1)-(4.5) when κ is fixed to an appropriate (small) value. Therefore, one can use an adapted generic solver to solve this subproblem. Using the new solution found, the procedure may continue by dividing the subregion defined by $\Delta(x, x^0) \geq \kappa + 1$ into two more subproblems where the smaller subregion is explored in the same way as before.

In [50], the information associated with a local branching descent is used to produce multiple optimality cuts in the case of classical Benders decomposition. Through this process, one is also able to improve the upper bound generated by the Benders algorithm. Furthermore, it is shown how local branching constraints can be used to replace or complement the feasibility cuts. We will use similar ideas to generate a series of valid inequalities that approximate the value of recourse in different subregions of X. Let us first define the two following subproblems :

$$
\begin{array}{llll}
(P_n) & \text{Min} & c^\top x + \mathcal{Q}(x) & \\
& \text{s.t} & Ax = b & \\
& & \Delta(x, x^i) \geq \kappa_i, \ i \in I^n & \\
& & \Delta(x, x^n) \leq \kappa_n & \\
& & x \in X & \\
\end{array}
\qquad
\begin{array}{llll}
(\overline{P}_n) & \text{Min} & c^\top x + \mathcal{Q}(x) & \\
& \text{s.t} & Ax = b & \\
& & \Delta(x, x^i) \geq \kappa_i, \ i \in I^n & \\
& & \Delta(x, x^n) \geq \kappa_n + 1 & \\
& & x \in X, & \\
\end{array}
$$

where I^n corresponds to a set of 0-1 vectors that may or may not represent feasible first stage solutions. A local branching step consists of solving subproblem P_n and then using the solution obtained to separate the feasible region of subproblem $\overline{P}_n$. If P_n is found to be infeasible, a diversification strategy is applied and the problem is solved again. In this case, the diversification strategy consists of increasing value κ_n to obtain

a larger feasible region for P_n.

A local branching descent is composed of a series of subproblems P_n for $n = 1, \ldots, m$ that correspond to the subregions explored by the algorithm. Let us suppose that one is able to find a lower bound $\overline{\Theta}_n$ (for which $\overline{\Theta}_n > L$) on the value of the recourse in each of the subregions associated with subproblems P_n, $n = 1, \ldots, m$. Let us also redefine the *current problem* in the following way :

$$\text{Min} \quad c^\top x + \Theta \tag{4.13}$$

$$\text{s.t.} \quad Ax = b \tag{4.14}$$

$$D_k x \geq d_k, \ k = 1, \ldots, s, \tag{4.15}$$

$$E_l x + \Theta \geq e_l, \ l = 1, \ldots, t, \tag{4.16}$$

$$\Delta(x, x^i) \geq 1, \ i \in \bigcup_{n=1}^{m} I^n, \tag{4.17}$$

$$0 \leq x \leq 1, \ \Theta \in \mathbb{R}. \tag{4.18}$$

In (4.17), one should realize that constraint $\Delta(x, x^i) \geq 1$ eliminates locally vector x^i since it imposes that the Hamming distance between x and x^i be at least one. If x^i is feasible, then $\Delta(x, x^i) \geq 1$ is equivalent to the optimality cut (4.12). If x^i is infeasible, then $\Delta(x, x^i) \geq 1$ may serve as a feasibility cut. Therefore, model (4.13)-(4.18) is either equivalent to (4.6)-(4.10) or it offers a better description of the feasible region. One can now derive the following result :

Proposition 4.3.1 (Local branching valid inequalities). *Let P_n, $n = 1, \ldots, m$, define a local branching descent and $\overline{\Theta}_n$, $n = 1, \ldots, m$, be valid lower bounds on the recourse value for each of the subproblems in the descent, then the following system of*

equations defines a set of valid inequalities for problem (4.13)-(4.18) :

$$\Theta \geq L + (\overline{\Theta}_n - L)w_n, \ n = 1, \ldots, m \tag{4.19}$$

$$\kappa_n - \Delta(x, x^n) \leq n_1 \sum_{j=1}^{n} w_j, \ n = 1, \ldots, m \tag{4.20}$$

$$\Delta(x, x^n) - \kappa_n \leq (1 - w_n)n_1, \ n = 1, \ldots, m \tag{4.21}$$

$$w_{n+1} \leq \overline{w}_n, \ n = 1, \ldots, m-1 \tag{4.22}$$

$$\sum_{j=1}^{n-1} w_j + \overline{w}_{n-1} = 1, \ n = 2, \ldots, m \tag{4.23}$$

$$w_n \in \{0, 1\}, \ n = 1, \ldots, m \tag{4.24}$$

$$\overline{w}_n \in \{0, 1\}, n = 1, \ldots, m-1, \tag{4.25}$$

where x^n, $n = 1, \ldots, m$, is a series of 0-1 vectors (that may or may not be feasible first stage solutions), such that $x^n \in \{x \mid \Delta(x, x^{n-1}) \leq \kappa_{n-1}, \Delta(x, x^i) \geq \kappa_i + 1, i \in I^{n-1}\}$, $n = 2, \ldots, m$.

Proof. The proposition will be proved by induction.

- Base case : (4.19)-(4.25) is a set of valid inequalities for (4.13)-(4.18) when $m = 1$.

If $m = 1$, then system (4.19)-(4.25) reduces to :

$$\Theta \geq L + (\overline{\Theta}_1 - L)w_1, \tag{4.26}$$

$$\kappa_1 - \Delta(x, x^1) \leq w_1 n_1, \tag{4.27}$$

$$\Delta(x, x^1) - \kappa_1 \leq (1 - w_1)n_1, \tag{4.28}$$

$$w_1 \in \{0, 1\}. \tag{4.29}$$

Let $\hat{x}$ be a feasible solution to problem (4.13)-(4.18). One should note that $\hat{x}$ is either integer or continuous and either feasible or not with respect to the original problem (4.1)-(4.5). There are three possible cases when one considers $\hat{x}$:

1. $0 < \Delta(\hat{x}, x^1) < \kappa_1$,
2. $0 < \kappa_1 < \Delta(\hat{x}, x^1)$,
3. $\Delta(\hat{x}, x^1) = \kappa_1$.

Case 1 : if $0 < \Delta(\hat{x}, x^1) < \kappa_1$, then constraint (4.27) forces $w_1 = 1$. If $w_1 = 1$, then (4.26) becomes $\Theta \geq \overline{\Theta}_1$ which is valid since $\Delta(\hat{x}, x^1) \leq \kappa_1$. Also, in this case, if $w_1 = 1$, then constraint (4.28) is satisfied.

Case 2 : if $0 < \kappa_1 < \Delta(\hat{x}, x^1)$, then constraint (4.28) forces $w_1 = 0$. If $w_1 = 0$, then (4.26) becomes $\Theta \geq L$ which is valid. Also, in this case, if $w_1 = 0$ then constraint (4.27) is satisfied.

Case 3 : if $\Delta(\hat{x}, x^1) = \kappa_1$, then constraints (4.27) and (4.28) allow $w_1 = 1$ or $w_1 = 0$. If $w_1 = 1$, then (4.26) becomes $\Theta \geq \overline{\Theta}_1$ which is valid since $\Delta(\hat{x}, x^1) \leq \kappa_1$. Otherwise, if $w_1 = 0$, then (4.26) becomes $\Theta \geq L$ which is also valid. Considering that the *current problem* is a minimization problem and since $\overline{\Theta}_1 > L$, then, in this case, variable w_1 is fixed to 0 ($w_1 = 0$) and (4.26) becomes $\Theta \geq L$.

- Hypothesis $H1$: (4.19)-(4.25) is a set of valid inequalities for (4.13)-(4.18) when $m = i$.

- Step : if $H1$ is true then (4.19)-(4.25) is a set of valid inequalities for (4.13)-(4.18) when $m = i + 1$.

If $m = i$, then system (4.19)-(4.25) becomes :

$$\Theta \geq L + (\overline{\Theta}_n - L)w_n, \ n = 1, \ldots, i \tag{4.30}$$

$$\kappa_n - \Delta(x, x^n) \leq n_1 \sum_{j=1}^{n} w_j, \ n = 1, \ldots, i \tag{4.31}$$

$$\Delta(x, x^n) - \kappa_n \leq (1 - w_n)n_1, \ n = 1, \ldots, i \tag{4.32}$$

$$w_{n+1} \leq \overline{w}_n, \ n = 1, \ldots, i - 1 \tag{4.33}$$

$$\sum_{j=1}^{n-1} w_j + \overline{w}_{n-1} = 1, \ n = 2, \ldots, i \tag{4.34}$$

$$w_n \in \{0, 1\}, \ n = 1, \ldots, i \tag{4.35}$$

$$\overline{w}_n \in \{0, 1\}, n = 1, \ldots, i - 1. \tag{4.36}$$

If one adds another level to the local branching descent (from $m = i$ to $m = i + 1$), one

adds to (4.30)-(4.36) the following inequalities :

$$\Theta \geq L + (\overline{\Theta}_{i+1} - L)w_{i+1}, \tag{4.37}$$

$$\kappa_{i+1} - \Delta(x, x^{i+1}) \leq n_1 \sum_{j=1}^{i+1} w_j, \tag{4.38}$$

$$\Delta(x, x^{i+1}) - \kappa_{i+1} \leq (1 - w_{i+1})n_1, \tag{4.39}$$

$$w_{i+1} \leq \overline{w}_i, \tag{4.40}$$

$$\sum_{j=1}^{i} w_j + \overline{w}_i = 1, \tag{4.41}$$

$$w_{i+1} \in \{0, 1\}, \tag{4.42}$$

$$\overline{w}_i \in \{0, 1\}. \tag{4.43}$$

Let us now define the following set : $X_i = \bigcup_{n=1}^{i} \{x \mid \Delta(x, x^n) \leq \kappa_n\}$. Once more, let $\hat{x}$ be a feasible solution to problem (4.13)-(4.18). There are two possible cases when one considers $\hat{x}$:

1. $\hat{x} \notin X_i$,
2. $\hat{x} \in X_i$.

Case 1 : if $\hat{x} \notin X_i$ (i.e. $\Delta(\hat{x}, x^n) > \kappa_n$ for $n = 1, \ldots, i$), then (4.32) forces $w_n = 0$, $n = 1, \ldots, i$. When one considers system (4.37)-(4.43), one must first see that since $w_n = 0$, $n = 1, \ldots, i$, then (4.41) implies that $\overline{w}_i = 1$. Constraint (4.40) becomes $w_{i+1} \leq 1$ which makes it possible for variable w_{i+1} to be equal to either 0 or 1. Since $\hat{x} \notin X_i$, then three cases are possible :

- $0 < \Delta(\hat{x}, x^{i+1}) < \kappa_{i+1}$,
- $0 < \kappa_{i+1} < \Delta(\hat{x}, x^{i+1})$,
- $\Delta(\hat{x}, x^{i+1}) = \kappa_{i+1}$.

Since $w_n = 0$, $n = 1, \ldots, i$, if $0 < \Delta(\hat{x}, x^{i+1}) < \kappa_{i+1}$, then constraint (4.38) forces $w_{i+1} = 1$. If $w_{i+1} = 1$, then (4.37) becomes $\Theta \geq \overline{\Theta}_{i+1}$ which is valid since $\Delta(\hat{x}, x^{i+1}) \leq \kappa_{i+1}$. Also, if $w_{i+1} = 1$, then constraint (4.39) is satisfied. If $0 < \kappa_{i+1} < \Delta(\hat{x}, x^{i+1})$, then constraint (4.39) forces $w_{i+1} = 0$. If $w_{i+1} = 0$, then (4.37) becomes $\Theta \geq L$ which is valid. Also, in this case, if $w_{i+1} = 0$ then constraint (4.38) is satisfied. If $\Delta(\hat{x}, x^{i+1}) = \kappa_{i+1}$,

then constraints (4.38) and (4.39) allow $w_{i+1} = 1$ or $w_{i+1} = 0$. If $w_{i+1} = 1$, then (4.37) becomes $\Theta \geq \overline{\Theta}_{i+1}$ which is valid since $\Delta(\hat{x}, x^{i+1}) \leq \kappa_{i+1}$. Otherwise, if $w_{i+1} = 0$, then (4.37) becomes $\Theta \geq L$ which is also valid. Once again, considering that the *current problem* is a minimization problem and since $\overline{\Theta}_{i+1} > L$, then, in this case, variable w_{i+1} is fixed to 0 ($w_{i+1} = 0$) and (4.37) becomes $\Theta \geq L$.

Case 2 : if $\hat{x} \in X_i$, then there are two possible cases to consider :

1. there exists at least one index n, $1 \leq n \leq i$, such that $\Delta(\hat{x}, x^n) < \kappa_n$,
2. $\Delta(\hat{x}, x^n) \not< \kappa_n$, for $n = 1, \ldots, i$.

Subcase 1 : If there exists at least one index n, $1 \leq n \leq i$, such that $\Delta(\hat{x}, x^n) < \kappa_n$, then constraints (4.31) and (4.34) imply that there exists exactly one index $1 \leq i^\star \leq i$ such that $w_{i^\star} = 1$. If $w_{i^\star} = 1$, then (4.30) reduces to $\Theta \geq \overline{\Theta}_{i^\star}$, which is valid because of $H1$. Considering system (4.37)-(4.43), since $w_{i^\star} = 1$, constraint (4.41) implies that $\overline{w}_i = 0$. Then by (4.40), $w_{i+1} = 0$. If $w_{i+1} = 0$, then (4.37) reduces to $\Theta \geq L$ which is valid. One must now consider two possible cases :

- $0 < \Delta(\hat{x}, x^{i+1}) \leq \kappa_{i+1}$,
- $0 < \kappa_{i+1} < \Delta(\hat{x}, x^{i+1})$.

If $0 < \Delta(\hat{x}, x^{i+1}) \leq \kappa_{i+1}$, then (4.38) is satisfied since $i^\star < i + 1$. If $0 < \kappa_{i+1} < \Delta(\hat{x}, x^{i+1})$, since $w_{i+1} = 0$, then (4.39) is satisfied.

Subcase 2 : if $\Delta(\hat{x}, x^n) \not< \kappa_n$, for $n = 1, \ldots, i$, and $\hat{x} \in X_i$, then, there exists an index set $J \subseteq \{1, \ldots, i\}$ such that $\Delta(\hat{x}, x^j) = \kappa_j$, $\forall j \in J$, and $\Delta(\hat{x}, x^j) > \kappa_j$, $\forall j \in \{1, \ldots, i\} \setminus J$. Without loss of generality, let us assume that $\Delta(\hat{x}, x^n) = \kappa_n$ for $n = 1, \ldots, j$, and $\Delta(\hat{x}, x^n) > \kappa_n$ for $n = j + 1, \ldots, i$. Constraints (4.32) imply that $w_n = 0$ for $n = j + 1, \ldots, i$. As for $n = 1, \ldots, j$, constraints (4.31) and (4.32) are always satisfied. Constraints (4.34) and (4.33) imply two possible alternatives : either there exists exactly one index $1 \leq i^\star \leq j$ such that $w_{i^\star} = 1$ and hence $\overline{w}_n = 1$ for $n = 1, \ldots, i^\star - 1$ **or else** $\overline{w}_n = 1$ for $n = 1, \ldots, j, \ldots, i - 1$ and $w_n = 0$ for $n = 1, \ldots, j, \ldots, i$. Let us consider the first alternative, if $w_{i^\star} = 1$ and $\overline{w}_n = 1$ for $n = 1, \ldots, i^\star - 1$ then (4.30) reduces to $\Theta \geq \overline{\Theta}_{i^\star}$. Otherwise, if $\overline{w}_n = 1$ for $n = 1, \ldots, i - 1$ and $w_n = 0$ for $n = 1, \ldots, i$ then (4.30) reduces to $\Theta \geq L$. Once again, since the *current problem* is a minimization

problem and given that $\overline{\Theta}_n > L$ for $n = 1, \ldots, i$ (hence $\overline{\Theta}_{i^*} > L$), therefore the second alternative will always be selected and (4.30) becomes $\Theta \geq L$, which is valid. This situation is identical to the one encountered in case 1 (i.e., case where $x \notin X_i$) and one must therefore consider the same three possibilities :

- $0 < \Delta(\hat{x}, x^{i+1}) < \kappa_{i+1}$,
- $0 < \kappa_{i+1} < \Delta(\hat{x}, x^{i+1})$,
- $\Delta(\hat{x}, x^{i+1}) = \kappa_{i+1}$.

