Résolution de grands problèmes stochastiques multi-étapes : Application à un problème de dimensionnement de capacités et de gestion de flux et de stocks

Georges Kolomvos

ÉCOLE CENTRALE DES ARTS ET MANUFACTURES
« ÉCOLE CENTRALE PARIS »

THÈSE

présentée par
Georges Kolomvos

pour l'obtention du

GRADE DE DOCTEUR

Spécialité : Génie Industriel

Laboratoire d'accueil : LGI

SUJET :
Résolution de grands problèmes stochastiques multi-étapes : Application à un problème de dimensionnement de capacités et de gestion de flux et de stocks

soutenue le : 19 janvier 2007

devant un jury composé de :

Leslie Trotter	**Président du jury**
Michel Minoux	**Directeur de thèse**
Yves Dallery	**Co-Directeur de thèse**
Jean-Claude Hennet	**Rapporteur**
Abdel Lisser	**Rapporteur**
Christian van Delft	**Examinateur**

2007 - 04

Remerciements

Je tiens à remercier tout d'abord Jean-Claude HENNET et Abdel LISSER qui, avec leur disponibilité et leur bonne humeur, m'ont fait l'honneur de rapporter sur ma thèse.

Je suis également reconnaissant à Leslie TROTTER qui a bien voulu s'intéresser à mon travail et présider le jury, ainsi qu'à Christian van DELFT qui m'a fait l'honneur d'être examinateur dans ce jury de thèse.

Je voudrais ensuite exprimer ma profonde reconnaissance à Michel MINOUX et à Yves DALLERY, mes deux directeurs de thèse. Avec leur soutien permanent, ils m'ont toujours aidé à résoudre les problèmes avec rigueur, ils m'ont écouté, aidé et formé tout au long de cette thèse.

Michel MINOUX est l'homme à qui je devrai toujours l'énorme chance d'avoir réalisé ma thèse au sein d'EDF R&D. Avec beaucoup de patience, il a passé plusieurs heures de réunion de travail avec moi, il m'a appris beaucoup de choses, il m'a toujours montré le bon chemin et il m'a monté le moral aux moments où ceci était nécessaire. Il mérite mon plus profond respect.

Yves DALLERY a su me guider et me conseiller, scientifiquement et amicalement, tout au long de ces travaux de recherche. Jean-Claude BOQUET et Yves DALLERY ont été les hommes qui m'ont permis de réaliser mes études de DEA et de thèse à l'École Centrale Paris. Qu'ils veuillent trouver ici l'expression de ma sincère reconnaissance.

Jean-Yves LUCAS a été mon encadrant industriel qui m'a suivi et m'a soutenu pendant ces trois ans. Je le remercie pour toutes ces discussions, scientifiques et autres, qui m'ont permis d'avancer dans mes travaux, qui m'ont fait réfléchir et qui m'ont

inspiré.

Il m'est bien évidemment impossible de ne pas citer Sylvain GUEDON, qui a toujours eu le temps pour m'écouter, me conseiller, me corriger et veiller avec Jean-Yves LUCAS à ce que cette thèse constitue, à part son caractère académique, aussi un outil qui permettra à EDF d'en tirer profit.

Je tiens à remercier tous les collègues à EDF et le chef du groupe I23 du département SINETICS, Olivier DUBOIS, dont la participation active au comité de pilotage de ma thèse a constitué une puissance motrice de mon travail.

Je remercie également tous les collègues à LGI, et plus spécialement les aimables secrétaires Anne PREVOT, Sylvie GUILLEMAIN et Corinne OLLIVIER, qui m'ont tellement facilité la vie avec leur bonne humeur, leur bonne volonté et leur constante disponibilité.

Je remercie spécialement les hommes avec lesquels j'ai passé la plus grande partie de mon temps pendant ces trois ans : mes voisins de bureau Ali CHEAITOU à LGI et Sébastien MEUNIER à EDF. Je n'aurai que de bons souvenirs.

Au plus bas de la pyramide se trouvent les fondements de tout ce que j'ai pu réaliser jusqu'à maintenant. J'exprime ma sincère gratitude à tous mes amis, à Eva STUDER, à mes parents, à mon frère et enfin à Georges SAHARIDIS qui, dans mon cœur, fera toujours partie de ma famille.

Table des matières

Avant Propos **1**

1 Introduction aux modèles de décision dans l'incertain **5**

1.1 Différents types d'incertitude et caractéristiques 5

1.2 Introduction à la programmation stochastique 7

1.3 Modèles avec des contraintes probabilistes 10

1.3.1 Contraintes probabilistes indépendantes 11

1.3.2 Contraintes probabilistes jointes 16

1.3.3 Espérance conditionnelle 16

1.3.4 Incertitude seulement dans la fonction objectif 17

1.4 Modèles de Recours 20

1.4.1 Différents cas de recours 22

1.4.2 Propriétés des programmes linéaires avec recours 26

1.5 Méthodes hybrides 27

1.6 Optimisation robuste 28

1.7 Distribution de l'optimum 30

1.8 Valeur espérée d'information parfaite (*Expected Value of Perfect Information – EVPI*) 31

1.9 Conclusions 32

2 Modèles en deux étapes **35**

2.1 Introduction 35

2.2 Méthode “L-Shaped” 36

2.2.1 ”L-Shaped” en version multi-coupes 41

2.3 Décomposition régularisée 42

2.4 Décomposition stochastique 46

2.5 Programmation mixte 50

2.6 Conclusions 51

3 Modèles multi-étapes **53**

3.1 Introduction 53

3.2 Modélisation du flux d’information 57

3.3 Décomposition imbriquée 59

3.4 L’algorithme de couverture progressive (*Progressive hedging*) 65

3.5 Méthode d’approximation quadratique diagonale (DQA) 69

3.6 Décomposition avec lagrangien augmenté 74

3.7 Recours itéré 78

3.8 Autres méthodes de décomposition 81

3.9 Méthodes de points intérieurs 84

3.10 Généralités sur les approches par échantillonnage 88

3.11 Conclusions 89

4 Application au dimensionnement de capacités **91**

4.1 Introduction 91

4.2 Les différentes sources d’aléas 94

4.3 Structure du réseau 96

4.4 Approvisionnement 98

4.5 Acheminement 99
4.6 Stockage 101
4.7 Une version simplifiée du problème 103
4.8 Modèle déterministe 105
4.9 Modèle stochastique 110
4.10 Conclusions 112

5 Nouvelles mises en œuvre de la décomposition imbriquée 115

5.1 Introduction 115
5.2 Structure d'un problème de dimensionnement 116
5.3 Algorithme de la décomposition imbriquée : recours complet 119
5.4 Réduction du temps de calcul passé sur une itération 124
5.4.1 Réduction du temps de calcul passé sur la résolution d'un nœud 124
5.5 Réduction du nombre d'itérations 127
5.5.1 Construction d'un ensemble de solutions candidates 128
5.5.2 Critère d'évaluation des solutions candidates 129
5.6 Expériences numériques 131
5.7 Résultats 133
5.7.1 Résultats concernant la réduction du temps de calcul par itération 133
5.7.2 Résultats concernant la réduction du nombre d'itérations 138
5.8 Bilan global des améliorations proposées 147
5.9 Conclusions 149

Conclusions – Perspectives 155

Bibliographie 158

A Échantillonnage au sein de la décomposition imbriquée 167

B Différentes manières d'échantillonner 175

C Résumé des dispositions légales applicables dans le secteur du gaz 179

C.1 Fournisseurs de gaz 179

C.2 Opérateurs de réseaux de transport de gaz 181

C.3 Opérateurs de réseaux de distribution de gaz 182

C.4 Opérateurs de stockage 182

C.5 Exploitants d'installations de gaz naturel liquéfié 182

C.6 Dispositions communes et diverses 182

Avant propos

" L'incertitude n'est pas dans les choses mais dans notre tête : l'incertitude est une méconnaissance[1] *"*. Malgré le fait que l'on puisse trouver bien des auteurs qui défendent la thèse selon laquelle les sources de l'incertitude font partie de la nature inhérente du système (l'incertitude se trouve donc dans les *choses*), on ne peut toutefois pas nier qu'aujourd'hui on dispose d'un certain nombre d'outils, nous permettant non pas de l'éradiquer, mais au moins de la maîtriser. La *maîtrise* de l'incertitude (qui se trouve dans *notre tête*) peut constituer le point qui démarquera celui qui la respecte de celui qui la méprise. Le premier peut en tirer du profit, le second ne fait que compter sur le hasard.