For each of these, the validity of system (4.37)-(4.43) can be proven as before and will not be repeated here. □

The system of inequalities (4.19)-(4.25) is used to bound the value of recourse following the local branching descent. If one considers a feasible solution $\hat{x}$ to problem (4.13)-(4.18), then a valid lower bound on the value of recourse is provided for $\hat{x}$ in the first subproblem P_n, $n = 1, \ldots, m$, for which solution $\hat{x}$ is feasible. Otherwise, the value of recourse is bounded by L. If $\overline{\Theta}_n > L$ for $n = 1, \ldots, m$, then (4.19)-(4.25) offers a better description of the possible values of the recourse function in the subregions explored in the local branching descent. Furthermore, since (4.19)-(4.25) bounds $\mathcal{Q}(x)$ in different subregions of X compared to only bounding the recourse associated with a feasible solution as in the case of the classical optimality cut (4.11), by using (4.19)-(4.25), one may obtain useful information on a wider range of solutions for problem (4.13)-(4.18).

One must now find a way to generate lower bounds on the subproblems explored during the local branching descent. To do so, one can either use a lower bounding functional that is specific to the problem being solved or, one can adopt a more general approach. It is observed in [50] that the local branching subproblems retain the same structure as the original problem. Therefore, since the L-shaped algorithm is applied on the original problem, one can also apply the same algorithm on the local branching subproblems. In [50], the local branching subproblems are solved until an $\epsilon-opt$ solution was found. To obtain the lower bounds $\overline{\Theta}_n$, $n = 1, \ldots, m$, one may use the same

approach. Subproblems P_n, $n = 1, \ldots, m$, can be solved using the L-shaped algorithm until an $\epsilon - opt$ solution is found. The lower bound on recourse can then be derived from the general lower bound $\underline{z}$. In this case, through the use of local branching, one is exploring the feasible region of the original problem (4.1)-(4.5).

There is another way of using (4.19)-(4.25), which is to generate them locally for each of the subproblems explored by the 0-1 integer L-shaped algorithm. In step 2 of the algorithm, one may either apply a local branching descent to cut a solution for which the separation routine is unable to find a feasibility cut that is violated or to tighten the lower bound $c^\top x^\nu + \Theta^\nu$ associated with the subproblem considered. By working locally on the subproblems of the 0-1 integer L-shaped algorithm, one reduces considerably the size of subproblems P_n, $n = 1, \ldots, m$, making it easier to find better quality lower bounds. The drawback of such an approach is that the inequalities that are found are only valid locally. However, if one is able to obtain better lower bounds for each of the subproblems explored by the L-shaped algorithm, then one obtains a better general lower bound $\underline{z}$. Furthermore, by increasing values $c^\top x^\nu + \Theta^\nu$, one is able to fathom the subproblems at an earlier stage. By reducing the number of active subproblems, one can accelerate the convergence of the algorithm. These principles will now be applied to the case of the single vehicle routing problem with stochastic demands.

4.4 The single vehicle routing problem with stochastic demands

To present the single vehicle routing problem with stochastic demands, let us first define an undirected graph $G(V, E)$, where $V = \{v_1, \ldots, v_n\}$ is a set of vertices and $E = \{(v_i, v_j) : v_i, v_j \in V, i < j\}$ is a set of edges. Defined on E is a symmetric matrix $C = [c_{ij}]$ that corresponds to the travel costs between vertices. Vertex v_1 represents a depot where a vehicle must start and finish a route. This route must visit each of the customers (i,e., $V \setminus \{v_1\}$) once while minimizing the total travel cost. Up until now, this problem represents the classical traveling salesman problem. However, the single

vehicle routing problem with stochastic demands has two additional characteristics. The vehicle has a limited capacity D and each of the customers $j \in V \setminus \{v_1\}$ has a nonnegative demand that may be stochastic. Numerical tests will be conducted on instances where each client $j \in V \setminus \{v_1\}$ has a demand that is a Normal random variable (i.e., $\xi_j \sim N(\mu_j, \sigma_j)$, truncated at zero) and all demands are independently distributed.

In this case, whenever the vehicle serves a certain client, there is always the risk that the observed demand exceeds the residual capacity. When such a failure occurs, partial delivery is performed and the recourse action taken is to return to the depot, to stock up (or to unload), and then go back to the customer where failure occured to finish the delivery and continue the route. One makes the hypothesis that the stochastic demands are unknown when routing decisions are being made. Therefore, the problem has two stages. One must first construct a route and then the cost of recourse may be determined. The optimization problem consists of finding a solution that minimizes simultaneously the original travel cost and the expected cost of recourse. The model is therefore defined as follows :

$$\text{Min} \quad \sum_{i<j} c_{ij} x_{ij} + \mathcal{Q}(x) \tag{4.44}$$

$$\text{s.t.} \quad \sum_{j=2}^{n} x_{1j} = 2, \tag{4.45}$$

$$\sum_{i<k} x_{ik} + \sum_{j>k} x_{kj} = 2, \ k = 2, \ldots, n, \tag{4.46}$$

$$\sum_{i \in S} \sum_{\substack{j \notin S \\ j>i}} x_{ij} + \sum_{i \notin S} \sum_{\substack{j \in S \\ j>i}} x_{ij} \geq 2, \ S \subseteq V, \ |S| \geq 3, \tag{4.47}$$

$$x_{ij} \in \{0, 1\}, \ 1 \leq i < j \leq n. \tag{4.48}$$

An important point to be made concerns the orientation of the routes that are considered. In a deterministic context, this concept is irrelevant since the cost of the solution remains the same for both orientations. However, in the stochastic case, because of the cost of recourse, the value of a solution depends on the orientation chosen. Since demands become known only when customers are visited, one must decide, a priori, which orientation to use. Therefore, the recourse function is : $\mathcal{Q}(x) = Min\{\mathcal{Q}^1(x), \mathcal{Q}^2(x)\}$,

where $\mathcal{Q}^1(x)$ and $\mathcal{Q}^2(x)$ correspond to the recourse cost for each of the two orientations. Let us define the $r-th$ feasible solution x^r by vector $V^r = (v_{r_1} = v_1, v_{r_2}, \ldots, v_{r_{n+1}} = v_1)$. If the goods to be delivered (or collected) are divisible, then one may measure the recourse of solution x^r for the first orientation in the following way (see [40]) :

$$\mathcal{Q}^1(x^r) = 2\sum_{j=2}^{n}\sum_{l=1}^{\infty} P\left(\sum_{s=2}^{j-1}\xi_{r_s} \leq lD < \sum_{s=2}^{j}\xi_{r_s}\right) c_{1r_j}.$$

The probability term is associated with the event of having the lth failure occur at customer r_j. The computation of $\mathcal{Q}^2(x^r)$ follows the same principles, one only needs to replace the index of ξ by r_{n+2-s}.

To solve (4.44)-(4.48) using the 0-1 integer L-shaped algorithm, one must first relax constraints (4.47) and (4.48) and then approximate the recourse function $\mathcal{Q}(x)$ using Θ. Constraints of type (4.47) are added in step **(2.3)** of the algorithm and one may use optimality cuts of type (4.11) or (4.12), whenever a feasible solution is obtained.

Gendreau and *al.* [23] were the first to apply the standard L-shaped algorithm to the single vehicle routing problem with stochastic demands. In 1999, Hjorring and Holt [32] proposed a new type of cut that uses information taken from partial routes for problem (4.44)-(4.48). A partial route is made up of three sets. Using the notation proposed by Laporte and *al.* [40], let us first define the two ordered sets $S = \{v_1, \ldots, v_s\}$ and $T = \{v_1, \ldots, v_t\}$. Sets S and T must respect the following condition : $S \cap T = \{v_1\}$. Let us now define a third set $U = V \setminus ((S \setminus \{v_s\}) \cup (T \setminus \{v_t\}))$. One easily sees that $S \cap U = \{v_s\}$ and $T \cap U = \{v_t\}$. Therefore, a partial route is made up of the two vectors $(v_1, \ldots, v_s)$ and $(v_t, \ldots, v_1)$ which define the beginning and end of the route, and of set U, which contains all vertices that are not yet ordered. If $(v_i, v_j) \in S$ or T refers to the case where v_i and v_j are consecutive in S or T, then let $W(x) = \sum_{(v_i,v_j)\in S} x_{ij} + \sum_{(v_i,v_j)\in T} x_{ij} + \sum_{v_i,v_j\in U} x_{ij} - |V| + 1$. If Q is a lower bound on the value of recourse for the partial route, then the following inequality is valid for problem (4.44)-(4.48) :

$$\Theta \geq L + (Q - L)W(x). \tag{4.49}$$

To obtain a lower bound Q, Hjorring and Holt [32] propose to create an artificial

vertex v_0 for which the demand is $\xi_0 = \sum_{v_i \in U \setminus \{v_s, v_t\}} \xi_i$ and the cost of return to the depot is $c_{10} = \min_{v_i \in U \setminus \{v_s, v_t\}} \{c_{1i}\}$. By using cost c_{10}, one can calculate a lower bound on the partial route by measuring the value of recourse for $(v_1, \ldots, v_s, v_0, v_t, \ldots, v_1)$. As for the general lower bound L, one can use the same technique as the one that Laporte and *al.* [40] propose for the case of the general stochastic vehicle routing problem.

Inequalities (4.49) may be added in step **(2.3)** of the 0-1 integer L-shaped algorithm since one can generate them using an integer or continuous solution x^ν. There is an important point to be made concerning the separation algorithm needed to find violated inequalities of type (4.49). Let us consider (x^ν, Θ^ν), a solution to a *current problem* solved by the L-shaped algorithm. Vector x^ν is either integer or continuous. Since there is only one vehicle in problem (4.44)-(4.48), one can always construct a partial route using x^ν for which $W(x^\nu) = 1$. Therefore, one obtains a violated inequality (4.49) using this partial route if $\Theta^\nu < Q$.

Let us now consider how local branching may be used in the case of the single vehicle routing problem with stochastic demands. One may start a local branching descent from any solution x^ν provided by the L-shaped algorithm. However, one needs an integer vector to create the first neighbourhood to explore. If x^ν is integer, then one may use the vector directly. If x^ν is continuous, then by rounding to the nearest integer each component of x^ν, one obtains an integer vector $\hat{x}^\nu$ which, considering the Hamming distance, is the nearest to x^ν. In this case, vector $\hat{x}^\nu$ will define the starting point of the local branching descent. There is an important point to be made when one considers vector $\hat{x}^\nu$:

Proposition 4.4.1. *Let $\hat{x}$ represent a feasible route, then constraint $\Delta(x, \hat{x}) \geq 4$ is a valid inequality for problem (4.13)-(4.18) in the case of the single vehicle routing problem with stochastic demands.*

Proof. The closest feasible solutions to $\hat{x}$ are in the $2-opt$ neighbourhood. The $2-opt$ neighbourhood of solution $\hat{x}$ is obtained by deleting two edges from route $\hat{x}$ and replacing them by two other edges that were not in route $\hat{x}$ in order to form a new feasible solution. To do so, one must have at least two (i,j) for which $\hat{x}_{ij} = 1 \rightarrow 0$ and at least two $(i,j) \in E$ for which $\hat{x}_{ij} = 0 \rightarrow 1$. Therefore, all solution in the $2-opt$ neighbourhood of $\hat{x}$ lie at a Hamming distance of 4 from $\hat{x}$. □

By rounding components of x^{ν}, one may obtain a feasible or infeasible integer vector $\hat{x}^{\nu}$. If $\hat{x}^{\nu}$ is feasible and $\Delta(x^{\nu}, \hat{x}^{\nu}) < 4$, then by adding constraint $\Delta(x, \hat{x}) \geq 4$ one is able to cut off solution x^{ν} and continue the separation process. Therefore, a local branching descent need not be performed if the previous condition is verified. Only when $\hat{x}^{\nu}$ is either infeasible or feasible but $\Delta(x^{\nu}, \hat{x}^{\nu}) \geq 4$, is a local branching descent applied using as starting point vector $\hat{x}^{\nu}$. One must now determine how to perform the local branching descent. There are three points to be adressed, one must first decide how to calculate the lower bounds needed, then one must find in each of the subproblems explored a new integer vector that will be used in the branching scheme and finally, one must decide how the branching decision will be taken.

The *current problem* (4.13)-(4.18) is used in order to obtain the lower bounds for the local branching subproblems P_n $(n = 1, \ldots, m)$ that are explored. One may use problem (4.13)-(4.18) at some point in the search tree of the 0-1 integer L-shaped algorithm to find local lower bounds. By doing so, the valid inequalities obtained can only be used locally on the subproblem considered. Another possibility, is to use problem (4.13)-(4.18) directly, making it possible to create general valid inequalities. Both strategies will be applied to obtain the computational results in the next section. To obtain values $\overline{\Theta}_n$ for $n = 1, \ldots, m$, the integrality constraints are temporarily reintroduced in (4.13)-(4.18). The separation algorithms are then used to improve the lower bounds in the local branching subregions that are explored. By doing so, one is using the L-shaped algorithm itself to increase the values obtained for $\overline{\Theta}_n$, $n = 1, \ldots, m$. To limit the effort spent improving the lower bounds, a maximum number of calls to the separation algorithms is imposed. The maximum number of calls was set to three to obtain the

computational results in the next section. If for a particular subregion thus explored, the associated subproblem P_n is solved exactly, then one obtains a feasible route x_n that is optimal in the subregion defined by P_n (i.e., $\mathcal{Q}(x_n) \leq \mathcal{Q}(x)\ \forall x \in P_n$). In this case, and one may set $\overline{\Theta}_n = \mathcal{Q}(x_n)$. Otherwise, the lower bound $\overline{\Theta}_n$ is simply derived from $\underline{z}_n = c^\top x_n + \Theta_n$. By proceeding in this way, one also obtains the integer vector (i.e., x_n) needed for the branching scheme.

Originally in [50], local branching is used to search for different optimality cuts, using as starting point the optimal solution to the Benders master problem. By doing so, one can simultaneously work on improving the upper and lower bounds used in the Benders decomposition algorithm. By using local branching in the L-shaped algorithm, one may hope to obtain the same benefits. Since the local branching descent may provide new feasible solutions, one can find better upper bounds. By adding the local branching valid inequalities to the subproblems generated by the L-shaped algorithm, one can improve the general lower bound. However, to keep the procedure effective, one must control the effort spent generating the local branching descent. It was found that to better explore the feasible region of problems (4.13)-(4.18), one should always branch using solutions x_n $(n = 1, \ldots, m)$ regardless of the quality of the lower bounds obtained. Therefore, the size of the neighbourhood is fixed to a certain value κ which remains unchanged. Furthermore, the effort spent on each descent will also be controled by fixing value m as to be the maximum depth allowed.