Le présent travail est motivé par une application réelle. Des sources de données qui jusqu'à hier pouvaient être considérées comme certaines, aujourd'hui sont remises en cause. La prise en compte des aspects aléatoires peut réduire considérablement le coût par rapport aux décisions statiques, implémentées jusqu'à hier, au détriment du temps de calcul pour obtenir une telle décision.

Notre travail est structuré de la façon suivante :

Le premier chapitre fait l'introduction dans le domaine de la programmation stochastique. Différents modèles de prise en compte de l'incertitude seront présentés, dont les modèles de recours, les contraintes probabilistes, la programmation robuste, et le problème de la distribution de l'optimum. Dans ce travail, on se focalisera plus précisément sur les modèles de recours, qui sont divisés, entre autres, en deux catégories : les modèles de recours en deux étapes et leur extension en plusieurs étapes.

[1] Jacques Bernoulli

Dans le deuxième chapitre on revisera quelques méthodes de décomposition concernant la résolution des problèmes pouvant être modélisés en tant que modèles de recours en deux étapes. Par ailleurs, à partir du deuxième chapitre, et pour le reste de ce travail, on considère travailler sur un espace d'incertitude décrit par un ensemble de scénarios. Ceci pourrait signifier soit que les scénarios nous sont donnés sans aucune information supplémentaire (par exemple historiques ou opinion des experts), soit que les variables aléatoires présentes dans les modèles sont discrétisées.

On étend notre étude des modèles en deux étapes aux modèles en plusieurs étapes, en introduisant au chapitre 3, les modèles de recours multi-étapes. La modélisation par arbre de scénarios ainsi que la modélisation avec les contraintes de non-anticipativité seront présentées et quelques méthodes, portant sur chacune des deux modélisations, seront également exposées. On se concentrera plus sur la décomposition imbriquée, la méthode de décomposition que l'on étudiera en détail plus loin dans ce travail.

Le chapitre 4 introduit l'application industrielle qui a motivé ce travail. Il s'agit d'une application réelle, issue du domaine énergétique, qui intéresse EDF, ainsi que tout autre acteur souhaitant participer au marché gazier, dont l'ouverture s'est effectuée en juillet 2004. Le modèle correspondant, en sa version déterministe, peut être exprimé en tant que programme linéaire. Sa version stochastique, comportant des termes aléatoires dans la fonction de coût et le second membre, présente d'une part une structure représentative de notre application, et d'autre part, est souvent rencontrée dans les problèmes de dimensionnement. Il s'agit des problèmes dont les décisions principales sont prises au début des périodes d'étude et influencent par conséquent toutes les autres décisions prises aux instants ultérieurs.

La structure particulière de l'application qui nous intéressera sera exploitée dans le chapitre 5. Les dimensions du problème augmentent très vite en fonction du nombre de scénarios (qui modélisent l'aléa), rendant les méthodes directes impraticables. La décomposition du problème semble être nécessaire. Nous développerons de nouvelles mises en œuvre de la décomposition imbriquée, qui comportent des techniques d'échantillonnage au sein de la méthode, le but étant le traitement de très grandes instances dont la taille ne peut pas être contenue dans la mémoire de la machine. Nous montrerons, à travers les expériences, que notre méthode devance systématiquement la décomposition

imbriquée standard sur les grandes instances, avec une amélioration moyenne qui se situe au dessus d'un facteur de 2.

On terminera ce travail avec les conclusions, faisant le bilan global de tout ce que l'on a pu capitaliser en termes de nouvelles idées sur l'échantillonnage au sein d'une méthode d'optimisation. Les perpectives porteront sur d'autres manières potentielles d'accélérer la résolution de grandes instances, ainsi qu'à la parallélisation de la procédure proposée.

Chapitre 1

Introduction aux modèles de décision dans l'incertain

1.1 Différents types d'incertitude et caractéristiques

Au cours des dernières années, des ingénieurs, des mathématiciens, des scientifiques et des décideurs ont manifesté un fort intérêt pour la prise en compte de l'incertitude. Différentes formes d'incertitude ont été identifiées dépendant de la nature de l'incertain ainsi que de la quantité d'information dont on dispose. Oberkampf et al [52] ont proposé un classement des différents types d'incertitude en trois catégories :

- Incertitude *aléatoire* ou *stochastique*
- Incertitude *épistémique*
- *Erreur*

L'incertitude aléatoire est l'effet d'une variation inhérente du système, issue de la nature aléatoire de certaines de ses données. Si on dispose d'un nombre d'expériences suffisant pour déduire une distribution probabiliste pour les données aléatoires, ou même si on dispose explicitement de cette loi de probabilité, alors on peut modéliser ces problèmes en se servant des éléments de la théorie des probabilités. Ce point de vue est couramment adopté dans de nombreux domaines. Il se peut que l'information sur une variable aléatoire ne soit pas donnée explicitement à travers sa fonction de répartition (ou sa moyenne, sa variance ou d'autres moments), ni à travers un en-

semble de scénarios, mais sous autre forme. Les données d'entrée peuvent être contenues dans des intervalles précis ou flous. De tels cas ne sont pas triviaux et la procédure d'estimation de l'incertitude associée à ces données inconnues peut s'avérer une tâche fastidieuse.

D'autre part, l'incertitude épistémique a ses sources dans l'information incomplète du système, l'ignorance de certains aspects ou dans le manque des données expérimentales. On peut aborder cette catégorie avec les outils des probabilités, mais il y a une certaine polémique concernant l'efficacité, ainsi que la justesse d'une telle approche.

L'erreur peut aussi d'une certaine manière être considérée comme étant une source *"d'incertitude numérique"*. C'est un défaut inévitable, parfois non négligeable, qui n'est pas issu d'un manque d'information, mais des limites de précision numériques lors de l'implémentation des calculs.

La première catégorie, comme on l'a dit, peut être abordée par la théorie des probabilités, alors que la troisième catégorie constitue un défaut que l'on ne peut pas éviter ; on peut au moins l'estimer. Le vrai défi selon [1] est de développer une méthodologie pour modéliser quantitativement l'incertitude épistémique. Le plus important inconvénient que l'on reproche à l'utilisation des méthodes probabilistes pour traiter des problèmes où l'incertitude est de type épistémique est qu'il n'y a pas de raisons particulières d'utiliser une distribution plutôt qu'une autre, tant que la distribution réelle qui gouverne le système nous est inconnue. D'autres méthodes, potentiellement plus performantes, ont été développées pour quantifier cette catégorie d'incertitude.

Tout système non-déterministe comprend de l'incertitude. Selon Klir et Smith [43], il y avait un à priori défavorable envers l'incertitude avant le $20^{\text{ème}}$ siècle, alors que la théorie des probabilités dominait sur tout ce qui comportait de l'incertain. D'un côté la généralisation de la théorie classique des ensembles en théorie des ensembles flous et d'un autre la généralisation de la théorie de la mesure en théorie des mesures monotones ont été développées depuis deux ou trois décennies pour que l'on ne puisse plus aujourd'hui affirmer que la théorie des probabilités est le seul cadre théorique capable de traiter l'incertitude.

En dehors de l'optimisation qui nous intéressera dans ce travail, d'autres domaines comme l'intelligence artificielle et les systèmes experts ont été intéressés par le

phénomène de l'incertitude, le but étant de construire des systèmes qui soient adaptables aux données d'entrée incertaines.

Dans ce cadre, des théories d'incertitude ont été développés, comme la théorie des ensembles flous (fuzzy theory), la théorie de Dempster-Shafer, les réseaux Bayésiens, la théorie des possibilités, et la théorie de prévisions inférieures et supérieures (*"theory of upper and lower previsions"*). La théorie des possibilités ainsi que la théorie des probabilités font partie de la théorie de l'évidence. Toutes ces théories nouvelles sur l'incertitude se résument dans l'article de Klir et Smith [43]. Le but est de comprendre, de représenter, d'agréger, de propager, de mesurer et enfin d'interpréter l'incertitude. Ces théories font partie de la grande famille de la *théorie généralisée de l'information* (*"generalized information theory"*).

La disponibilité ou le manque d'information est finalement le seul critère pour classifier l'incertitude dans une des catégories décrites ci-dessus. Creuser sur les sources d'information reste d'importance capitale. Les sources disponibles sont soit des informations statistiques complètes, soit des informations statistiques ou empiriques incomplètes ou incohérentes, soit des intervalles. Plus fréquents sont d'ailleurs les cas où un mélange des sources différentes nous est disponible. Il se peut par exemple que les intervalles dans lesquelles est contenu un paramètre incertain soient estimés à partir des historiques. Ou encore, étant donnée une distribution, déduite à partir de l'historique du phénomène qui semble lui être appropriée, sa moyenne soit déterminée par un expert. Un expert peut aussi donner son avis sur l'estimation d'un paramètre de façon imprécise mais pourtant naturelle, comme par exemple "*cette valeur est plus vraisemblable que celle-ci*" ou "*ceci ne me semble pas trop probable*".

1.2 Introduction à la programmation stochastique

La notion d'incertitude dans la programmation mathématique est apparue pour la première fois dans les années '50 avec les travaux de Bellman, Dantzig [27], Cooper et Charnes [25], Beale [4], et elle a rencontré depuis un développement rapide. La figure 1 montre la tendance en termes de publications d'articles [72] depuis le début de la programmation stochastique jusqu'à nos jours. Dans la suite on parlera d'incertitude dans

le sens d'incertitude aléatoire (comme définie par [52]), où une partie de l'information nécessaire pour la compréhension complète d'un phénomène est inconnue. L'incertitude dans les problèmes d'optimisation touche notamment les coûts de production, les prix des marchés, les pénalités en cas des violations des contrats, aussi bien que la demande de clients, les délais de livraison, les temps de traitement, la disponibilité des machines et d'autres coefficients technologiques.

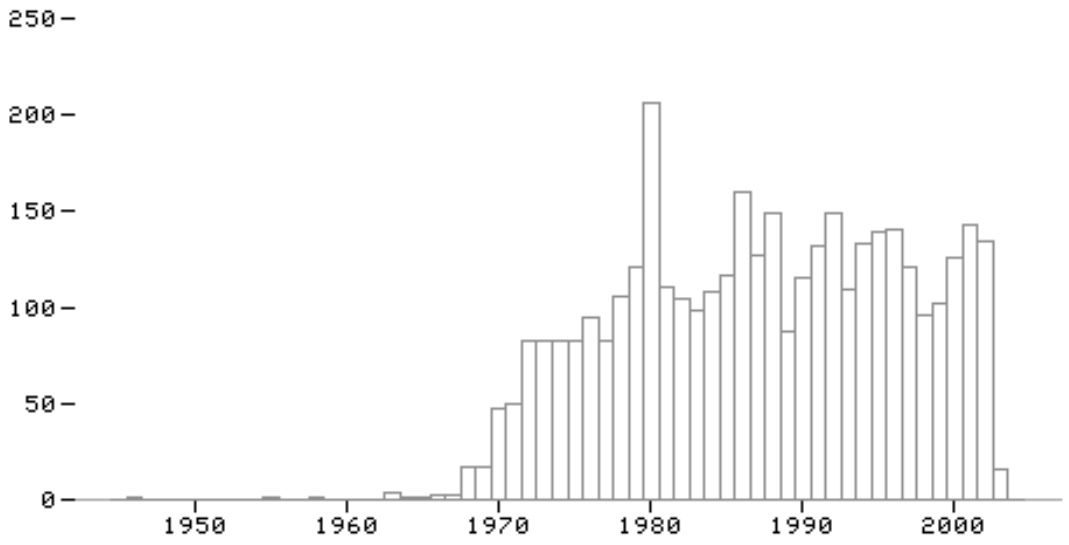


Fig. 1: Nombre de publications annuelles depuis les années '50 jusqu'en mai 2003.

Il se peut que l'on connaisse partiellement certains aspects du phénomène à travers soit des scénarios, soit un historique, soit une loi de probabilité, soit des moments de la variable aléatoire (par exemple espérance mathématique, variance). Des scénarios peuvent être crées à partir des historiques, (comme par exemple les ventes des derniers années, les températures de la dernière décennie, la mortalité d'une certaine population), ou bien à partir de l'opinion d'experts qui peuvent prévoir le comportement du phénomène incertain. Si la loi de probabilité de la variable aléatoire est connue de façon analytique, on peut soit créer des méthodes analytiques (la plupart des fois ceci est un travail non-trivial) qui tiennent compte de cette loi de probabilité, soit générer un ensemble de scénarios suivant cette loi.

Le but que l'on se fixe dans un problème d'optimisation est de prendre la *meilleure* décision vis à vis des situations qui comportent de l'incertitude. Le regard vers l'incertitude reste subjectif et varie d'un décideur à l'autre. On peut par exemple souhaiter que dans l'espace de tout évènement possible, la solution adoptée soit telle que son coût soit minimal en moyenne. Ainsi, les approches de l'incertitude comprennent des techniques orientées vers la minimisation (maximisation) de l'espérance mathématique (ou d'autres moments), la minimisation de la déviation par rapport aux cibles (*deviation from goals*), la minimisation des coûts maximaux, la minimisation de l'écart entre le meilleur et le pire cas, ou même la simple satisfaction des contraintes avec une probabilité donnée d'avance qui représente le taux de fiabilité du système.

Lorsque l'on parle de la programmation stochastique, on entend la programmation mathématique sous incertitude. La programmation stochastique n'est pas une famille de problèmes, de modèles et d'outils différents des autres domaines d'optimisation (programmation linéaire, non-linéaire, dynamique) ; au contraire, elle constitue un complément de ces familles lorsque la notion d'incertitude intervient. On se permet donc de parler de programmation stochastique linéaire, non-linéaire ou bien de programmation stochastique dynamique. Dans le présent travail on s'intéresse à la programmation stochastique linéaire ainsi qu'à la programmation stochastique dyanamique.

Les approches qui dominent sur la modélisation et la résolution des problèmes de la programmation stochastique sont les suivantes :

- Modèles avec des contraintes probabilistes.
- Modèles de recours.
- Modèles robustes.