One can now conclude this section by describing how all these valid inequalities are generated at the separation step of the 0-1 integer L-shaped algorithm. Step **(2.3)** starts by a search of violated subtour elimination constraints. The heuristics that are used to find these cuts, are the ones that Lysgaard and *al.* [43] present for capacity inequalities. These procedures were obtained from the CVRPSEP package proposed by the authors. Subtour elimination constraints are added as long as the heuristics are able to find them. When these heuristics fail, separation continues by searching for violated inequalities of type (4.49). If a violated cut is found, one restarts the separation step by searching for subtour elimination constraints. As is shown by Hjorring and Holt [32], inequalities of type (4.49) are necessary to solve efficiently the single vehicle routing problem with

stochastic demands. Since these inequalities provide useful information concerning the value of recourse, by using them, one prevents against the risk of enumeration. When one is unable to cut the current solution using either constraints (4.47) or inequalities of type (4.49), then a local branching descent is performed using fixed values κ and m. Local branching is used once to tighten the lower bound associated with the active subproblem. At the end of the separation step, the local branching system (4.19)-(4.25) is added to (4.13)-(4.18) and the subproblem is solved once using the integrality constraints. In doing so, we obtain the lower bound $c^\top x^\nu + \Theta^\nu$. The solution process can then proceed as before.

4.5 Computational results

To properly test the different ideas proposed in this paper a series of test problems were generated. The problem generator used follows the same principles as the one proposed in [32]. Therefore, the graph vertices were generated in a $[0, 100]^2$ square following uniform distributions and the cost matrix was then set to be the Euclidean distances between vertices. Each customer was assigned an average demand following a $[1, 10]$ uniform distribution and the standard deviation was set to be 30% of the mean. As in [32], problems of sizes $n = 20,\ 30, \ldots,\ 90$ were created. For each size, five instances were generated for which $\overline{f} = 95\%,\ 97.5\%, \ldots,\ 110\%$ (where $\overline{f} = \sum_{i=1}^{n} \mu_i / D$), for a total of 280 instances.

The first algorithm implemented is the standard L-shaped algorithm for which the partial route cuts of Hjorring and Holt [32] are added. This is the benchmark on which the implementations using local branching are compared. Using local branching, both cut generation strategies are implemented. The LB implementation refers to the case were cuts are generated locally. As for the LB1 implementation, it refers to the case were cuts are generated globally. Parameter m is fixed to three on all runs. As for the size of the local branching neighbourhoods, runs are made with $\kappa = 4,\ 6,\ 8$. All three algorithms are implemented using the branching package proposed by Gendron and *al.* [25]. All experiments are performed on a 2.4 GHz AMD Opteron 64 bit processor. A

maximum time of 1200 seconds is imposed on all runs (as was also imposed by Hjorring and Holt [32]) and the optimality gap considered is $\epsilon = 1\%$.

To measure the basic tradeoff in using local branching cuts, the standard implementation is first compared to the LB implementation, for which $\kappa = 4$ (LB-4). The average solution times and average gap are reported for these two implementations in Table 4.1. Results are aggregated in the following way : (< 60) refers to those instances that are solved in less than 60 seconds by the standard algorithm, ($[60, 1200]$) to those that take between 60 and 1200 seconds and (> 1200) includes those instances that the standard implementation is unable to solve in the maximum alotted time. As for the undefined (*und.*) line, it refers to instances for which the standard algorithm is unable to find a feasible solution after 1200 seconds of computation time. Whenever LB-4 is either able to solve, or obtain a feasible solution to, an instance for which the standard algorithm fails, then the (*sol.*) line is used. The (*not sol.*) line refers to those instances for which both implementations failed. All times reported are in seconds. Whenever the implementations are unable to solve the instances in the maximum time allowed, then the average gap obtained is given.

When comparing the standard and LB-4 implementations, one first observes that the results for LB-4 are significantly better. The L-shaped algorithm is either faster or obtains better results when one uses the local branching cuts. The only exception is the case of the easy instances (< 60) for the problems of size $n = 60$, for which the average time of LB-4 is 6.71 seconds compared to 5.69 seconds for the standard implementation, and the problems of size $n = 90$, for which the times are 8.97 seconds compared to 6.14 seconds. In the case of the medium instances ($[60, 1200]$), which corresponds to a total of 30 problems, the sum of average times for the standard implementation is 2603.8 seconds compared to 183.38 seconds for LB-4. This corresponds to a reduction in solution times by a factor that is slightly over 14. Furthermore, 24 of the instances that the standard implementation is unable to solve in the maximum time allowed, are solved by LB-4. These instances are sometimes solved quite efficiently by LB-4, as is the case for problems of size $n = 80$, where five of the (> 1200) instances, for which the average gap obtained by the standard algorithm is 1.63%, are solved, on average,

n	Times	nb.	Standard	LB-4
20	< 60	35	0.15	0.1
30	< 60	31	2.58	0.83
	$[60, 1200]$	1	134.02	1.53
	> 1200 *sol.*	3	3.05%	359.27
40	< 60	30	4.93	1.79
	$[60, 1200]$	3	578.67	28.44
	> 1200 *sol.*	1	2.95%	136.32
	> 1200 *not sol.*	1	3.00%	1.37%
50	< 60	26	5.95	2.25
	$[60, 1200]$	2	578.67	8.92
	> 1200 *sol.*	2	2.15%	109.36
	> 1200 *not sol.*	5	2.82%	1.59%
60	< 60	16	5.69	6.71
	$[60, 1200]$	5	450.16	30.16
	> 1200 *sol.*	6	2.41%	528.05
	> 1200 *not sol.*	8	3.58%	1.93%
70	< 60	14	3.51	2.55
	$[60, 1200]$	6	200.4	13.48
	> 1200 *sol.*	3	1.90%	157.34
	> 1200 *not sol.*	10	3.78%	2.10%
	und. sol.	2	-	369.69
80	< 60	11	10.91	8.21
	$[60, 1200]$	9	392.99	64.15
	> 1200 *sol.*	5	1.63%	67.31
	> 1200 *not sol.*	6	2.33%	1.46%
	und. sol.	3	-	4.84%
	und. not sol.	1	-	-
90	< 60	13	6.14	8.97
	$[60, 1200]$	4	268.89	36.7
	> 1200 *sol.*	2	1.21%	595.47
	> 1200 *not sol.*	12	3.85%	2.59%
	und. sol.	1	-	2.02%
	und. not sol.	3	-	-

TAB. 4.1 – Solution times and average gap : Standard vs. LB-4

in a little more than one minute (67.31 seconds) by LB-4. Also, out of the 24 hard instances solved by LB-4, two are in the undefined (*und.*) category (i.e., the problems of size $n = 70$). Finally, when both implementations are unable to solve the problems, the average gap obtained with LB-4 is systematically lower than with the standard algorithm.

These results may be explained when one examines the separation algorithm used in the LB implementation. In all cases, the separation process starts by searching for violated subtour elimination constraints. At this point, one is trying to establish first stage feasible solutions. If the separation heuristics fail, then the algorithm turns to the recourse value to try to cut the current solution. Cuts of type (4.49) are used as much as possible to better express the recourse function. When one is no longer able to generate a cut based on a partial solution, then in the standard implementation, the algorithm must branch. However, in the LB implementation, a local branching descent is applied. The lower bounds derived from the neighbourhoods explored during the local branching descent tend to be tighter. Therefore, by using the local branching inequalities, one can better express the recourse value around a solution that the partial route inequalities were unable to cut. In this case, system (4.19)-(4.25) is used to complement the cuts of type (4.49) that were found for the subproblem being separated. The overall benefit of this approach is that the lower bounds obtained for each subproblem created by the algorithm are tighter. Furthermore, since the local branching search may provide new feasible solutions, one is also able to find better upper bounds at an earlier stage. In turn, both of these benefits tend to make the solution process much faster.

Let us now examine the results obtained by both cut strategies when one increases parameter κ. To do so, Tables 4.2 and 4.3 give us the number of instances aggregated according to the previous classification but for all implementations (standard, LB and LB1) and for all runs ($\kappa = 4,\ 6,\ 8$). If one sums the results reported for all lines (< 60) and ($[60, 1200]$), then one obtains the total number on instances that were solved before the time limit was reached for all implementations on all runs. When examining Table 4.2, one may see that the standard implementation was able to solve 204 instances compared to 230 for LB-4 and LB-6 and 231 for LB-8. Results are almost identical

n	Times	Standard	LB-4	LB-6	LB-8
20	< 60	35	35	35	35
30	< 60	31	32	33	32
	$[60, 1200]$	1	3	1	2
	> 1200	3	0	1	1
40	< 60	30	33	33	34
	$[60, 1200]$	3	1	1	0
	> 1200	2	1	1	1
50	< 60	26	29	29	29
	$[60, 1200]$	2	1	2	2
	> 1200	7	5	4	4
60	< 60	16	20	20	20
	$[60, 1200]$	5	7	8	7
	> 1200	14	8	7	8
70	< 60	14	22	21	21
	$[60, 1200]$	6	3	4	4
	> 1200	13	10	10	10
	und.	2	0	0	0
80	< 60	11	18	17	18
	$[60, 1200]$	9	7	8	7
	> 1200	11	9	10	10
	und.	4	1	0	0
90	< 60	13	16	16	16
	$[60, 1200]$	4	3	2	4
	> 1200	14	13	14	13
	und.	4	3	3	2

TAB. 4.2 – Instances solved : standard vs. LB

when one increases the value of κ in the case of LB. By generating the cuts locally, it seems that one does not need to search large neighbourhoods to obtain good results. With LB-4, one seems to obtain the best tradeoff between the quality of results and computational effort.

n	Times	Standard	LB1-4	LB1-6	LB1-8
20	< 60	35	35	35	35
30	< 60	31	32	31	31
	[60, 1200]	1	2	3	2
	> 1200	3	1	1	2
40	< 60	30	31	31	30
	[60, 1200]	3	2	3	4
	> 1200	2	2	1	1
50	< 60	26	25	27	26
	[60, 1200]	2	5	4	4
	> 1200	7	5	4	5
60	< 60	16	19	19	19
	[60, 1200]	5	3	3	3
	> 1200	14	13	13	13
70	< 60	14	19	17	19
	[60, 1200]	6	4	7	5
	> 1200	13	12	11	11
	und.	2	0	0	0
80	< 60	11	16	13	11
	[60, 1200]	9	9	12	13
	> 1200	11	9	9	10
	und.	4	1	1	1
90	< 60	13	15	15	12
	[60, 1200]	4	3	3	7
	> 1200	14	14	14	14
	und.	4	3	3	2

TAB. 4.3 – Instances solved : standard vs. LB1

In the case of LB1, the situation is different. When analyzing Table 4.3, one may see that 217 instances are solved by LB1-4, 223 by LB1-6 and 221 by LB1-8. When one increases the size of κ, one is able to solve more problems in the maximum time

allowed. In this case, the LB1-6 run seems to give the best results. By using larger neighbourhoods, the cuts will bound the recourse value for larger portions of the feasible region of the *current problem*. Therefore, since the LB1 implementation generates the cuts globally (cuts are reused), larger neighbourhoods tend to make the cuts useful on a larger number of subproblems explored in the search tree.

When comparing LB to LB1, one observes that by generating the cuts locally, one is able to solve a greater number of instances (230 for LB-4 compared to 223 for LB1-6). The explanation for this resides in the observation that by reusing the cuts, as is done in LB1, one makes the last subproblem processed in the separation procedure a lot harder to solve. Therefore, for a fixed maximum computation time, the LB1 implementation explores a smaller region of the search tree compared to the LB implementation. This proves to be a drawback on certain instances. However, when both strategies are compared to the standard algorithm, it is always preferable to apply the local branching cuts.

This section will now be concluded by analyzing how the results of both the LB and LB1 implementations vary for hard instances when the maximum time allowed is increased. To do so, a subset of hard instances was chosen (the hard instances for $n = 70,\ 80,\ 90$). Results obtained by LB-4 and LB1-6 will be compared when the maximum time allowed is increased to 2400, 3600, 4800 and 6000 seconds. In Table 4.4, results obtained on the 48 hard instances used are classified in three columns : the instances solved in the maximum time allowed (*sol.*), the instances that can not be solved but for which a feasible solution is obtained (*not sol.*) and the instances for which the algorithms are unable to find at least one feasible solution (*und.*). Both the number of instances and average results in seconds (for (*sol.*)), or percentage of gap (for (*not sol.*)) are given in Table 4.4.

Once again, the LB algorithm seems to outperform LB1. As the maximum computation time is increased, one may see that LB-4 is able to solve more instances compared to LB1. For 6000 seconds of computation time, LB-4 solves 18 instances compared to 12 for LB1-6. Furthermore, if one considers the (*not sol.*) column, one may see that

	LB			LB1		
Max. Time	*sol.*	*not sol.*	*und.*	*sol.*	*not sol.*	*und.*
1200	12/228.24	32/2.42%	4	10/275.04	34/3.00%	4
2400	15/595.33	31/2.49%	2	10/275.04	35/2.98%	3
3600	17/884.89	29/2.44%	2	12/751.01	34/2.93%	2
4800	18/1078.25	29/2.52%	1	12/751.01	34/2.87%	2
6000	18/1078.25	29/2.45%	1	12/751.01	34/2.84%	2

TAB. 4.4 – Results on hard instances : LB vs. LB1

the average gap obtained is slightly better for LB-4. Again, these results seem to show that the local branching cuts are better used locally to the subproblems explored by the L-shaped algorithm. The LB algorithm explores a larger portion of the search tree which seems to produce better results on the problems that are considered here.

4.6 Conclusion

In this paper, a new type of valid inequalities is proposed for the 0-1 integer L-shaped algorithm. These inequalities are based on the information gathered through the use of local branching descents. They provide a broader description on the value of recourse compared to the classical optimality cut. These inequalities can be implemented in the L-shaped algorithm using two different cut generation strategies (local and global). Numerical tests are conducted on a series of single vehicle routing problems with stochastic demands. Results show that there is a clear advantage in using these new inequalities. An interesting avenue of research would be to generalize the approach proposed here to the case of general vehicle routing problems with stochastic demands. It would also be interesting to find new separation strategies using these local branching cuts.

Chapitre 5

A Hybrid Monte Carlo Local Branching Algorithm for the Single Vehicle Routing Problem with Stochastic Demands

Au chapitre précédent nous avons présenté de nouvelles inégalités valides s'appliquant à l'algorithme « L-shaped » binaire. Ces inégalités ont été spécialisées au problème de tournées d'un véhicule avec demandes stochastiques. Ce faisant, nous avons obtenu un algorithme exact plus efficace pour le problème considéré. Cependant, certaines instances demeuraient difficiles à résoudre exactement en un temps de calcul raisonable. Ces instances ont donc motivé le développement d'une approche de résolution heuristique efficace.

Le problème de tournées d'un véhicule avec demandes stochastiques est défini comme étant le problème du voyageur de commerce, auquel une composante stochastique est ajoutée. Dans la version stochastique, le véhicule a une capacité limitée et les clients ont des demandes aléatoires qui ne sont pas connues à l'avance. Par conséquent, lorsqu'un client est visité, il est possible que la capacité résiduelle du véhicule ne soit

pas suffisante pour répondre aux besoins. Dans un tel cas, une livraison partielle est effectuée, et le véhicule fait un aller-retour entre le dépôt et le client afin de décharger (ou recharger) et compléter le service. Le déplacement supplémentaire pour effectuer l'aller-retour en cas de rupture représente le coût du recours associé au problème. Le problème de tournées d'un véhicule avec demandes stochastiques consiste donc à construire une tournée qui visite chacun des clients et qui minimise simultanément les coûts de déplacement et le coût total moyen du recours. Ces problèmes sont difficiles à résoudre, car il faut tenir compte de la complexité combinatoire du problème du voyageur de commerce à laquelle s'ajoute le coût supplémentaire stochastique des solutions. Les instances où le nombre de client est élevé et où les risques de rupture sont grands sont particulièrement difficiles à résoudre exactement.