Les travaux de [66] et [16] présentent l'état de l'art de la programmation stochastique, alors que [39] se focalise aux méthodes de résolution de programmation stochastique en nombres entiers. La littérature autour de la programmation stochastique devient de plus en plus riche, les ouvrages les plus importants dans le domaine étant ceux de [55], [18], [64] et [37].

Dans le présent chapitre on présentera brièvement les différentes approches employées dans la programmation stochastique et on citera des références pour une explication plus détaillée. Dans les paragraphes qui suivent on s'intéressera d'abord aux

modèles probabilistes en donnant un petit exemple démonstratif. D'autres cas de modélisation seront introduits. On passera ensuite à la grande famille des modèles de recours et on étudiera également leurs propriétés. Une brève description des modèles robustes suivra dans le paragraphe 1.6 ainsi que le problème de distribution de l'optimum au paragraphe 1.7 suivant. Au paragraphe 1.8 on introduira la notion importante de la valeur espérée d'information parfaite. On terminera le chapitre avec les conclusions que l'on en tirera.

1.3 Modèles avec des contraintes probabilistes

Des problèmes dont la modélisation appartient à la catégorie des modèles probabilistes sont apparus pour la première fois dans [25] sous le nom de *programmation sous contraintes probabilistes* (*"chance constrained programming"*). Soit le problème d'optimisation linéaire suivant :

$$\begin{aligned} &\min\ c^T x \\ &\textit{sous contraintes :} \\ &Tx \geq \xi \\ &Ax = b \\ &x \geq 0 \end{aligned} \tag{1.1}$$

ξ est une variable aléatoire[1]. Admettons que la fonction de coût ne dépend pas de la variable aléatoire ξ. La contrainte $Tx \geq \xi$ pourrait s'exprimer comme

$$\max\ \mathbb{P}(\,Tx\ \geq\ \xi)$$

ou encore :

$$\mathbb{P}(\,Tx\ \geq\ \xi) \geq p \tag{1.2}$$

p étant une probabilité réglée par l'utilisateur qui reflète par exemple la fiabilité d'un système ou une tolérence acceptable. Ces contraintes s'appellent *jointes* ou *simultanées*.

[1] Une variable pouvant contenir plusieurs composantes aléatoires, on l'appelera également *vecteur aléatoire*.

L'interprétation des contraintes (1.2) reflète l'exigence que la contrainte $Tx \geq \xi$ soit satisfaite dans $p\%$ des cas, et pas nécessairement toujours. Surtout pour un problème où la non-satisfaction d'une contrainte est tolérée dans $p\%$ des cas, une éxigeance de p=100% aurait été trop contraignante. Ceci est souvent le cas dans les problèmes de dimensionnement et de fiabilité où l'exigence par rapport aux probabilités de défaillance se modélise avec les contraintes probabilistes.

Un problème plus facile à traiter serait de considérer l'incertitude sur chaque contrainte séparément et ainsi obtenir le système :

$$\mathbb{P}(\, T_i x \,\geq\, \xi_i) \geq p_i \tag{1.3}$$

T_i représentant la ligne i de la matrice T et p_i la probabilité que cette contrainte soit satisfaite. Ces contraintes s'appellent également *"contraintes point à point"*.

Le choix entre les modélisation (1.2) et (1.3) dépendra du fait que les contraintes sont indépendantes les unes des autres ou non. Ceci est imposé par le problème lui même. Si par exemple les contraintes portent sur la même entité (par exemple un ensemble de machines dans une usine dont les temps de traitement dépendent de la puissance électrique des générateurs de l'usine) la modélisation (1.2) semble être plus appropriée. Si pourtant les entités correspondant aux contraintes sont bien distinctes la formulation (1.3) est plus exacte.

Par ailleurs, si les composantes du vecteur aléatoire ξ sont indépendantes, on pourrait déduire :

$$\mathbb{P}\{\, Tx \,\geq\, \xi\} \geq p \quad \Leftrightarrow \quad \mathbb{P}\{\, T_1 x \,\geq\, \xi_1\} \cdot \ldots \cdot \mathbb{P}\{\, T_n x \,\geq\, \xi_n\} \geq p \tag{1.4}$$

ce qui diffère de (1.3).

1.3.1 Contraintes probabilistes indépendantes

On note ici que le terme *indépendantes* porte sur les contraintes probabilistes et non pas – nécessairement – sur la variable aléatoire. Soit le problème suivant :

$$
\begin{aligned}
&\min c^T x \\
&\textit{sous contraintes :} \\
&\mathbb{P}\{\xi \,|\, T_i x \geq h(\xi_i)\} \geq p_i \quad \forall\, i \\
&Ax = b \\
&x \geq 0
\end{aligned}
\tag{1.5}
$$

où $h(\xi_i)$ suit la même distribution que la variable aléatoire ξ_i, et T_i est la i-ligne de la matrice technologique T. On note bien que pour l'instant la matrice T est déterministe.

La contrainte probabiliste (1.3) (que l'on retrouve dans le programme (1.5)) peut être remplacée par sa forme équivalente :

$$
\mathbb{P}\{T_i x \geq h(\xi_i)\} \geq p_i \quad \Leftrightarrow \quad F_i(T_i x) \geq p_i \quad \Leftrightarrow \quad T_i x \geq F_i^{-1}(p_i) \tag{1.6}
$$

et donc le problème devient trivial, puisque la valeur $F_i^{-1}(p_i)$ pour les différents p_i est déterministe.

On va maintenant supposer que la matrice technologique est stochastique, c'est-à-dire qu'elle comprend des éléments qui ne sont connus qu'à travers une loi de probabilité. Le second membre reste cette fois déterministe. Le modèle de Kataoka [65] peut constituer une première approche pour étudier cette catégorie grace à sa simplicité. On considère le problème suivant :

$$
\begin{aligned}
&\min c \\
&\textit{sous contraintes :} \\
&\mathbb{P}\{\xi^T x \geq c\} \geq a \\
&\mathbb{1}^T x = M \\
&x \geq 0
\end{aligned}
\tag{1.7}
$$

ξ étant un vecteur aléatoire de dimension n dont la distribution est supposée normale, $x \in \mathbb{R}^n$ et $\mathbb{1}^T$ représentant le vecteur de dimension n ayant toutes ses composantes égales à 1. De plus, soit μ_i la moyenne de toute composante i du vecteur ξ_i et C la matrice de covariance.

On effectue un changement de variable pour passer à la distribution normale centrée réduite :

$$z = \frac{\xi^T x - \mathbb{E}\{\xi^T x\}}{\sqrt{var(\xi^T x)}} \tag{1.8}$$

z étant la variable aléatoire suivant la distribution normal centrée réduite $\mathcal{N}(0,1)$.

En substituant

$$\mathbb{E}\{\xi^T x\} = \mu^T x \qquad var(\xi^T x) = x^T C x$$

on obtient :

$$\mathbb{P}\{\xi^T x \geq c\} = \mathbb{P}\{\frac{(\xi-\mu)^T x}{\sqrt{x^T C x}} \geq \frac{d-\mu^T x}{\sqrt{x^T C x}}\} = 1 - \Phi(\frac{d-\mu^T x}{\sqrt{x^T C x}}) \tag{1.9}$$

où Φ la fonction de répartition de la variable z.