L'heuristique proposée a été développée à partir d'une observation concernant le poids relatif des deux types de coûts. Les résultats numériques de l'article précédent ont démontré que, dans le cas d'instances raisonables, les coûts de déplacement associés aux tournées optimales sont plus importants que les coûts moyens du recours. Par conséquent, les tournées optimales stochastiques demeurent habituellement de bonnes solutions aux problèmes d'approximation déterministe, bien que les VSS (valeurs des solutions stochastiques) dans le contexte présent justifient l'utilisation de la version stochastique du modèle. L'heuristique effectue donc, de façon itérative, des phases de recherche autour de solutions réalisables dont les coûts de déplacement sont faibles. La procédure combine à la fois la recherche par séparation locale et la simulation Monte Carlo. La séparation locale permet de contrôler la complexité associée à l'exploration du domaine réalisable. Des principes de séparation locale sont utilisés à la fois pour identifier des solutions différentes dont les coûts de déplacement sont faibles et pour effectuer des phases de recherche permettant de trouver de bonnes solutions stochastiques. La simulation Monte Carlo est utilisée, quant à elle, afin d'approximer le recours pour la problématique considérée. En contrôlant simultanément les complexités combinatoire et stochastique, nous obtenons une procédure efficace et robuste pour le problème de tournées d'un véhicule avec demandes stochastiques.

A Hybrid Monte Carlo Local Branching Algorithm for the Single Vehicle Routing Problem with Stochastic Demands

Walter Rei
Université de Montréal

Michel Gendreau
Université de Montréal

Patrick Soriano
HEC Montréal

November 2006

Abstract

We present a new algorithm that uses both local branching and Monte Carlo sampling in a multi-descent search strategy for solving 0-1 integer stochastic programming problems. This procedure is applied to the single vehicle routing problem with stochastic demands. Computational results show the usefulness of this new approach to solve hard instances of the problem.

Keywords. Local branching, Monte Carlo sampling, stochastic vehicle routing problems

5.1 Introduction

There has been a great deal of research done on vehicle routing problems (VRP). Both exact algorithms and metaheuristic procedures have been proposed for deterministic cases of the VRP (see [63]). However, in practice, one rarely has access to perfect information concerning the parameters of a problem. Therefore, in recent years, stochastic versions have been considered where certain parameters of the VRP are modeled by random variables. By solving stochastic routing problems, one can obtain significantly better solutions whenever there is uncertainty in the situation being modeled. These problems are therefore very interesting for real life applications but they are unfortunately notoriously hard to solve.

In this paper, the problem that is studied is the single vehicle routing problem with stochastic demands (SVRPSD). The SVRPSD is defined as follows : let $G(V, E)$ be an undirected graph, where $V = \{v_1, \ldots, v_N\}$ is a set of vertices and $E = \{(v_i, v_j) : v_i, v_j \in V, i < j\}$ is a set of edges. Defined on E is a symmetric matrix $C = [c_{ij}]$ that corresponds to the travel costs between vertices. Vertex v_1 represents a depot from which the vehicle must start and finish its route. If one searches for a route that visits all vertices once and minimizes the total travel cost, then one is in fact solving the well known travelling salesman problem (TSP). The TSP is an NP-hard problem which has been extensively studied, see [29].

The SVRPSD is obtained by adding a particular component to the classical TSP problem. Let us suppose that the vehicle has a limited capacity D and that each vertex $j \in V \setminus \{v_1\}$ corresponds to a customer that has a nonnegative demand ξ_j that is stochastic. Let us also make the following hypothesis : $\forall j \in V \setminus \{v_1\}$, demand ξ_j only becomes known when the vehicle arrives at the location of customer j. In this case, whenever a customer is visited, the residual capacity of the vehicle may not suffice to fulfill the observed demand. When such a failure occurs, one must take a recourse action that will entail an extra cost.

The model used here is based on the classical two stage stochastic programming

formulation. In the first stage one constructs a route that visists all customers once. In the second stage, the determined route is followed and demands become known. When a failure occurs, partial delivery is performed and the recourse action taken is to return to the depot, to stock up (or to unload), and then go back to the customer where failure occured to finish the delivery and continue the route. In this case, the extra cost incurred is the traveling cost to the depot and the return cost to the customer location. The optimization problem consists of finding a route that minimizes the sum of both the total travel cost as well as the expected cost of recourse. For a complete description of the models that can be used as well as the properties associated with them, the reader is refered to the papers of Dror and *al.* [18, 17].

What makes the SVRPSD a hard problem to solve is the combination of both the inherent complexity of the TSP and the stochastic cost associated with the feasible solutions. If one defines the expected filling rate of the vehicle as : $\overline{f} = \sum_{j=1}^{N} \mathbf{E}[\xi_j]/D$, then as $\overline{f}$ increases so does the risk of failures. Instances where both the number of customers and $\overline{f}$ are large are very hard to solve optimaly (see [51]). The main contribution of this paper is to propose a new heuristic algorithm that solves efficiently such hard instances. The heuristic developped is a hybrid method that uses both local branching and Monte Carlo sampling. Certain characteristics of the SVRPSD will be exploited in the implementation of the method. However, the methodology that is proposed is quite general and can be applied to other stochastic problems.

The remainder of this paper is divided as follows. In section 2, the model used for the SVRPSD is presented as well as a brief description of the solution methods to solve it. In section 3, various Monte Carlo methods that have been developped to solve stochastic programming problems are reviewed. Section 4 includes a description of the heuristic that is proposed in this paper. This is followed by the computational results obtained on the SVRPSD in section 5. Finally, section 6 presents some concluding remarks.

5.2 The SVRPSD

The model for the SVRPSD is defined as follows :

$$\text{Min} \quad \sum_{i<j} c_{ij}x_{ij} + \mathcal{Q}(x) \tag{5.1}$$

$$\text{s.t.} \quad \sum_{j=2}^{N} x_{1j} = 2, \tag{5.2}$$

$$\sum_{i<k} x_{ik} + \sum_{j>k} x_{kj} = 2, \ k = 2, \ldots, N, \tag{5.3}$$

$$\sum_{i\in S}\sum_{\substack{j\notin S \\ j>i}} x_{ij} + \sum_{i\notin S}\sum_{\substack{j\in S \\ j>i}} x_{ij} \geq 2, \ S \subseteq V, \ |S| \geq 3, \tag{5.4}$$

$$x_{ij} \in \{0,1\}, \ 1 \leq i < j \leq N. \tag{5.5}$$

Function $\mathcal{Q}(x)$ in (5.1) is the recourse function which represents the expected cost of recourse. It should be specified that under some assumptions, given a feasible route x, function $\mathcal{Q}(x)$ can be easily computed, as described in the paper of Laporte and *al.* [40]. Constraints (5.2) and (5.3) are used to make sure that the route starts and ends at the depot and that each customer is visited once. Inequalities (5.4) are the subtour elimination constraints. Finally, constraints (5.5) impose the integrality restriction on the variables of the problem.

The principal solution approach that is used to solve problem (5.1)-(5.5) is based on the 0-1 integer L-shaped algorithm presented by Laporte and Louveaux [39]. Following this approach, Benders decomposition is applied to the problem. The recourse function $\mathcal{Q}(x)$ is replaced in the objective by variable Θ which is then bounded by a series of optimality cuts (or any other lower bounding functionnals). In addition, constraints (5.4) and (5.5) are relaxed from the model. Optimality cuts as well as constraints (5.4) are then gradually added to the relaxed problem following a branch and cut framework.

Gendreau and *al.* [23] were the first to apply the standard L-shaped algorithm to the single vehicle routing problem with stochastic demands. In 1999, Hjorring and Holt [32] proposed a new type of cut that uses information taken from partial routes. A partial route is made up of three sets. Using the notation proposed by Laporte and *al.* [40], let

us first define the two ordered sets $S = \{v_1, \ldots, v_s\}$ and $T = \{v_1, \ldots, v_t\}$. Sets S and T must respect the following condition : $S \cap T = \{v_1\}$. Let us now define a third set $U = V \setminus ((S \setminus \{v_s\}) \cup (T \setminus \{v_t\}))$. One easily sees that $S \cap U = \{v_s\}$ and $T \cap U = \{v_t\}$. Therefore, a partial route is made up of the two vectors $(v_1, \ldots, v_s)$ and $(v_t, \ldots, v_1)$ that define the beginning and end portions of the route and of set U, which contains all vertices that are not yet ordered. If $(v_i, v_j) \in S$ or T refers to the case where v_i and v_j are consecutive in S or T, then let $W(x) = \sum_{(v_i,v_j)\in S} x_{ij} + \sum_{(v_i,v_j)\in T} x_{ij} + \sum_{v_i,v_j \in U} x_{ij} - |V| + 1$. If Q is a lower bound on the value of recourse for the partial route and L is a general lower bound on $\mathcal{Q}(x)$, then the following inequality is valid for problem (5.1)-(5.5) :

$$\Theta \geq L + (Q - L)W(x). \tag{5.6}$$

In [32], the authors present a lower bounding technique to obtain value Q. Laporte and *al.* [40] generalize (5.6) to the case of multiple vehicles. They also develop a new technique to obtain a better general lower bound L.

Recently, Rei and *al.* [51] proposed a new type of valid inequalities that apply to the case of the 0-1 integer L-shaped algorithm. These inequalities are based on local branching descents and are applied to problem (5.1)-(5.5). The implementation of the 0-1 integer L-shaped algorithm for the SVRPSD proposed in [51] produces the best results for the case where demands are Normal random variables (i.e., $\xi_j \sim N(\mu_j, \sigma_j)$) and where all random variables are independently distributed. However, instances where both the filling rate and the number of customers are large still present a tremendous challenge which justifies the development of efficient heuristics for this problem.

Heuristics have been proposed for related versions of problem (5.1)-(5.5). Gendreau and *al.* [24] proposed a tabu search algorithm for routing problems where customers and demands are stochastic. In 2000, Yang and *al.* [70] have proposed a series of heuristics for routing problems with stochastic demands for which restocking is considered. Restocking allows the vehicle to return to the depot before visiting the next customer on the route. By doing so, one may prevent failures. Recently, Bianchi and *al.* [5] have also implemented a series of metaheuristics for stochastic routing problems that allow restocking. Secomandi [56, 57] proposes neuro-dynamic programming algorithms for the

case where re-optimization is applied to the SVRPSD. In this case, as demands become known, the ordering of the customers that have not yet been visited may be changed depending on the state of the situation. Finally, Chepuri and Hommem-De-Mello [11] solve an alternate formulation of the SVRPSD using the cross-entropy method. The alternate formulation considered allows the possibility that certain customers may not be serviced by the vehicle. However, a penalty function is used to dissuade such situations.

5.3 Monte Carlo sampling in stochastic programming

In this section a general presentation of how Monte Carlo sampling has been used in stochastic programming is provided. Note however that this section does not aim at being exhaustive but focuses on presenting the principal results and solution approaches within this field. Let us first define the classical stochastic programming problem with fixed recourse as follows :

$$\text{Min} \quad c^\top x + \mathcal{Q}(x) \tag{5.7}$$

$$\text{s.t.} \quad Ax = b \tag{5.8}$$

$$x \in \overline{X}, \tag{5.9}$$

where $\mathcal{Q}(x) = \mathbf{E}_\xi[Q(x, \xi(\omega))]$ and $Q(x, \xi(\omega)) = Min_y\{q(\omega)^\top y \mid Wy = h(\omega) - T(\omega)x,\ y \in \overline{Y}\}$, $\forall \omega \in \Omega$. Monte Carlo sampling is mainly used in two different ways to solve problem (5.7)-(5.9). As presented by Linderoth and *al.* [41] sampling is either used in an interior fashion or in an exterior fashion. When sampling is used in an interior fashion, one is actually trying to solve directly problem (5.7)-(5.9) but whenever the algorithm being used requires information concerning the recourse function, then sampling is applied to approximate this information. In the exterior approach, instead of trying to solve the stochastic problem directly, one uses sampling beforehand as a way to approximate the recourse function. One can then apply any adapted deterministic optimization algorithm to solve the approximated problem.

The first type of methods that have been proposed using the interior approach are based on the L-shaped algorithm presented by Van Slyke and Wets [66] for the

case of continuous stochastic programming problems with fixed recourse. The first to introduce sampling in the L-shaped algorithm were Dantzig and Glynn [16]. The algorithm proposed in [16] uses sampling to estimate the cuts needed in the solution process. Samples are determined so as to obtain a given confidence level. To improve the convergence rate, importance sampling is used for the generation of the scenarios. Since the size of the samples needed can become quite large, the authors also propose a parallel implementation of the method to reduce solution times.

Another method that uses sampling in an L-shaped based algorithm is the stochastic decomposition approach proposed by Higle and Sen [31] for the case of problems with complete recourse (i.e., $Q(x, \xi(\omega)) < \infty$ regardless of x and $\forall \omega \in \Omega$). The idea behind stochastic decomposition is to use larger samples to produce cuts as the number of iterations of the L-shaped algorithm increases. At iteration ν, the algorithm uses ν independently generated samples to produce the next optimality cut. Previously generated cuts are updated in such a way that they become redundant and are subsequently dropped as the algorithm proceeds. Details concerning the convergence and implementation of this approach are provided in [31]. The authors also elaborate on the use of stopping rules, which include both error bound estimates and tests on optimality conditions.

Finally, stochastic quasi-gradient methods, see Ermoliev [19], have also applied Monte Carlo sampling in an interior fashion. In this case, sampling is used to produce a subgradient or quasi-gradient for which a descent direction may be obtained. The algorithm proceeds by taking a step in the direction that is defined. A projection is then applied onto the set of feasible first stage solutions.

Techniques that use Monte Carlo sampling in an exterior fashion are generally based on the use of sample average approximations of the recourse function. Let $X = \{x \mid Ax = b,\ x \in \overline{X}\}$ be the set of first stage constraints, then one may rewrite problem (5.7)-(5.9) as : $\min_{x \in X} f(x)$ where $f(x) = \mathbf{E}_\xi[c^\top x + Q(x, \xi(\omega))] = c^\top x + \mathbf{E}_\xi[Q(x, \xi(\omega))]$. If $\{\omega^1, \ldots, \omega^n\}$ is a subset of randomly generated events of Ω, then function $\widehat{f}_n(x) = c^\top x + \frac{1}{n} \sum_{i=1}^{n} Q(x, \xi(\omega^i))$ is a sample average approximation of $f(x)$. One may now define

the approximating problem in the following way : $\min_{x \in X} \widehat{f}_n(x)$.