La forme équivalente de la contrainte probabiliste du problème (1.7) est donc la suivante :

$$\begin{array}{l} 1 - \Phi(\frac{d-\mu^T x}{\sqrt{x^T C x}}) \geq a \Rightarrow \\ \mu^T x + \Phi^{-1}(1-a)\sqrt{x^T C x} \geq c \end{array} \tag{1.10}$$

qui correspond à un espace convexe si les conditions suivantes sont remplies :

1. C est une matrice semidefinie positive. Quand la fonction de répartition est normale multivariée, ceci est le cas ;
2. $\Phi^{-1}(1-a) \geq 0$,[2] ce qui se produit si et seulement si $p \geq \frac{1}{2}$.

Remarquons que si $x^T C x$=0, en d'autres termes, si les variables ξ_i sont indépendantes, alors la contrainte revient à être $\mu^T x \geq c$, qui est en fait la contrainte probabiliste où on a remplacé les variables aléatoires par leurs moyennes.

On peut suivre des étapes similaires au cas où la partie aléatoire est contenue dans la fonction objectif.

Si on a plusieurs contraintes strochastiques, on répète la procédure décrite ci-dessus pour toutes les lignes T_i de la matrice T. On rappelle que l'on est dans le cas où les variables aléatoires intervenant dans les contraintes sont indépendantes. Par ailleurs on peut substituer la condition que la variable aléatoire suit la loi normale,

[2] On rappelle que si f est une fonction convexe, $-f$ est concave. Donc $kf(x)$ est convexe si $k \geq 0$.

par la condition que sa distribution soit log-concave ([54]). Il faut également noter que la plupart des distributions connues (par exemple uniforme, normale, wishart, beta, dirichlet, cauchy, pareto, gamma) sont des distributions log-concaves.

Exemple

Supposons que l'on dispose de deux dés dont les réalisations possibles ξ *et* ω *appartiennent à l'espace* $\{1,2,3,4,5,6\}\times\{1,2,3,4,5,6\}$ *avec une probabilité pour chaque évènement (couple de réalisations) de* $1/36$. *Le jeu que l'on se propose consiste à lancer les dés de telle façon que la somme de deux résultats soit supérieure à 5. Dans ce cas, on remporte un gain égal à 25 unités monétaires, alors qu'au contraire, on a aucun bénéfice. Pour que le jeu ne soit pas du pur hasard, on dispose de deux leviers sur les valeurs réalisées par les dés : considérons deux variables* x *et* y *appartenant à l'espace* $\mathbb{R}$. *On est libre de choisir les valeurs de ces deux variables, qui vont être multipliées repsectivement par la réalisation de chacune des deux dés. Par exemple, choisissons* x=1 *et* y=2.5 *et supossons que l'on lance les deux dés, le résultat étant* ξ=3, ω=2. *En calculant* $x\xi + y\omega$ *on prend la valeur 8 qui est bien supérieure à 5. On aurait intérêt de choisir des valeurs* x *et* y *très grandes (*x=5 *et* x=5 *aurait été suffisant) pour s'assurer que l'on arrive toujours à la valeur donnée, 5. En revanche, les règles imposent que pour chaque unité de la valeur* x *on est obligé de céder 3 unités monétaires de notre gain, et donc gagner* $3x$ *de moins (*$25-3x$*). Il en va de même pour la variable* y *où le coût de ce levier est de 2 unités monétaires. Par conséquent, en règlant les variables* x *et* y *chacune à 5 ne nous fait rien gagner. Quelles valeurs* x, y *faut-il choisir ?*

Le plus naturel serait d'essayer de céder le minimum de gain avec un minimum de risque. Évidemment la solution souhaitée constituerait le meilleur compromis entre ces deux facteurs. En parlant du risque, on peut se donner des seuils, c'est-à-dire un niveau de risque de perte acceptable. Un risque de 10% entendrait que dans 10% des cas possibles, on perd le jeu. Le décideur se pose ses propres limites par rapport au risque assumé. Considérons le cas du risque de 10%.

On cherche alors ces variables x, y *qui satisfont la contrainte suivante :*

$$\xi x + \omega y \geq 5 \quad \textit{dans 90\% des cas} \tag{1.11}$$

Dans ce simple exemple avec des variables aléatoires discrètes on pourrait énumérer toutes les réalisations possibles de deux dés. Donc, pour tout couple (x, y), *on saurait combien de cas auraient été satisfaits, et ainsi on serait en mesure de décider si la solution proposée est comprise dans notre intervalle de risque. Cette tâche s'effectue en introduisant une variable binaire* z *qui mesure le nombre de violations de contraintes. Elle est activée dès qu'une violation se produit, et elle est désactivée sinon.*

$$\begin{array}{l} \min\ 3x + 2y \\ \textit{sous contraintes :} \\ \xi_i x + \omega_i y + 5z_i \geq 5 \quad \forall\, i \in |\Xi \times \Omega| \\ \sum_i p_i z_i \leq 0.1 \\ x, y \in \mathbb{R}_+ \\ z_i \in \{0,1\} \quad \forall\, i \end{array} \tag{1.12}$$

p_i *étant la probabilité ou l'importance accordée à chaque réalisation des variables aléatoires (autrement dit, à chaque scénario), i.e 1/36 dans cet exemple,* Ξ *et* Ω *étant l'ensemble* $\{1,2,...,6\}$. *Le coefficient de* $z \in \mathbb{R}^{36}$ *dans*

le premier bloc de contraintes est choisi de telle manière que l'inégalité soit toujours satisfaite, même dans le cas où x=y=0.

La modélisation (1.12) *est une modélisation linéaire en nombres entiers, et le problème résultat peut être résolu avec des logiciels d'optimisation déterministe.*

Le cas des variables continues

On remplace les dés par un générateur de nombres aléatoires prenant leurs valeurs dans l'intervalle $[1, 6]$*. On ne parle plus de variables discrètes mais continues. Le nombre de scénarios (réalisations) n'est donc pas dénombrable, ce qui ne nous permet pas d'énumérer toute réalisation possible pour construire autant de contraintes. L'interprétation de la contrainte* (1.11) *peut être donnée par la forme* (1.13) *:*

$$\mathbb{P}\{\xi x + \omega y \geq 5\} \geq 0.9 \tag{1.13}$$

Le fait que la probabilité que $\xi x + \omega y \geq 5$ *doit être supérieure à 90% signifie que si on avait la possibilité de générer* N *couples* (ξ, ω) *aléatoires, les solutions* (x, y) *devraient être telles que la* (1.13) *soit satisfaite pour* $0.9N$ *cas, ce qui donne une probabilité de 90% de satisfaction de la contrainte. On rappelle qu'un risque de 0% correspondrait à un choix qui répond au pire scénario, donc au scénario* ξ=1 *et* ω=1*.*

On considère la contrainte (1.13)*. On souhaite passer de cette forme à une forme déterministe. À ce point là, il nous faut des hypothèses sur les variables* ξ *et* ω*. On va suggérer, ce qui intuitivement semble être vrai, que ces deux variables sont indépendantes, bien que le cas des variables dépendantes ne change pas dramatiquement la difficulté du problème (cf au modèle de Kataoka)*[3]*. On fait encore l'hypothèse que les variables suivent la loi normale avec moyenne* $\bar{\xi}$ *et* $\bar{\omega}$ *et variance* σ_ξ *et* σ_ω *respectivement. Pour passer à la distribution centrée réduite on effectue l'opération de changement de variable :*

$$\begin{array}{l} z = \frac{(\xi-\bar{\xi})x+(\omega-\bar{\omega})y}{\sqrt{x^2\sigma_\xi^2+y^2\sigma_\omega^2}} \qquad \Leftrightarrow \\ \xi x + \omega y = z\sqrt{x^2\sigma_\xi^2 + y^2\sigma_\omega^2} + \bar{\xi}x + \bar{\omega}y \end{array} \tag{1.14}$$

Dès lors, la forme (1.13) *est équivalente à la forme suivante :*

$$\begin{array}{l} z_{10\%}\sqrt{x^2\sigma_\xi^2 + y^2\sigma_\omega^2} + \bar{\xi}x + \bar{\omega}y \leq 5 \qquad \Leftrightarrow \\ \bar{\xi}x + \bar{\omega}y \leq 3 - 0.25\sqrt{x^2\sigma_\xi^2 + y^2\sigma_\omega^2} \end{array} \tag{1.15}$$

Voici alors la contrainte déterministe que l'on voulait ressortir, pourtant elle n'est manifestement pas linéaire.

△

[3] Les oprérations restent les mêmes ; ce ne sont que les termes au dénominateur qui changent en faisant intervenir la covariance.

1.3.2 Contraintes probabilistes jointes

On étend le modèle de Katoka [65] au cas où la matrice technologique T comprend plusieurs lignes. La forme générique de tels problèmes qui contiennent des paramètres incertains dans la matrice technologique, ainsi qu'au second membre est la suivante.

$$\begin{aligned} &\min c^T x \\ &\textit{sous contraintes :} \\ &\mathbb{P}\{\xi \,|\, T(\xi)x \leq 0\} \geq p \\ &Ax = b \\ &x \geq 0 \end{aligned} \tag{1.16}$$

Dans la forme (1.16) on a également pris en compte l'aléa au second membre en considérant :

$$(T, -\xi) \cdot \begin{pmatrix} x \\ x_{n+1} \end{pmatrix} \leq 0$$

et dans le polyhèdre $Ax{=}b$ on considère avoir la contrainte $x_{n+1}{=}1$. Le vecteur x est de dimension n et la matrice T de dimensions $m \times n$.

La convexité des contraintes jointes n'est pas triviale. De simples exemples peuvent être décrits dont l'espace de solutions n'est pas nécessairement convexe ([37]). La convexité des espaces de solutions issus des contraintes probabilistes peut être assurée dans certains cas (cf à [55]) où la distribution jointe des éléments de la matrice T est normale. On se réfère à [55] et aux différents travaux cités pour des méthodes de résolution des modèles avec des contraintes probabilistes jointes.

1.3.3 Espérance conditionnelle

Une autre approche de modélisation serait de mesurer la moyenne des violations d'une contrainte et d'exiger que elle ne dépasse pas un seuil d. On reprend le problème (1.1) et on étudie la contrainte portant sur la variable aléatoire ξ. L'espérance mathématique $\mathbb{E}\{\xi_i - T_i x\}$ exprime la moyenne de la violation de la contrainte correspondante (que la contrainte soit violée ou pas). On pourrait également s'intéresser

exclusivement à la violation moyenne aux cas où on a violé la contrainte, c'est-à-dire :

$$\mathbb{E}\left\{\xi_i - T_i x | \xi_i - T_i x > 0\right\} < d_i, \quad i = 1, 2, \ldots, r. \tag{1.17}$$

Dans (1.17) on demande que la moyenne des violations de la contrainte i – aux cas où la contrainte a été effectivement violée – ne dépasse pas la valeur d_i.

La forme (1.17) est toujours une forme stochastique. Pour passer en forme déterministe, il faut se servir d'un théorème [55] (bien connu dans la théorie de fiabilité) qui stipule que si ξ_i a une distribution de probabilité continue en $\mathbb{R}^1$, si sa densité de probabilité est concave, et si $\mathbb{E}\{\xi\}$ existe, alors $\mathbb{E}\left\{\xi_i - T_i x | \xi_i - T_i x > 0\right\}$ existe également et elle décroît en fonction de $T_i x > 0$.

En introduisant la notation [55] :

$$L_i(T_i x) = \mathbb{E}\left\{\xi_i - T_i x | \xi_i - T_i x > 0\right\}, \quad i{=}1, \ldots, r. \tag{1.18}$$

on aboutit au résultat

$$L_i(T_i x) \leq d_i \quad \Leftrightarrow \quad T_i x \geq L_i^{-1}(d_i), \quad i{=}1, \ldots, r. \tag{1.19}$$

qui est un système des contraintes linéaires. On note que la (1.19) est obtenue par le fait que la $L_i(T_i x)$ est décroissante en fonction de $T_i x > 0$.

1.3.4 Incertitude seulement dans la fonction objectif

Dans la cas où l'aléa n'intervient que dans la fonction objectif, ce qui est mis en question n'est pas la réalisabilité du programme, puisque le domaine des solutions réalisables n'est pas affecté par les éléments aléatoires. Ce que l'on considère est la qualité de la solution obtenue et d'une certaine manière sa robustesse, à savoir les variations en termes de coûts pour différentes réalisations de la variable aléatoire.

On considère la version générique suivante :

$$\begin{aligned} &\min h(x, \xi) \\ &\textit{sous contraintes :} \\ &Ax = b \\ &x \geq 0 \end{aligned} \tag{1.20}$$

Il existe différentes manières de traiter ce type de problèmes : (i) minimiser l'espérance de la fonction objectif, (ii) appliquer un modèle du type Markowitz [48] ou (iii) convertir le modèle en contrainte probabiliste.

La minimisation de l'espérance conduirait au problème (1.21) qui semble être la façon la plus simple et naturelle pour traiter le cas où l'incertitude n'affecte que la fonction objectif. Or, ceci peut amener à des décisions dont le coût réel ne soit pas reflété par le coût espéré du modèle.

$$\begin{aligned} &\min \mathbb{E}\left\{h(x,\xi)\right\} \\ &\textit{sous contraintes :} \\ &Ax = b \\ &x \geq 0 \end{aligned} \tag{1.21}$$

Le danger derrière cette modélisation se positionne sur les deux aspects suivants :

- Pour que la valeur de la fonction objectif du problème (1.21) ait un sens réel, il faut que le système fonctionne sur un long horizon d'étude. Ce n'est seulement que dans ce cas, que l'on pourrait prétendre que le coût moyen des résultats $h(x^*, \xi^j)$ tende vers l'espérance $\mathbb{E}\left\{h(x,\xi)\right\}$.
- Si la variance des résultats $h(x^*, \xi^j)$, pour différentes réalisations j, est importante – voire infinie dans certains cas extrêmes -, minimiser l'espérance n'a aucune importance. Pratiquement, et à titre d'exemple, cela signifierait que la résolution du problème (1.21) nous fournit la décision optimale x^* qui minimise d'une part le coût moyen sur un horizon long, mais d'une autre part pour une certaine réalisation ξ^j, certes d'une probabilité faible mais non nulle, le coût devient trop élevé.

Les modèles du type Markowitz [48] remedient à ce problème de la grande variation des coûts de solutions : ici on minimise d'une part l'espérance des coûts mais on veille également à ce que la variation soit également minimisée. Une règle simple d'un modèle du type Markowitz est la suivante (μ représente la moyenne et σ l'écart type) :

$$\begin{aligned} &\text{Si } \exists\, x : \mu(x) = \mu(x^*) \text{ alors il faut que } \sigma(x) > \sigma(x^*) \quad \text{ou} \\ &\text{Si } \exists\, x : \sigma(x) = \sigma(x^*) \text{ alors il faut que } \mu(x) > \mu(x^*) \end{aligned}$$

Ceci signifie pour notre décision optimale x^*, qu'il ne faut pas y avoir une autre décision

x qui a le même coût moyen et une variance plus faible, ou une autre décision qui a la même variance que la notre mais un meilleur coût moyen (problème de minimisation). Un modèle du type Markowitz pourrait être le suivant :

$$\begin{array}{l} \min \quad \alpha\mathbb{E}\left\{h(x,\xi)\right\} + \beta\sqrt{Var\left\{h(x\,\xi)\right\}} \\ \textit{sous contraintes :} \\ Ax = b \\ x \geq 0 \end{array} \tag{1.22}$$

Pour une étude de ce type de modèles on se réfère aux travaux récents de Ruszczyński et Shapiro [60].