It is shown in Mak and *al.* [46] that if one considers the average value of the approximating problem over all possible samples, then one obtains a lower bound on the optimal value of problem (5.7)-(5.9), that is : $\mathbf{E}\left[\min_{x \in X} \widehat{f}_n(x)\right] \leq \min_{x \in X} f(x)$. In [46], the same type of reasoning is also applied to the case where one is trying to compute the value of a first stage feasible solution. Let $\tilde{x}$ be a feasible first stage solution, then one may show that $\mathbf{E}\left[\widehat{f}_n(\tilde{x})\right] \geq f(\tilde{x})$. Therefore, by using unbiased estimators for $\mathbf{E}\left[\min_{x \in X} \widehat{f}_n(x)\right]$ and for $\mathbf{E}\left[\widehat{f}_n(\tilde{x})\right]$, one can construct confidence intervals on the optimal gap associated with solution $\tilde{x}$. Unbiased estimators can be obtained by using batches of subsets $\{\omega^1, \ldots, \omega^n\}$. Let $\widehat{f}_n^j$ be the jth sample average approximation function using a randomly generated subset of size n and let $\widehat{v}_n^j = \min_{x \in X} \widehat{f}_n^j(x)$, for $j = 1, \ldots, m$. Then $L_m^n = \frac{1}{m} \sum_{j=1}^{m} \widehat{v}_n^j$ and $U_m^n = \frac{1}{m} \sum_{j=1}^{m} \widehat{f}_n^j(\tilde{x})$ can be used to estimate the gap associated with $\tilde{x}$. In [46], some variance reduction techniques are also presented.

Under certain conditions, if $\hat{x}_n$ is an optimal solution to problem $\min_{x \in X} \widehat{f}_n(x)$, then it can be shown that $\hat{x}_n$ converges with probability 1 to the set of optimal solutions to (5.7)-(5.9) as $n \to \infty$. Furthermore, when the probability distribution of ξ is discrete, given some assumptions, Shapiro and Homem-De-Mello [60] show that $\hat{x}_n$ is an exact optimal solution to (5.7)-(5.9) for n large enough. The authors also demonstrate that the probability associated with the event of $\hat{x}_n$ not being an optimal solution to (5.7)-(5.9) tends to zero exponentially fast as $n \to \infty$. Using these results, Kleywegt and *al.* [37] elaborate the sample average approximation (or SAA) method.

The SAA method randomly generates batches of samples of random events and then solves the approximating problems. Each solution obtained is an approximation of the optimal solution to the original stochastic problem. Estimates on the optimal gap using bounds L_m^n and U_m^n are then generated to obtain a stopping criteria. Value n may be increased if either the gap or the variance of the gap estimator is to large. In [37], the authors also discuss the use of postprocessing procedures that provide some guarantees as to the quality of the solution chosen by the algorithm. The SAA method was adapted for the case of stochastic programs with integer recourse by Ahmed and Shapiro [1].

Recently, Linderoth and *al.* [41] have produced a series of numerical experiments using the SAA method which show the usefulness of the approach.

5.4 Monte Carlo local branching hybrid algorithm

The SAA algorithm has been successfully applied to obtain good quality solutions for a variety of stochastic problems for which direct solution approaches are inefficient (see [67], [53] and [41]). However, one is not always able to solve efficiently the approximating problems needed for the SAA approach. This situation has been observed in the case of hard istances of the SVRPSD. In this section, a heuristic that uses both local branching and Monte Carlo sampling will be presented to obtain good quality solutions even if the approximating problems obtained after sampling are still too difficult to solve in a reasonable time. This section will be divided in two subsections : the first will include a presentation of the local branching methodology, in the second subsection, a description of the solution approach using Monte Carlo sampling and local branching will be provided.

5.4.1 Local branching

The local branching solution approach was introduced by Fischetti and Lodi [20] as a way to solve hard mixed integer problems. The idea behind this method is to take advantage of the efficiency of generic solvers, such as CPLEX, for solving small integer 0-1 problems. Therefore, one can divide the feasible space of a problem into a series of smaller subregions and then use a generic solver to explore each of the subregions thus created.

To better illustrate this approach, let us apply local branching to the case of problem (5.7)-(5.9). To do so, let us first consider that problem (5.7)-(5.9) has binary first stage variables. Let us also suppose that the stochastic problems to be solved are such that all feasible first stage solutions are also feasible in the second stage (i.e., relative complete recourse). In this case, if vector x is of size n_1, then the set of first stage constraints

may be defined in the following way : $X = \{x \mid Ax = b,\ x \in \overline{X} \cap \{0,1\}^{n_1}\}$. Again, if $f(x) = c^\top x + \mathcal{Q}(x)$, then problem (5.7)-(5.9) becomes :

$$\text{Min} \quad f(x) \tag{5.10}$$

$$\text{s.t.} \quad x \in X. \tag{5.11}$$

Let x^0 be a vector of 0-1 values such that $x^0 \in X$. Using x^0, let function : $\Delta(x, x^0) = \sum_{j \in S_0} (1 - x_j) + \sum_{j \in N_1 \setminus S_0} x_j$, where $N_1 = \{1, \ldots, n_1\}$ and $S_0 = \{j \in N_1 \mid x_j^0 = 1\}$, define the Hamming distance relative to x^0. Using function $\Delta(x, x^0)$ and a fixed integer value κ, one may divide problem (5.10)-(5.11) into two subproblems : the first having first stage feasible region $\{x \mid x \in X,\ \Delta(x, x^0) \leq \kappa\}$ and the second having $\{x \mid x \in X,\ \Delta(x, x^0) \geq \kappa + 1\}$. When κ is fixed to an appropriate (small) value, constraint $\Delta(x, x^0) \leq \kappa$ can considerably reduce the size of the feasible region of problem (5.10)-(5.11). Therefore, one can use an adapted generic solver to solve this subproblem efficiently. The subregion defined by $\Delta(x, x^0) \geq \kappa + 1$ is left for further exploration.

Let us now consider two finite index sets I^ν and J^ν such that $x^k \in X, \forall k \in I^\nu \cup J^\nu$. If x^ν is a feasible first stage solution such that $\nu \notin I^\nu$ and $\kappa_i, \forall i \in I^\nu \cup \{\nu\}$, is a series of fixed integer values, then let us define the following two subproblems :

$$\begin{array}{llll} (P_\nu) & Min. & f(x) & \\ & \text{s.t} & \Delta(x, x^j) \geq 1, j \in J^\nu & \\ & & \Delta(x, x^i) \geq \kappa_i, i \in I^\nu & \\ & & \Delta(x, x^\nu) \leq \kappa & \\ & & x \in X & \end{array} \qquad \begin{array}{llll} (\overline{P}_\nu) & Min. & f(x) & \\ & \text{s.t} & \Delta(x, x^j) \geq 1, j \in J^\nu & \\ & & \Delta(x, x^i) \geq \kappa_i, i \in I^\nu & \\ & & \Delta(x, x^\nu) \geq \kappa + 1 & \\ & & x \in X. & \end{array}$$

The local branching algorithm proceeds by solving subproblem P_ν using the generic solver. Subproblem P_ν is either feasible, in which case one obtains a solution $x^{\nu+1}$, or infeasible. If one obtains $x^{\nu+1}$, then either $f(x^{\nu+1}) < f(x^\nu)$ or $f(x^{\nu+1}) \geq f(x^\nu)$. If $f(x^{\nu+1}) < f(x^\nu)$ then the algorithm sets $\kappa_\nu = \kappa + 1$, $I^{\nu+1} = I^\nu \cup \{\nu\}$ and $J^{\nu+1} = J^\nu$. Constraint $\Delta(x, x^\nu) \leq \kappa$ is replaced by $\Delta(x, x^\nu) \geq \kappa_\nu$, which gives us subproblem $\overline{P}_\nu$. Following the same branching scheme, solution $x^{\nu+1}$ is then used to separate the feasible region of $\overline{P}_\nu$, thus creating subproblems $P_{\nu+1}$ and $\overline{P}_{\nu+1}$. At this point, $P_{\nu+1}$

becomes the next subproblem to be solved. In the case where $f(x^{\nu+1}) \geq f(x^{\nu})$ or P_{ν} is infeasible, a diversification procedure is applied. The diversification procedure follows the principle that in order to obtain a better solution (or a feasible subproblem), then the feasible region of P_{ν} must be increased. Therefore, if $f(x^{\nu+1}) \geq f(x^{\nu})$, then constraint $\Delta(x, x^{\nu+1}) \geq 1$ is added to the subproblem and $J^{\nu+1} = J^{\nu} \cup \{\nu + 1\}$. By doing so, one eliminates from further consideration a solution $x^{\nu+1}$ whose value is no better than that of x^{ν}. In order to increase the size of the current subproblem feasible region, constraint $\Delta(x, x^{\nu}) \leq \kappa$ is replaced by $\Delta(x, x^{\nu}) \leq \kappa + \lceil \frac{\kappa}{2} \rceil$. By fixing $I^{\nu+1} = I^{\nu}$, one obtains $P_{\nu+1}$, which will be the next subproblem to be solved in the search process.

It should be specified that the branching decision may be applied using a different criteria then the one that has been evoked. Furthermore, for the diversification strategy, one may also use a different increase in the update of constraint $\Delta(x, x^{\nu}) \leq \kappa$. Local branching offers a general search context that one may adapt to the type of problem being solved. In [20], the authors impose a time limit for the solution of the subproblems. A series of diversification mechanisms derived from local search metaheuristics are also proposed. For the purpose of this paper, we will simply define a local branching descent as being a series of subproblems P_0, P_1, ..., that are solved to optimality or until a specified time limit is reached. The structure of each descent will be described in the next subsection.

5.4.2 Monte Carlo sampling and local branching

Monte Carlo sampling can be used to approximate the recourse function in (5.7)-(5.9). In doing so, one alleviates the stochastic complexity of the problem. By using local branching to explore X, one is able to control the combinatorial complexity associated with the first stage of problem (5.7)-(5.9). We will now show how these strategies may be used in a coordinated fashion creating what we will refer to as a multi-descent algorithm for the SVRPSD. To do so, we will first explain the multi-descent scheme and then describe the local branching descent structure used.

Multi-descent scheme

For the moment, let us suppose that one is able to solve efficiently local branching subproblems whitout resorting to sampling. Let us also consider the mean value problem (MVP), or expected value problem as in [7], associated with (5.7)-(5.9). The MVP problem is obtained by taking the random parameters of the stochastic problem and replacing them by their mean values. Using the formulation introduced in the previous sections, the MVP problem can be stated as follows :

$$\text{Min} \quad c^\top x + Q(x, \overline{\xi}) \tag{5.12}$$

$$\text{s.t.} \quad x \in X, \tag{5.13}$$

where $\overline{\xi} = \mathbf{E}[\xi]$. If $\overline{x}$ is an optimal solution to (5.12)-(5.13) and $x^\star$ is an optimal solution to the stochastic problem (5.10)-(5.11), then it is a well known result that $f(x^\star) \leq f(\overline{x})$ (see [7]). Actually, $f(\overline{x}) - f(x^\star)$ defines the value of the stochastic solution (VSS), which can be arbitrarily small or large depending on the problem considered.

In the case of routing problems, Louveaux [42] showed the importance of using the stochastic formulation when one considers the VSS. Although the VSS may be large, there is an important point to be made concerning the relative weight of the first stage objective function $c^\top x$ versus the recourse function $\mathcal{Q}(x)$ in the case of the SVRPSD. A problem where the expected filling rate $\overline{f}$ is small is in general easier to solve because it resembles the TSP. In this case, the MVP (5.12)-(5.13), or simply the TSP ($\min_{x \in X} c^\top x$), offers a good approximation for the original problem (5.10)-(5.11). As $\overline{f}$ increases so does the risk of failures and $\overline{x}$ becomes a potentially bad route when considering the stochastic formulation. However, $x^\star$ usually remains a good solution for the MVP (or the TSP). The reason for this is that, with the exception of extreme cases, the travel cost of a route ($c^\top x$) generally outweighs the value of recourse ($\mathcal{Q}(x)$). Therefore, a route for which the travel cost is high will unlikely be optimal even if the recourse value is small. The main idea behind the algorithm proposed in this paper will be to use as starting point solution $\overline{x}$, or any other route whose travel cost is low, and then try to close the gap to obtain $x^\star$.

If one defines x^0 as a feasible solution to the stochastic problem, then let P_0^k, P_1^k, ... , $P_{l_k}^k$ be the kth finite local branching descent starting from x^0. Let us suppose that only solution x^0 is eliminated using the Hamming distance function in problem (5.10)-(5.11). In this case, the last subproblem solved (i.e. $P_{l_k}^k$) is :

$$\text{Min} \quad f(x) \tag{5.14}$$

$$\text{s.t.} \quad \Delta(x, x^0) \geq 1 \tag{5.15}$$

$$\Delta(x, x^i) \geq \kappa_i, i \in I^{l_k} \tag{5.16}$$

$$\Delta(x, x^{l_k}) \leq \kappa_{l_k} \tag{5.17}$$

$$x \in X. \tag{5.18}$$

Since a different solution is used each time the branching decision is taken in a local branching descent, then from P_0^k, ... , $P_{l_k}^k$, one obtains at least l_k different first stage solutions.

From a multi-descent point of vue, one needs a feasible first stage solution to start a new descent. If one considers the kth local branching descent, then solutions $x^1, \ldots, x^{l_k}$, are all possible starting points. Using x^1, ... , x^{l_k}, the strategy that was chosen was to identify the best solution found in the last descent, $x^{k^\star} \in \arg\min\{f(x^i) \mid i = 1, \ldots, l_k\}$, add constraint $\Delta(x, x^{k^\star}) \geq 1$ to problem (5.10)-(5.11) and then use $x^{k^\star}$ as the new starting point of descent $k+1$. It should be noted that, in the case of the SVRPSD, $x^{k^\star}$ represents a feasible route. Therefore, constraint $\Delta(x, x^{k^\star}) \geq 1$ may be replaced by $\Delta(x, x^{k^\star}) \geq 4$, since all other feasible routes lie at a Hamming distance of at least four from $x^{k^\star}$ (see [51]). This type of descent will be refered to as a base descent. Since constraint $\Delta(x, x^{k^\star}) \geq 1$ is added to problem (5.10)-(5.11), the best solution that one finds in the $k+1$th descent will be different from the ones obtained in the previous k descents. Furthermore, since only solution $x^{k^\star}$ is eliminated, the algorithm can always come back and explore similar neighbourhoods from descent k to $k+1$. Base descents enable the algorithm to intensify the search around solutions that are found to be locally good. The drawback of this strategy is that if one eliminates all good feasible solutions in a certain vicinity of X or if one is exploring uninteresting neighbourhoods, then by only applying base descents, the procedure can take too long to reach different

subregions.

To counter this potential problem, another strategy, using the local branching constraints, was applied in the multi-descent approach. If one considers descent k, then one may be satisfied by the extent of the exploration carried out in the subregions defined by subproblems $P_0^k, \ldots, P_{l_k}^k$. If one sets $I^{l_k+1} = I^{l_k} \cup \{l_k\}$ and $\kappa_{l_k} = \kappa_{l_k} + 1$, then one may be interested in applying the next local branching descent from a first stage solution defined by : $x \in \{x \in X \mid \Delta(x, x^i) \geq \kappa_i, \forall i \in I^{l_k+1}\}$. This corresponds to applying the next descent from a feasible solution to subproblem $\overline{P}_{l_k}^k$ associated with (5.14)-(5.18). To obtain this solution, one can add constraints $\Delta(x, x^i) \geq \kappa_i$, $\forall i \in I^{l_k+1}$, to the MVP (5.12)-(5.13) and then use the optimal solution to this new problem as a starting point for the $k+1$th descent. In the case of the SVRPSD, to make the search for this new solution easier, function $Q(x, \bar{\xi})$ is dropped from the objective and the problem that is used is the TSP. This amounts to starting the next local branching descent from a route whose travel cost is low but which is in a region that has not yet been explored.