Une dernière approche serait de transformer la fonction objectif en la contrainte probabiliste suivante :

$$\begin{array}{l} \min d \\ \textit{sous contraintes :} \\ \mathbb{P}\{h(x,\xi) \leq d\} \geq p \\ Ax = b \\ x \geq 0 \end{array} \tag{1.23}$$

où p une probabilité donnée à l'avance, le plus souvent très grande.

Le choix de la probabilité p est pourtant un choix abstrait. Or, notons que ceci est une caractéristique générale des problèmes qui comportent des paramètres incertains seulement à la fonction objectif. L'approche de la minimisation de l'espérance est souvent inappropriée comme on l'a expliqué auparavant. Employer un modèle du type Markowitz pour controler la variance comporte également le choix des paramètres α et β qui, lui aussi, est un choix abstrait.

Dans cette famille de modèles, le preneur de décision a le premier rôle lors de la modélisation de son problème. C'est lui qui détermine les paramètres reflétant son attitude vis à vis du risque. Dans cette direction, on trouve également une large littérature autour des *fonctions d'utilité*. La fonction objectif à optimiser est modélisée de telle manière que le résultat corresponde plus aux attentes du décideur ([76]).

1.4 Modèles de Recours

La programmation stochastique s'intéresse essentiellement à des problèmes où on doit prendre une décision sur le champ sous présence d'incertitude, sans attendre la réalisation des certaines variables aléatoires ; on fait souvent appel à ce type de problèmes sous le nom de *"here and now problems"*.

Dans la famille *"here and now"* on est obligé de déterminer au moment t_0 certaines valeurs x_0^j qui ne peuvent plus changer dans la suite, quoi qu'il arrive aux instants ultérieurs. Il se peut qu'une décision à l'instant initial t_0 coûte trop cher dans le futur, cet effet faisant émerger la nécessité d'une approche systématique de la prise de telles décisions.

Dans la suite on étudiera plus particulièrement la programmation stochastique linéaire avec recours. L'idée est d'optimiser les décisions que l'on doit prendre au moment t_0 tout en tenant compte des conséquences de ces décisions pour toute réalisation des aléas qui pourrait se produire aux instants ultérieurs. Le mot *"recours"* révèle la possibilité dont on dispose de remédier à une décision éventuellement trop optimiste à l'instant t_0, en mettant en œuvre des actions correctives, évidemment plus chères, qui satisfont pourtant les contraintes posées aux instants ultérieurs t_i, $i > 0$. On précise qu'après avoir pris la décision au moment t_0, on passe au moment t_1 où tous les évènements inconnus jusqu'à cet instant se révèlent. Par conséquent, l'étude de tels cas a un intérêt seulement si on dispose de quelques informations sur les réalisations des évènements qui nous sont inconnus au moment t_0 (information comme par exemple la distribution de probabilité, des scénarios, la moyenne, la variance etc).

Le problème peut s'étendre en plusieurs étapes suivant l'horizon d'étude, mais considérons d'abord le cas de deux étapes seulement. Les variables de la première étape sont considérées comme étant des variables structurantes, stratégiques qui ne peuvent plus changer à la seconde étape. La décision à la première étape doit être prise en respectant des contraintes propres à cette étape. La seconde étape comprend des actions (pénalisantes) qui peuvent être entreprises afin de satisfaire des contraintes qui font intervenir des variables de la première étape. Toutes ces idées sont résumées dans le programme suivant :

$$\begin{array}{ll} & \min\ c^T x + \mathbb{E}_\xi \left\{ Q(x,\xi) \right\} \\ \text{s.c.} & Ax = b \\ \text{avec} & Q(x,\xi) = \min_{y \in \mathcal{Y}}\ q^T(\xi) y(\xi) \\ \text{s.c.} & T(\xi)x + W(\xi)y(\xi) = h(\xi) \end{array} \tag{1.24}$$

où x et c sont des vecteurs de $\mathbb{R}^{n_1}$, A est une matrice de dimension $m_1 \times n_1$, $b \in \mathbb{R}^{m_1}$ le vecteur du second membre, et de façon similaire $y, q \in \mathbb{R}^{n_2}$, $h \in \mathbb{R}^{m_2}$ le vecteur aléatoire qui dépend de la variable aléatoire $\xi \in \mathbb{R}^r$, $\mathbb{E}_\xi$ l'espérance mathématique par rapport à la variable aléatoire ξ, alors que T et W sont des matrices dépendant de la variable ξ et de dimensions appropriées ($T : m_2 \times n_1$ et $W : m_2 \times n_2$).

Les variables x représentent les décisions avant que toute réalisation de la variable aléatoire ξ ne soit connue, alors que les variables y représentent les décisions prises, une fois que ξ sera observé. On cherche à minimiser sur les variables de la première étape $x \in \mathcal{X}$, mais on veille en même temps à ce que les coûts au second niveau, après la réalisation de la variable aléatoire ξ, soient minimaux *en moyenne*.

Il faut bien noter que les variables x de la première étape resteront les mêmes quelle que soit la réalisation de la variable aléatoire ξ, alors que les variables y de la seconde étape dépendront de cette réalisation, puisque elles cherchent à minimiser ses conséquences. Le fait que l'on ne se permet pas de faire dépendre x de ξ, c'est-à-dire que pour tout ξ_i d'avoir une décision x_i différente, s'appelle la *non-anticipativité* et c'est la propriété qui couple les variables du premier avec celles du second niveau. Si on ne tenait pas compte de cette propriété, alors pour toute réalisation de la variable aléatoire ξ_i on changerait à volonté les variables de la première étape, en choisissant chaque fois une valeur x_i telle qu'elle minimise la fonction de coût.

Dans de nombreuses applications on poura supposer qu'il existe une mesure de probabilité $\mathbb{P}$ telle que $\mathbb{P}\{\xi = \xi^s\} = p_s$ avec un support[4] fini Ξ donné par $\Xi = \{\xi^1, \xi^2, \cdots \xi^S\} \subset \mathbb{R}^r$. L'ensemble des réalisations possibles Ξ est fini, et les réalisations ξ^i s'appellent souvent des *scénarios*, dont p est la probabilité d'occurrence ou bien l'importance que l'on leur accorde. Cette hypothèse va nous permettre de passer de (1.24) au programme suivant :

[4]Avec le terme *support* on entend le plus petit ensemble fermé $\Xi \subset \mathbb{R}^p$ tel que $\mathbb{P}\{\Xi\}$=1

$$
\begin{array}{lccccccccccc}
\min_{x\in\mathcal{X},y^s\in\mathcal{Y}^s} & cx & + & p^1q^1y^1 & + & p^2q^2y^2 & + & \cdots & + & p^Sq^Sy^S & & \\
& Ax & & & & & & & & & = & b \\
& T^1x & + & W^1y^1 & & & & & & & = & h^1 \\
& T^2x & & & + & W^2y^2 & & & & & = & h^2 \\
& \vdots & & & & & & \ddots & & & & \\
& T^Sx & & & & & & & + & W^Sy^S & = & h^S \\
& x\in\mathcal{X} & & y^1\in\mathcal{Y}^1 & & y^2\in\mathcal{Y}^2 & & \cdots & & y^S\in\mathcal{Y}^S & &
\end{array}
\tag{1.25}
$$

où les indices $1, 2, ...S$ font référence aux scénarios correspondants. Par ailleurs dans notre cas, $\mathcal{X}$ désigne un polyhèdre convexe construit par un ensemble de contraintes linéaires.