Therefore, a meta phase will be defined as a series of local branching descents whose strating point will be an optimal solution (defined by x^{k+1}) to the following problem :

$$\text{Min} \quad c^\top x \tag{5.19}$$

$$\text{s.t.} \quad \Delta(x, x^i) \geq \kappa_i, \ i \in I^{l_j+1}, \ j = 1, \ldots, k \tag{5.20}$$

$$x \in X. \tag{5.21}$$

It should be specified that an optimal solution to (5.19)-(5.21) is not necessarily needed. A good feasible solution can be sufficient. The feasible first stage solution x^{k+1} does not lie in any of the neighbourhoods explored in the previous k descents. Constraint $\Delta(x, x^{k+1}) \geq 1$ is added to problem (5.10)-(5.11), and the next series of descents is executed. Meta phases provide a diversification strategy in the multi-descent scheme. It should be noted that the local branching constraints are only used in order to find a new starting point. They are not used in the following descents. This will allow the algorithm to come back to subregions which have already been visited if the solutions found are locally good.

Descent structure

One should now examine how the local branching descents may be performed. Since local branching subproblems of type P_ν may be hard to solve efficiently, sampling will be used. There is an important point to be made concerning the size of the samples that one may use for this approximation. Since the local branching search strategy is aimed at controling the complexity associated with the first stage of problem (5.7)-(5.9), then one may use larger samples in the approximation of the recourse function for P_ν compared to the original stochastic problem (5.10)-(5.11).

The original branching decision, in a local branching descent, is taken on the basis of the objective value of the solutions considered. When sampling is used, the information provided by the objective function of the approximated subproblems will no longer be completely accurate. Furthermore, if the feasible region of a subproblem becomes too large, it may turn out to be impossible to solve it efficiently. Therefore, the strategy that is used in this paper, is to keep value κ fixed and apply the branching decision each time a new solution is found. A descent will include a fixed number of levels, where each level will be comprised of a series of subproblems that are approximated using the same sample of random events.

We will now briefly describe the algorithm that will be used to solve the local branching subproblems. Each subproblem will be solved to optimality or until a specified time limit is reached. The procedure used will be the branch and cut algorithm presented by Rei and *al.* [51]. There are three types of cuts that are generated by the algorithm : subtour elimination constraints (5.4), partial route cuts (5.6) and local branching valid inequalities as defined in [51]. Constraints (5.4) are obtained by using the procedures in the CVRPSEP package proposed by Lysgaard and *al.* [43]. These constraints are valid for all local branching subproblems explored by the algorithm. Therefore, a pool of cuts will be defined in order to reuse previously identified constraints. Both partial route cuts and local branching valid inequalities use information on the recourse function. These cuts are therefore only valid for subproblems in the same level of a descent, that is, when the recourse function is approximated using the same sample. Partial route

cuts will be reused on all subproblems in a given level. However, following the results obtained in [51], the local branching valid inequalities will be generated locally for each subproblem since this strategy was found to be more efficient.

In a given local branching descent, let $\{\omega_1^p, \ldots, \omega_n^p\}$ be the pth subset of randomly generated events of Ω for $p = 1, \ldots, m$, where m is the number of levels in the descent. If each level is made up of q subproblems ($P_{\nu p}$, $\nu = 1, \ldots, q$), then the local branching descent produces $m \times q$ different solutions. At the end of a descent, one must identify the best solution obtained. If x^i, $i = 1, \ldots, m \times q$, are the feasible solutions found, then one may use the m batches of randomly generated subsets to estimate the objective value of each solution. Therefore, let $\hat{f}_n^p(x) = c^\top x + \frac{1}{n}\sum_{j=1}^{n} Q(x, \xi(\omega_j^p))$ and $U_m^n(x) = \frac{1}{m}\sum_{p=1}^{m} \hat{f}_n^p(x)$, then the best solution obtained in iteration k will be estimated as being $x^{k^\star} \in \arg\min\{U_m^n(x^i) \mid i = 1, \ldots, m \times q\}$. This criterion directly follows the general principle that states that it is usually easier to find an ordering of the solutions rather than to correctly estimate their recourse value, see Fu [22].

For each local branching descent, one obtains a different feasible first stage solution that is identified as being the best one found in the neighbourhoods explored. Each of these solutions are obtained using different samples of random events. When the search process ends, one is left with the problem of having to identify the best solution found. Using simulation, a variety of methods have been proposed to deal with the problem of selecting between a finite number of possibilities. Following the classification provided by Fu [21], these methods can use the principles of multiple comparisons or ranking and selection. In the case of the SVRPSD, since one is now interested in evaluating the best recourse value obtained, then $Q(x)$ will be measured for these solutions.

5.5 Computational results

In order to assess the performance of the proposed algorithm, we selected a subset of instances from the paper of Rei and *al.* [51]. These instances were choosen to be the problems of size N = 60, 70, 80 and 90 for which $\overline{f}$ = 1.025, 1.05, 1.075 and 1.10 and

that were classified as being hard to solve optimaly by the classical L-shaped algorithm (see [51]). We thus obtained a total of 60 instances. All results reported will be averages over the number of instances (nb. i.) for all values of N and $\overline{f}$. Also, a time limit of 60, 120, 180 and 240 seconds was imposed respectively on the solution process for the local branching subproblems for the instances of size $N = 60$, 70, 80 and 90.

Tests will be conducted in three phases. In the first phase, we will establish the appropriate structure of the local branching descents. This will include finding the best value for the size of the neighbourhoods (parameter κ) as well as the number of scenarios (parameter n) that should be used in order to solve the local branching subproblems. We will also determine the number of subproblems (parameter q) that should be included in each level of a descent. In the second phase, we will look at how results vary when the number of base descents is changed in the meta phases. Finally, we will analyse the quality of the results obtained by the best strategy for the multi-descent local branching algorithm. To do so, we will compare it to the L-shaped algorithm proposed in [51] for which a large time limit of 6000 seconds is imposed. It should be specified that all results for the heuristic algorithm are average values over five runs. Also, all experiments were performed on a 2.4 GHz AMD Opteron 64 bit processor.

To establish the appropriate size of the neighbourhood as well as the number of scenarios necessary, the heuristic algorithm is first applied to produce one descent, starting from the solution to the TSP, where the number of levels is six ($m = 6$) and the number of subproblems for each level is one ($q = 1$). By doing so, each run performed includes a total of six subproblems solved. By fixing m and q, one can better see the tradeoffs between both parameters κ and n. In Table 5.1, results are reported for the following values : $\kappa = 4$, 6 and 8 and $n = 100$, 200 and 300. Since in all cases the descent is performed using the same starting point, the algorithm will tend to search the same region of the problem. Therefore, one first observes that the differences in the quality of the solutions obtained are very small. Results in Table 5.1 also include the number of times each run obtained the best solutions for a given κ (Local best ($*$)), and, overall values of κ (Absolute best ($\dagger$)). By analyzing these results, one is better

			$\kappa = 4$			$\kappa = 6$			$\kappa = 8$		
N	$\overline{J}$	ub. i.	$n = 100$	$n = 200$	$n = 300$	$n = 100$	$n = 200$	$n = 300$	$n = 100$	$n = 200$	$n = 300$
60	1.025	1	1314.54	1313.2	1312.43∗†	1314.72	1313.72	1312.43∗†	1314.32	1312.43∗†	1312.43∗†
	1.050	3	1347.5	1344.68∗	1346.85	1344.34	1343.24∗	1345.85	1341.08∗†	1343.21	1343.47
	1.075	5	1333.97	1334.05	1333.74∗	1332.26	1332.28	1331.99∗†	1332.99	1333.06	1332.88∗
	1.100	5	1343.14	1343.13	1341.96∗	1338.03	1337.96∗†	1339.03	1341.01	1340.2∗	1340.41
70	1.025	3	1434.6∗	1434.72	1434.82	1430.89∗†	1433.67	1433.53	1433.25	1433.13	1432.38∗
	1.050	3	1401.91	1401.51	1401.33∗	1401.8	1401.11∗†	1401.8	1401.74	1401.3	1401.19∗
	1.075	5	1455.5	1454.82∗	1454.82∗	1444.4	1442.78∗	1445.72	1443.68	1442.68∗†	1443.89
	1.100	4	1498.74	1497.45∗	1499.62	1495.57	1498.28	1495.25∗	1496.08	1494.53∗†	1495.1
80	1.025	2	1483.34	1481.99∗†	1485.77	1482.98∗	1483.06	1483.17	1483.09	1482.94∗	1483.92
	1.050	2	1494.04∗	1494.43	1494.6	1494.18	1494.03	1493.93∗†	1494.72	1493.96	1493.93∗†
	1.075	5	1491.76∗	1491.87	1491.95	1487.55∗	1488.32	1488.13	1487.23	1486.55∗†	1487
	1.100	5	1503.97	1501.01	1500.12∗	1495.02	1494.89	1494.09∗†	1494.19∗	1494.36	1496.33
90	1.025	2	1576.52	1577.63	1575.63∗	1575.27∗	1575.31	1575.63	1574.74	1573.96	1573.08∗†
	1.050	5	1606.11	1606.29	1605.87∗	1605.56∗	1606.56	1606.45	1604.73∗†	1604.96	1605.58
	1.075	5	1605.54∗	1606.28	1605.92	1604.48	1604.62	1603.92∗	1602.51	1602.01∗†	1604.73
	1.100	5	1598.81∗	1599.23	1599.41	1599.02	1598.76	1598.59∗	1596.54∗†	1596.55	1596.71
Local best (∗)			5	4	8	5	4	7	4	7	6
Absolute best (†)			0	1	1	1	2	4	3	5	3

TAB. 5.1 – Results : parameters κ and n

able to distinguish which of the parameter settings produced the best local search. For any given value κ, one observes that larger values of n produce the best solutions in general. For $\kappa = 4$ and 6, $n = 300$ is best and as for $\kappa = 8$, $n = 200$ seems to be slightly better than $n = 300$. Including all runs, when one compares the number of times each value of κ obtains the best overall values then one can see that $\kappa = 4$ is best on a total of two occasions, $\kappa = 6$ on seven occasions and $\kappa = 8$ on 11 occasions. Therefore, it seems that by fixing $\kappa = 8$, one obtains the best local search. Furthermore, since one generally obtains better results for larger values of n, then for all following runs we will set : $\kappa = 8$ and $n = 200$.

We will now examine how the concept of levels influences the quality of the results obtained. As was previously mentioned, solving several subproblems within a given level is interesting since the branch and cut algorithm may reuse the partial route cuts on all subproblems that are created using the same sample. In turn, this will accelerate the solution process for all subproblems on a given level. However, by reusing the same samples, one may also limit the search process. When solving a local branching subproblem, the quality of the solution obtained is dependent on the sample used to approximate the recourse function. Results will be poor if a non representative sample

N	$\overline{f}$	nb. i.	6-1	3-2	2-3	1-6
60	1.025	1	1312.43*	1313.00	1313.00	1313.02
	1.050	3	1343.21	1342.11*	1343.26	1343.33
	1.075	5	1333.06	1333.31	1333.02	1332.43*
	1.100	5	1340.2*	1340.99	1342.79	1340.47
70	1.025	3	1433.13	1432.72*	1433.49	1433.16
	1.050	3	1401.3	1401.54	1401.52	1400.83*
	1.075	5	1442.68*	1443.71	1443.92	1443.48
	1.100	4	1494.53*	1494.96	1494.85	1495.59
80	1.025	2	1482.94*	1483.41	1484.59	1483.79
	1.050	2	1493.96	1494.08	1493.92*	1494.10
	1.075	5	1486.55	1485.84*	1486.76	1487.03
	1.100	5	1494.36*	1494.40	1493.58	1493.99
90	1.025	2	1573.96*	1574.40	1575.21	1575.21
	1.050	5	1604.96*	1606.05	1606.60	1605.86
	1.075	5	1602.01	1600.31*	1603.53	1602.64
	1.100	5	1596.55	1596.69	1596.29	1595.17*
Best (*)			7	4	2	3

TAB. 5.2 – Results : parameters m and q

is used. If the same non representative sample is applied on different subproblems then one can seriously limit the search process in the neighbourhoods that are explored.

In Table 5.2 the quality of the solutions obtained are presented when $m = 6$, 3, 2 and 1 and $q = 1$, 2, 3 and 6. For example, 6-1 will refer to the case where the total number of levels is six ($m = 6$) and each level contains one subproblem ($q = 1$). For these tests the number of descents is again limited to one, for a total of six subproblems solved in each run. By using the same starting point for each descent and by fixing both $\kappa = 8$ and $n = 200$, one is able to clearly see how results vary with the structure of the descent. The total number of times each run obtained the best results is also reported (Best (*)). Results in Table 5.2 seem to indicate that 6-1 outperforms all others. One obtains the best results on seven occasions with 6-1, compared to four for 3-2, two for 2-3 and three occasions for 1-6. It would seem that by using different samples for each subproblem, one better hedges against the risk of relying heavily on non representative samples. Therefore, the type of descent that will be used in the multi-descent approach will be made up of levels for which $q = 1$ and where subproblems are created using $\kappa = 8$ and $n = 200$.

We will now examine the multi-descent search strategies that one may use for a given number of overall base descents. We fix the total number of base descents to six and each local branching descent performed is limited to a depth of three levels ($m = 3$), which produces a total of 18 subproblems solved by the algorithm. In Table 5.3, results are reported for the cases where the number of base descents varies in each meta phase. Therefore, the 6/1 column refers to the case where six meta phases of size one are performed. The 3/2 column represents runs made up of three meta phases of size two. The 2/3 column is for the case where two meta phases of size three are carried out. Finally, the 1/6 column is the case where six consecutive base descents are performed. These results will help to establish the relative importance that one should give to the diversification strategy versus the intensification strategy. Once again, if one observes the number of times each run is the overall best (Best (*)), then one may see that 6/1 is better on eight occasions, 3/2 on six, 2/3 on two and 1/6 on four. For a given level of effort (18 local branching subproblems), it would seem that by applying more

N	$\overline{f}$	nb. i.	6/1	3/2	2/3	1/6
60	1.025	1	1310.94*	1312.43	1311.35	1312.07
	1.050	3	1342.24	1341.70	1341.08	1340.85*
	1.075	5	1332.04	1333.25	1331.92*	1331.92*
	1.100	5	1336.14*	1336.23	1336.26	1336.73
70	1.025	3	1430.79*	1431.99	1431.44	1432.39
	1.050	3	1399.65	1399.53*	1400.70	1401.14
	1.075	5	1440.37*	1441.03	1441.57	1442.53
	1.100	4	1492.70	1492.69*	1493.01	1493.62
80	1.025	2	1481.49	1481.04*	1481.94	1481.49
	1.050	2	1493.92*	1493.92*	1493.92*	1493.92*
	1.075	5	1483.76*	1485.43	1485.14	1485.70
	1.100	5	1489.91	1489.77*	1490.56	1489.95
90	1.025	2	1572.23	1571.98*	1572.11	1573.23
	1.050	5	1603.24*	1603.50	1604.63	1604.06
	1.075	5	1597.96*	1600.70	1598.80	1599.63
	1.100	5	1594.94	1594.84	1594.47	1594.06*
Best (*)			8	6	2	4

TAB. 5.3 – Results : Meta phases/Base descents

diversification, one obtains better results. Therefore, in this case, the best strategy concerning the size of the meta phases is to set it equal to one.

We will conclude this section by comparing the multi-descent heuristic with the L-shaped algorithm of Rei and *al.* [51]. The heuristic algorithm is applied by specifying the total number of meta phases of size one to be done. Again each descent will have a depth of three levels (m =3). As for the L-shaped algorithm, a maximum time of 6000 seconds is imposed for the solution process. In Table 5.4 all instances solved are separated into three categories according to the results obtained by the L-shaped algorithm. The *sol.* category refers to all cases were the L-shaped algorithm was able to solve the problem for an optimality gap of $\epsilon \leq 1\%$. The *not sol.* category includes all instances that the L-shaped algorithm was unable to solve in the maximum time allowed but where at least one feasible solution was obtained. Finally, the *und.* category refers to the cases where the L-shaped algorithm was unable to solve the problem and no feasible solution was found before the maximum time allowed was reached. According to this classification, one obtains 25 instances in the *sol.* category, 34 in the *not sol.* category and only one in the *und.* category. In order to see how results vary for the multi-descent scheme, runs for the heuristic algorithm are made for two, four, six and eight meta phases. Results in Table 5.4 include both the best solution values found by both algorithms as well as the solution times in seconds, which are the values reported between parentheses.

If one considers those instances that are solved by the L-shaped algorithm (*sol.*), one first observes that the solutions obtained by the multi-descent heuristic are in almost all cases either optimal or near optimal. For this category, the total mean results show that the 2/1 runs obtain an average value of 1538.67 in 639.53 seconds of computation time. As for the L-shaped algorithm, it obtains 1536.93 in 1331.62 seconds. The heuristic finds near optimal solutions in half the computation time when compared to the exact algorithm. Furthermore, as the number of meta phases increases, the quality of results converge to the optimal values. The 8/1 runs produce a total average value of 1536.51 versus 1536.93 for the L-shaped algorithm, whose results have an average gap of less than or equal to one percent. One should note that the total average computation time of 8/1 is larger than that of the exact algorithm (2625.08 seconds versus 1331.62

N	$\overline{f}$	Category	nb. i.	2/1	4/1	6/1	8/1	L-shaped
60	1.025	*sol.*	1	1312.43 (248.25)	1311.71 (496.91)	1310.94 (873.40)	1310.54 (1150.11)	1310.06 (637.28)
		not sol.	0	-	-	-	-	-
		und.	0	-	-	-	-	-
	1.05	*sol.*	3	1343.63 (350.27)	1342.8 (757.1)	1342.24 (1128.43)	1340.85 (1520.49)	1340.91 (275.76)
		not sol.	0	-	-	-	-	-
		und.	0	-	-	-	-	-
	1.075	*sol.*	3	1360.17 (342.32)	1359.12 (730.58)	1357.93 (1093.31)	1357.73 (1491.66)	1358.67 (1488.22)
		not sol.	2	1293.21 (411.18)	1293.21 (820.96)	1293.21 (1231.25)	1293.21 (1641.56)	1281.72 (6034.74)
		und.	0	-	-	-	-	-
	1.10	*sol.*	2	1312.55 (395.91)	1311.68 (770.25)	1311.88 (1173.97)	1311.88 (1545.52)	1311.58 (3890.89)
		not sol.	3	1356.13 (386.24)	1352.73 (775.50)	1352.31 (1195.63)	1351.95 (1574.10)	1353.74 (6038.66)
		und.	0	-	-	-	-	-
70	1.025	*sol.*	3	1431.04 (388.04)	1430.79 (732.81)	1430.79 (1115.05)	1430.79 (1476.41)	1433.46 (157.34)
		not sol.	0	-	-	-	-	-
		und.	0	-	-	-	-	-
	1.05	*sol.*	2	1387.53 (395.17)	1385.72 (791.08)	1385.93 (1272.56)	1380.74 (1680.38)	1380.77 (369.70)
		not sol.	1	1427.08 (688.92)	1427.07 (1440.92)	1427.07 (2075.92)	1426.68 (2898.15)	1427.57 (6011.45)
		und.	0	-	-	-	-	-
	1.075	*sol.*	1	1401.03 (766.75)	1397.65 (1554.18)	1397.65 (2397.02)	1397.65 (3139.22)	1397.65 (2887.62)
		not sol.	4	1453.92 (698.36)	1451.52 (1408.84)	1451.05 (2152.15)	1450.27 (2886.54)	1455.58 (6020.53)
		und.	0	-	-	-	-	-
	1.10	*sol.*	0	-	-	-	-	-
		not sol.	4	1495.48 (790.11)	1493.25 (1568.15)	1492.70 (2328.58)	1492.90 (3178.22)	1493.06 (6011.10)
		und.	0	-	-	-	-	-
80	1.025	*sol.*	2	1481.78 (876.32)	1481.04 (1614.56)	1481.49 (2531.47)	1480.33 (3347.57)	1480.14 (106.97)
		not sol.	0	-	-	-	-	-
		und.	0	-	-	-	-	-
	1.05	*sol.*	1	1505.64 (1120.15)	1505.33 (2182.12)	1505.33 (3347.52)	1505.33 (4362.17)	1505.33 (4391.95)
		not sol.	1	1482.53 (1099.36)	1482.53 (2268.15)	1482.53 (3521.99)	1482.53 (4642.29)	1468.72 (6004.49)
		und.	0	-	-	-	-	-
	1.075	*sol.*	2	1392.86 (824.91)	1392.86 (1601.04)	1392.86 (2234.21)	1392.86 (3216.07)	1392.87 (41.24)
		not sol.	3	1549.60 (1098.36)	1547.70 (2262.29)	1544.37 (3387.57)	1544.09 (4625.32)	1554.97 (6027.58)
		und.	0	-	-	-	-	-
	1.10	*sol.*	2	1505.56 (939.47)	1505.47 (1980.21)	1502.50 (2994.12)	1500.63 (3984.65)	1497.24 (2656.43)
		not sol.	3	1483.51 (1144.09)	1481.45 (2340.40)	1481.51 (3512.41)	1480.00 (4765.84)	1497.28 (6012.34)
		und.	0	-	-	-	-	-
90	1.025	*sol.*	1	1638.93 (891.35)	1636.97 (1495.22)	1636.97 (2449.83)	1636.77 (3602.31)	1636.37 (1043.17)
		not sol.	1	1508.08 (770.21)	1508.08 (2005.50)	1507.34 (3174.54)	1507.04 (4283.28)	1510.94 (6084.15)
		und.	0	-	-	-	-	-
	1.05	*sol.*	2	1590.80 (1040.17)	1590.20 (2293.46)	1589.90 (3588.56)	1590.00 (4703.11)	1591.27 (2060.68)
		not sol.	2	1612.69 (1511.47)	1610.03 (2984.18)	1608.91 (4452.84)	1608.95 (6076.69)	1611.88 (6013.45)
		und.	1	1618.76 (1455.36)	1618.56 (3237.7)	1618.56 (5702.97)	1618.56 (7304.46)	- (6294.68)
	1.075	*sol.*	0	-	-	-	-	-
		not sol.	5	1601.41 (1571.91)	1600 (3174.45)	1597.96 (4757.07)	1596.95 (6377.13)	1601.17 (6007.51)
		und.	0	-	-	-	-	-
	1.10	*sol.*	0	-	-	-	-	-
		not sol.	5	1596.63 (1535.45)	1595.88 (3111.71)	1594.94 (4721.05)	1594.01 (6278.16)	1592.30 (6016.41)
		und.	0	-	-	-	-	-
Total		*sol.*	25	1538.67 (639.53)	1537.81 (1278.27)	1537.38 (1955.92)	1536.51 (2625.08)	1536.93 (1331.62)
		not sol.	34	1505.44 (1052.37)	1503.77 (2141.12)	1502.8 (3227.68)	1502.24 (4343.87)	1505.13 (6019.95)
		und.	1	1618.76 (1455.36)	1618.56 (3237.7)	1618.56 (5702.97)	1618.56 (7304.46)	- (6294.68)

TAB. 5.4 – Results : Multi-descent algorithm vs. L-shaped

seconds). However, what these results seem to indicate is that the heuristic is robust since it generates near optimal solutions relatively quickly, and in any case, if the number of meta phases is increased then the procedure converges to optimality.

We will now analyse the results obtained on those instances that were not solved by the L-shaped algorithm (*not sol.* and *und.*). The detailed results show that the multi-descent heuristic obtains better results in ten cases compared to only three for the exact algorithm. If one considers those instances for which $\overline{f} = 1.075$ and 1.10, then the heuristic is usually better than the L-shaped algorithm and the differences can be quite significant. The multi-descent algorithm can obtain similar results a lot faster, as is the case for the three instances of size $N = 60$ and $\overline{f} = 1.10$ where the average results are 1351.95 in 1574.10 seconds for 8/1 compared to 1353.74 in 6038.66 seconds for the L-shaped algorithm; or, it can obtain better results in favorable times, as in the case of the three instances of size $N = 80$ and $\overline{f} = 1.10$ where the average results are 1480.00 in 4765 seconds for 8/1 compared to 1497.28 in 6012.34 seconds for the exact algorithm. Out of the three cases where the exact algorithm obtained better results, only two are important to analyse ($N = 60$, $\overline{f} = 1.075$ and $N = 80$, $\overline{f} = 1.05$). In the case of $N = 90$ and $\overline{f} = 1.10$, results are quite similar, 1594.01 in 6278.16 seconds for 8/1 versus 1592.30 in 6016.41 seconds for the L-shaped algorithm. For the two instances for which $N = 60$ and $\overline{f} = 1.075$, the 8/1 runs obtain 1293.21 in 1641.56 seconds compared to 1281.72 in 6034.74 seconds for the L-shaped algorithm. In this case, the computation times are not comparable. If one increases the number of meta phases to 16/1, then the results obtained are 1282.03 in 3355.66 seconds. Once again, the heuristic produces comparable results in a favorable time. For the single instance of size $N = 80$ and $\overline{f} = 1.05$, the same observation could not be made. In this case, the exact algorithm obtains 1468.72 in 6004.49 seconds. The 8/1 runs produce 1482.53 in 4642.29 seconds and when the number of meta phases is increased to 16/1, the results are 1478.56 in 9667.59 seconds. For this problem, the strategy used in the multi-descent scheme was unable to outperform the L-shaped algorithm. However, it remains a single case out of the 60 instances tested. The total mean results show that the multi-descent algorithm produces equivalent results in a lot less time, the 2/1 runs obtain 1505.44 in

1052.37 seconds compared to 1505.13 in 6019.95 seconds for the L-shaped algorithm. As one increases the number of meta phases, the results obtained are improved while the computation times remain favorable, the 8/1 runs obtain 1502.24 in 4343.87 seconds. Finally, the single instance in the *und* category can be used to illustrate that the heuristic is able to produce solutions quickly even if the exact algorithm was unable to.

5.6 Conclusion

In this paper, we propose a new hybrid algorithm that combines both local branching and Monte Carlo sampling in a multi-descent search strategy for integer 0-1 stochastic programming problems. By controling simultaneously the inherent complexities associated with both the first stage problem and the recourse function, one is able to better limit the effort needed to solve the approximating subproblems. Furthermore, by using the local branching constraints in order to obtain diversification for the search strategy, one is able to better explore the feasible region of the original problem. This method was specialized to the case of the SVRPSD and was proven to be quite effective to solve hard instances of the problem. One should also point out that the algorithmic principles that were used are all quite general. In future work, it would be interesting to see how one could adapt these ideas to other stochastic programming problems.

Chapitre 6

Conclusion

L'objectif principal de cette thèse a été de développer des algorithmes performants pour résoudre des problèmes de programmation stochastique en nombres entiers. Puisque plusieurs des algorithmes développés dans ce domaine utilisent la décomposition de Benders, le premier article présenté dans le cadre de ce travail propose une méthode permettant d'accélérer cette approche générale de décomposition. L'idée proposée est d'appliquer des phases de recherche de type séparation locale en utilisant, comme point de départ, les solutions aux problèmes maîtres relaxés. Ce faisant, il est possible de trouver à chaque itération de l'algorithme de nouvelles coupes d'optimalité. La séparation locale permet également de générer des coupes valides qui améliorent la description du domaine réalisable définie par les relaxations. Cette nouvelle approche multi-coupes permet d'améliorer la borne inférieure générée par l'algorithme de Benders. De plus, puisque les phases de séparation locale trouvent rapidement des solutions réalisables différentes, il est possible d'améliorer simultanément la borne supérieure de l'algorithme. L'expérimentation numérique effectuée sur des problèmes de conception de réseaux offrent un exemple des bénéfices de l'approche. Dans ce cas, la séparation locale permet de réduire de façon significative le nombre de coupes qui sont nécessaires à la résolution des problèmes. En améliorant simultanément les bornes inférieure et supérieure, la séparation locale permet à l'algorithme de converger beaucoup plus rapidement.

La contribution du premier article est d'ordre méthodologique, car il est question d'améliorer l'approche générale de Benders. Il est important de préciser que la séparation locale peut être appliquée à tout contexte d'optimisation impliquant des variables de décisions binaires. Ce type de recherche nécessite uniquement l'utilisation d'un optimiseur générique. Dans le contexte de la décomposition de Benders, il est possible d'utiliser l'algorithme de Benders lui-même comme optimiseur générique. Par conséquent, la méthode proposée peut être adaptée à toutes les problématiques où la décomposition de Benders est utilisée.

Le deuxième article de cette thèse spécialise l'idée d'utiliser la séparation locale au sein de la décomposition de Benders au cas de l'algorithme « L-shaped » binaire. L'algorithme « L-shaped » binaire est utilisé pour résoudre des problèmes de programmation stochastique où les variables de première étape sont binaires et où le recours est entier. Dans ce cas, la séparation locale sert à générer de nouvelles inégalités valides qui bornent les valeurs du recours associées à différentes régions du domaine réalisable de première étape. Ces nouvelles inégalités générales permettent d'améliorer la description du recours dans les problèmes relaxés résolus par l'algorithme et peuvent être utilisées en complément des coupes d'optimalité définies dans ce contexte. Pour cet article, ces inégalités sont appliquées au cas des problèmes de tournées d'un véhicule avec demandes stochastiques. Deux stratégies sont présentées pour l'utilisation de ces inégalités au niveau de la résolution du problème considéré. Une procédure de génération de plans coupants, qui intègre l'ensemble des coupes valides pour le problème, est également développée. L'expérimentation numérique, effectuée sur un ensemble de 280 instances, démontre que ces nouvelles inégalités améliorent de façon significative l'algorithme « L-shaped » binaire appliqué à cette problématique.

Les contributions de cet article sont à la fois méthodologique et pratique. D'un point de vue méthodologique, les inégalités proposées sont applicables à tous les problèmes stochastiques pouvant être résolus par l'algorithme « L-shaped » binaire. Au niveau pratique, la spécialisation de ces inégalités au cas des problèmes de tournées d'un véhicule avec demandes stochastiques a permis de développer l'approche de résolution exacte la plus efficace pour cette problématique. Par contre, l'expérimentation numérique a

également permis d'identifier des instances particulièrement difficiles à résoudre de façon exacte, et ce, malgré l'amélioration proposée. Ceci a motivé le développement d'une approche de résolution heuristique efficace.

Le dernier article présenté dans le cadre de cette thèse propose donc une heuristique pour les problèmes de tournées d'un véhicule avec demandes stochastiques. L'heuristique qui est développée est une approche hybride alliant la recherche par séparation locale et la simulation de Monte Carlo. La simulation de Monte Carlo est utilisée afin de réduire la complexité inhérente au recours. La séparation locale permet, quant à elle, d'explorer efficacement le domaine réalisable de première étape tout en contrôlant la complexité combinatoire du problème. Des stratégies de diversification et d'intensification sont incorporées au sein de l'heuristique qui opère selon une approche de type multi-descentes. L'expérimentation numérique, effectuée sur un ensemble d'instances particulièrement difficiles à résoudre de façon exacte, a permis de voir que l'heuristique est à la fois robuste, car elle permet de trouver des solutions optimales ou quasi optimales, et efficace, car elle génère de bonnes solutions très rapidement.

Les contributions de cet article sont principalement d'ordre pratique, car ce qui est proposé est une approche heuristique efficace pour les problèmes de tournées d'un véhicule avec demandes stochastiques. Par contre, d'un point de vue méthodologique, il est important de noter que les principes de recherche utilisés dans le cadre de cette méthode sont généraux et peuvent être facilement adaptés à d'autres problématiques.

Dans le cadre de ce travail, l'approche qui a été préconisée afin de produire des algorithmes efficaces a été l'hybridation de méthodes. Plus précisément, la séparation locale a été utilisée au sein de différentes approches de résolution classiques afin de rendre celles-ci plus performantes. Plusieures avenues intéressantes de recherche sont possibles à partir du travail qui a été accompli dans cette thèse. Les approches méthodologiques proposées sont générales, ce qui permet de les appliquer à différentes problématiques. La décomposition de Benders est de plus en plus utilisée à l'extérieur du cadre de l'optimisation linéaire. Par conséquent, il serait intéressant de voir comment la séparation locale pourrait servir l'algorithme de Benders dans ces contextes. Au niveau de l'optimisation

stochastique, les problèmes de tournées de véhicules posent des défis importants. Par conséquent, un développement intéressant futur serait d'étendre les algorithmes proposés au cas des problèmes impliquant plusieures véhicules et des coûts de déplacements stochastiques.

Bibliographie

[1] AHMED, S. et A. SHAPIRO, «The sample average approximation method for stochastic programs with integer recourse», Optimization Online, http ://www.optimization-online.org, February 2002.

[2] AHMED, S., M. TAWARMALANI et N.V. SAHINIDIS, «A finite branch and bound algorithm for two-stage stochastic integer programs», *Mathematical Programming*, vol. 100, 2004, pp. 355–377.

[3] AVERBAKH, I.L., «An iterative method of solving two-stage discrete stochastic programming problems with additively separable variables», *USSR Computational Mathematics and Mathematical Physics*, vol. 31, no. 6, 1991, pp. 21–27.

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

[5] BIANCHI, L., M. BIRATTARI, M. CHIARANDINI, M. MANFRIN, M. MASTROLILLI, L. PAQUETE, O. ROSSI-DORIA et T. SCHIAVINOTTO. «Hybrid metaheuristics for the vehicle routing problem with stochastic demands». Tech. rep., IDSIA / USI-SUPSI Dalle Molle Institute for Artificial Intelligence, Galleria 2, 6928 Manno, Switzerland, March 2005.

[6] BIRGE, J.R et F. LOUVEAUX, «A multicut algorithm for two-stage stochastic linear programs», *European Journal of Operations Research*, vol. 34, 1988, pp. 384–392.

[7] BIRGE, J.R. et F. LOUVEAUX, *Introduction to Stochastic Programming*. Springer, 1997.

[8] CAROE, C.C. et R. SCHULTZ, «Dual decomposition in stochastic integer programming», *Operations Research Letters*, vol. 24, 1999, pp. 37–45.

[9] CAROE, C.C. et J. TIND, «A cutting-plane approach to mixed 0-1 stochastic integer programs», *European Journal of Operational Research*, vol. 101, 1997, pp. 306–316.

[10] CAROE, C.C. et J. TIND, «L-shaped decomposition of two-stage stochastic programs with integer recourse», *Mathematical Programming*, vol. 83, 1998, pp. 451–464.

[11] CHEPURI, K. et T. HOMMEM-DE-MELLO, «Solving the vehicle routing problem with stochastic demands using the cross-entropy method», *Annals of Operations Research*, vol. 134, 2005, pp. 153–181.

[12] CODATO, G. et M. FISCHETTI. «Combinatorial benders cuts for mixed-integer linear programming». In *Integer Programming and Combinatorial Optimization*, D. Bienstock et G. Nemhauser, Eds. Lectures Notes in Computer Science : Springer-Verlag, 2004, pp. 178–195.

[13] COSTA, A.M., «A survey on benders decomposition applied to fixed-charge network design problems», *Computers and operations research*, vol. 32, 2005, pp. 1429–1450.

[14] CÔTÉ, G. et M.A. LAUGHTON, «Large-scale mixed integer programming : Benders-type heuristics», *European Journal of Operational Research*, vol. 16, 1984, pp. 327–333.

[15] CRAINIC, T.G., A. FRANGIONI et B. GENDRON, «Bundle-based relaxation methods for multicommodity capacitated fixed charge network design problems», *Discrete Applied Mathematics*, vol. 112, 2002, pp. 73–99.

[16] DANTZIG, G.B. et P.W. GLYNN, «Parallel processors for planning under uncertainty», *Annals of Operations Research*, vol. 22, 1990, pp. 1–21.

[17] DROR, M., «Modeling vehicle routing with uncertain demands as a stochastic program : Properties of the corresponding solution», *European Journal of Operational Research*, vol. 64, 1993, pp. 432–441.

[18] DROR, M., G. LAPORTE et P. TRUDEAU, «Vehicle routing with stochastic demands : properties and solution frameworks», *Transportation Science*, vol. 23, no. 3, 1989, pp. 166–176.

[19] ERMOLIEV, Y. «Stochastic quasigradient methods». In *Numerical techniques for stochastic optimization* (1988), Springer-Verlag, Berlin, pp. 141–186.

[20] FISCHETTI, M. et A. LODI, «Local branching», *Mathematical Programming*, vol. 98, 2003, pp. 23–47.

[21] FU, M.C., «Optimization via simulation : a review», *Annals of Operations Research*, vol. 53, 1994, pp. 199–247.

[22] FU, M.C., «Optimization for simulation theory vs. practice», *INFORMS Journal on Computing*, vol. 14, no. 3, 2002, pp. 192–215.

[23] GENDREAU, M., G. LAPORTE et R. SÉGUIN, «An exact algorithm for the vehicle routing problem with stochastic demands and customers», *Transportation Science*, vol. 29, no. 2, 1995, pp. 143–155.

[24] GENDREAU, M., G. LAPORTE et R. SÉGUIN, «A tabu search heuristic for the vehicle routing problem with stochastic demands and customers», *Operations Research*, vol. 44, no. 3, 1996.

[25] GENDRON, B., T.G. CRAINIC, A. FRANGIONI et F. GUERTIN. «Oobb : Object-oriented tools for parallel branch-and-bound». In *PAREO, Mont-Tremblant, Canada* (16–21 january 2005).

[26] GEOFFRION, A.M., «Elements of large-scale mathematical programming part 1 : Concepts», *Management Science*, vol. 11, 1970, pp. 652–675.

[27] GEOFFRION, A.M., «Elements of large-scale mathematical programming part 2 : Synthesis of algorithms and bibliography», *Management Science*, vol. 11, 1970, pp. 676–691.

[28] GRAVER, J.E, «On the foundations of linear and integer linear programming i», *Mathematical Programming*, vol. 8, 1975, pp. 207–226.

[29] GUTIN, G. ET A.P. PUNNEN, Eds., *The Traveling Salesman Problem and Its Variations*. Springer, 2002.

[30] HEMMECKE, R. et R. SCHULTZ, «Decomposition of test sets in stochastic integer programming», *Mathematical Programming*, vol. 94, 2003, pp. 323–341.

[31] HIGLE, J.L. et S. SEN, *Stochastic Decomposition A statistical method for large scale stochastic linear programming*. Kluwer Academic Publishers, 1996.

[32] HJORRING, C. et J. HOLT, «New optimality cuts for a single-vehicle stochastic routing problem», *Annals of Operations Research*, vol. 86, 1999, pp. 569–584.

[33] HOLMBERG, K., «On the convergence of cross decomposition», *Mathematical Programming*, vol. 47, 1990, pp. 269–296.

[34] HOLMBERG, K., «On using approximations of the benders master problem», *European Journal of Operational Research*, vol. 77, 1994, pp. 111–125.

[35] HOOKER, J.N., *Logic-Based Methods for Optimization : Combining Optimization and Constraint Satisfaction.* John Wiley and Sons, 2000.

[36] KLEIN-HANEVELD, W.K. et M.H. VAN DER VLERK, «Stochastic integer programming : General models and algorithms», *Annals of Operations Research*, vol. 85, 1999, pp. 39–57.

[37] KLEYWEGT, A.J., A. SHAPIRO et T. HOMMEM-DE-MELLO, «The sample average approximation method for stochastic discrete optimization», *SIAM Journal on Optimization*, vol. 12, no. 2, 2001, pp. 479–502.

[38] KUHN, H.W. ET A.W. TUCKER, Eds., *Linear Inequalities and Related Systems.* Princeton University Press, 1956.

[39] LAPORTE, G. et F. LOUVEAUX, «The integer l-shaped method for stochastic integer programs with complete recourse», *Operations Research Letters*, vol. 13, 1993, pp. 133–142.

[40] LAPORTE, G., F. LOUVEAUX et L. Van HAMME, «An integer l-shape algorithm for the capacitated vehicle routing problem with stochastic demands», *Operations Research*, vol. 50, no. 3, 2002, pp. 415–423.

[41] LINDEROTH, J., A. SHAPIRO et S. WRIGHT, «The empirical behavior of sampling methods for stochastic programming», *Annals of Operations Research*, vol. 142, 2006, pp. 215–241.

[42] LOUVEAUX, F. «An introduction to stochastic transportation models». In *Operations Research and Decision Aid Methodologies in Traffic and Transportation Management*, M. Labbe, G. Laporte, K. Tanczos, et P. Toint, Eds., vol. 166 of *Computer and Systems Sciences.* Springer-Verlag, 1998, pp. 244–263.

[43] LYSGAARD, J., A.N. LETCHFORD et R.W. EGLESE, «A new branch-and-cut algorithm for the capacitated vehicle routing problem», *Mathematical Programming*, vol. 100, 2004, pp. 423–445.

[44] MADANSKY, A., «Inequalities for stochastic linear programming problems», *Management Science*, vol. 6, 1960, pp. 197–204.

[45] MAGNANTI, T.L. et R.T. WONG, «Accelerating benders decomposition algorithmic enhancement and model selection criteria», *Operations Research*, vol. 29, no. 3, 1981, pp. 464–484.

[46] MAK, W-K., D. MORTON et R.K. WOOD, «Monte carlo bounding techniques for determining solution quality in stochastic programs», *Operations Research Letters*, 1999, pp. 47–56.

[47] MCDANIEL, D. et M. DEVINE, «A modified benders'partitioning algorithm for mixed integer programming», *Management Science*, vol. 24, no. 3, 1977, pp. 312–319.

[48] NEMHAUSER, G.L. et L.A. WOLSEY, *Integer and Combinatorial Optimization.* Wiley-Interscience, New-York, NY, 1988.

[49] NORKIN, V.I., Y.M. ERMOLIEV et A. RUSZCZYŃSKI, «On optimal allocation of indivisibles under uncertainty», *Operations Research*, vol. 46, 1998, pp. 381–395.

[50] REI, W., J.F CORDEAU, M. GENDREAU et P. SORIANO. «Accelerating benders decomposition by local branching». Tech. rep., Center for Research on Transportation, C.P. 6128, succursale Centre-ville Montréal QC H3C 3J7 Canada, January 2006.

[51] REI, W., M. GENDREAU et P. SORIANO. «Local branching cuts for the 0-1 integer l-shaped algorithm». Tech. rep., Center for Research on Transportation, C.P. 6128, succursale Centre-ville Montréal QC H3C 3J7 Canada, January 2007.

[52] ROCKAFELLAR, R.T. et R.J-B WETS, «Scenarios and policy aggregation in optimization under uncertainty», *Mathematics of Operations Research*, vol. 16, no. 1, 1991, pp. 119–147.

[53] SANTOSO, T., S. AHMED, M. GOETSCHALCKX et A. SHAPIRO, «A stochastic programming approach for supply chain network design under uncertainty», *European Journal of Operational Research*, vol. 167, 2005, pp. 96–115.

[54] SCHULTZ, R., «Stochastic programming with integer variables», *Mathematical Programming*, vol. 97, 2003, pp. 285–309.

[55] SCHULTZ, R., L. STOUGIE et M.H. VAN DER VLERK, «Solving stochastic programs with integer recourse by enumeration : A framework using grobner basis reductions», *Mathematical Programming*, vol. 83, 1998, pp. 229–252.

[56] SECOMANDI, N., «Comparing neuro-dynamic programming algorithms for the vehicle routing problem with stochastic demands», *Computers & Operations Research*, vol. 27, 2000, pp. 1201–1225.

[57] SECOMANDI, N., «A rollout policy for the vehicle routing problem with stochastic demands», *Operations Research*, vol. 49, no. 5, 2001, pp. 796–802.

[58] SEN, S. et J.L. HIGLE, «The c^3 theorem and a d^2 algorithm for large scale stochastic mixed-integer programming : Set convexification», *Mathematical Programming*, vol. 104, 2005, pp. 1–20.

[59] SEN, S. et H.D. SHERALI, «Decomposition with branch-and-cut approaches for two-stage stochastic mixed-integer programming», *Mathematical Programming*, vol. 106, 2006, pp. 203–223.

[60] SHAPIRO, A. et T. HOMMEM-DE-MELLO, «On the rate of convergence of optimal solutions of monte carlo approximations of stochastic programs», *SIAM Journal on Optimization*, vol. 11, no. 1, 2000, pp. 70–86.

[61] SHERALI, H.D. et B.M.P. FRATICELLI, «A modification of benders decomposition algorithm for discrete subproblems : An approach for stochastic programs with integer recourse», *Journal of Global Optimization*, vol. 22, 2002, pp. 319–342.

[62] THOMAS, R.R., «A geometric buchberger algorithm for integer programming», *Mathematics Of Operations Research*, vol. 20, no. 4, 1995, pp. 864–884.

[63] TOTH, P. ET D. VIGO, Eds., *The Vehicle Routing Problem*. Monographs on Discrete Mathematics and Applications. SIAM, 2001.

[64] VAN DER VLERK, M.H., «Convex approximations for complete integer recourse models», *Mathematical Programming*, vol. 99, 2004, pp. 287–310.

[65] VAN ROY, T.J., «Cross decomposition for mixed integer programming», *Mathematical Programming*, vol. 25, 1983, pp. 46–63.

[66] VAN SLYKE, R.M et R.J.-B. WETS, «L-shaped programs with applications to optimal control and stochastic programming», *SIAM Journal on Applied Mathematics*, vol. 17, no. 4, 1969, pp. 638–663.

[67] VERWEIJ, B., S. AHMED, A.J. KLEYWEGT, G. NEMHAUSER et A. SHAPIRO, «The sample average approximation method applied to stochastic routing problems : a computational study», *Computational Optimization and Applications*, vol. 24, 2003, pp. 289–333.

[68] WEISMANTEL, R., «Test sets of integer programs», *Mathematical Methods of Operations Research*, vol. 47, 1998, pp. 1–37.

[69] WOLLMER, R.D., «Two stage linear programming under uncertainty with 0-1 integer first stage variables», *Mathematical Programming*, vol. 19, 1980, pp. 279–288.

[70] YANG, W., K. MATHUR et R.H. BALLOU, «Stochastic vehicle routing problem with restocking», *Transportation Science*, vol. 34, no. 1, 2000, pp. 99–112.