Dans le programme (1.25) on distingue bien la structure d'un modèle avec recours. L'impact des variables x de la première étape sur la seconde étape devient évident, puisque les variables x rendent le problème non-décomposable. C'est la propriété de la non-anticipativité dont on a déjà parlé.

On s'aperçoit comment un tel programme peut déjà avoir de grandes dimensions. Le nombre de scénarios S peut être grand, puisque si on a V variables aléatoires avec S_v valeurs possibles chacune, on obtient $S{=}\prod_{v=1}^{V} S_v$ scénarios. Par exemple, pour $V{=}10$ variables aléatoires (demande de 10 clients), avec $S_v{=}5$ valeurs possibles chacune, on obtient $S{=}5^{10}$ scénarios, déjà pour un problème simple.

Or, si on relaxait la propriété de la non-anticipativité, la structure du programme serait beaucoup plus abordable, puisque l'on pourrait le décomposer en S sous-problèmes indépendants les uns des autres. Des méthodes de décomposition existent de plusieurs types et elles seront étudiées au chapitre 2.

1.4.1 Différents cas de recours

En observant la formulation (1.24) on peut distinguer différents cas de recours en fonction des types de (q, y, W) ou plus généralement des $q(\xi)$, $y(\xi)$, et $W(\xi)$. Les cas qui méritent une étude plus spéciale et qui sont cités dans la littérature sont le recours fixe, le recours complet et le recours simple.

Définition 1.1 (Recours fixe) *Le recours est considérée comme étant fixe ou déterministe si les valeurs q et W d'un programme avec recours comme le* (1.24) *ne dépendent pas de ξ, leurs valeurs sont donc à priori connues.*

Si on considère le terme q du programme (1.24) comme des pénalités pour toute action corrective y que l'on doit effectuer à la seconde étape, le recours fixe exige que ces pénalités ne dépendent pas de la réalisation de la variable aléatoire. Ce cas est assez courant d'ailleurs : dans le cas des problèmes de transport, si le nombre de camions (décision à la première étape) n'est pas suffisant pour couvrir la forte demande du lendemain, on fait appel à la location, dont le prix ne dépend pas de la demande de notre clientèle. Il existe toutefois des cas les pénalités dépendent des variables aléatoires, lorsque par exemple on est obligé, à cause d'une forte demande, d'acheter des quantités de gaz auprès du marché spot, le prix à payer dépendra de la demande.

Définition 1.2 (Recours complet) *Le recours fixe est complet si $\forall\, x \in \mathbb{R}^{n_1}$ il existe $y \in \mathcal{Y}$ tels que $Q(x,\xi) < +\infty\ \forall\, \xi \in \Xi$.*

Définition 1.3 (Recours relativement complet) *Le recours fixe est relativement complet si $\forall\, x \in \mathcal{X} \subseteq \mathbb{R}^{n_1}$ il existe $y \in \mathcal{Y}$ tels que $Q(x,\xi) < +\infty\ \forall\, \xi \in \Xi$.*

Notons que la différence entre les deux types de recours porte sur l'ensemble des x pour lesquels il existe une solution au second niveau. Si cet ensemble est le domaine de définition de x à la première étape le recours est relativement complet, tandis que si c'est $\mathbb{R}^{n_1}$ alors le recours est complet.

Selon cette définition, si pour chaque décision au premier niveau, on peut garantir, pour tout scénario pouvant se produire, l'existence des décisions au second niveau, et par conséquent on arrive toujours à satisfaire les contraintes du second niveau, alors le recours est complet. Dans le cas contraire, le système n'a pas moyen de remédier aux décisions, éventuellement destructives, de la première étape. Considérons à titre d'exemple le problème de transport stochastique. Si on décide à la première étape d'nstaller des entrepôts qui ne disposent pas d'une capacité suffisante pour couvrir la demande réelle (révélée à la seconde étape) qui peut se produire suivant un scénario (pessimiste), le problème résultant devient non-réalisable, puisque on n'arrivera jamais

à satisfaire la demande clientèle. Ceci est un exemple où le recours n'est pas complet. Si, au contraire, on a la possibilité de recourir à d'autres actions, par exemple locations des entrepôts, supplémentaires, alors on arrivera à satisfaire la demande à un coût certes plus élevé mais fini.

Dans les applications, le plus souvent, on choisit la matrice W telle que le recours soit complet et les matrices correspondantes portent le nom de *matrices de recours complet*. Une question intéressante est de savoir à l'avance si une matrice donnée est une matrice de recours complet. On peut démontrer (par exemple cf à [37]) qu'une matrice W de dimension $m \times n$ est une matrice de recours complet si et seulement si :

- son rang est égal à m ;
- si on considère que les colonnes linéairement indépendantes sont les m premières colonnes $W_1, W_2, ..., W_m$, alors les contraintes linéaires (1.26) admettent une solution réalisable.

$$\begin{aligned} Wy &= 0 \\ y_i &\geq 1, \quad i = 1, 2, ..., m \\ y &\geq 0 \end{aligned} \tag{1.26}$$

Si la matrice de notre problème n'est pas une matrice de recours complet, alors pour assurer la réalisabilité du problème par rapport aux variables de la première étape, il faut ajouter des contraintes dites *induites*. Les contraintes induites sont des contraintes supplémentaires en première étape qui doivent former un ensemble qui contient toutes les solutions x de la première étape, telles que pour toute réalisation de la variable aléatoire ξ, le programme en seconde étape soit réalisable, donc pour tout ξ^j, j=1, ..., r il existe y tel que $T(\xi^j)x + Wy^j = h(\xi^j)$, $y^j \geq 0$, j=1, ..., r. On peut montrer que si le support $\Xi = \{\xi^1, \xi^2, ..., \xi^r\}$ est soit fini, soit un polyèdre borné convexe, alors les contraintes induites forment toujours un ensemble polyédrique $\mathcal{I}$ convexe. Après avoir précisé quel est cet ensemble $\mathcal{I}$, on exige que les x soient compris dans $\mathcal{X} \bigcap \mathcal{I}$, donc on ne cherche que des solutions x qui satisfassent les contraintes induites, appartenant en même temps à leur espace de définition $\mathcal{X}$. Plus de détails seront donnés lors de la présentation de la méthode *"L-Shaped"* au paragraphe 2.2 et plus précisément lors de

la description des coupes de réalisabilité qui jouent le même rôle que les contraintes induites.

Définition 1.4 (Recours simple) *Le recours fixe est simple si sa forme canonique est celle-ci :*

$$\begin{pmatrix} q \\ W \end{pmatrix} = \begin{pmatrix} q^+ & q^- \\ I & -I \end{pmatrix}$$

où q^+, q^- sont les $1 \times n_2$ vecteurs de pénalités et I est la $n_2 \times n_2$ matrice identité.

Pour mieux comprendre le rôle que le recours simple joue, considérons le recours suivant :

$$y_1 - y_2 = h_1(\xi) - z_1 \quad \text{avec} \quad z_1 = T_1 x \quad (T_1 \text{ étant la 1-ère ligne de la matrice } T)$$

Nous voyons que le second membre est connu à la seconde étape, puisque la variable ξ qui influence $h_1(\xi)$ s'est réalisée et les décisions x sont déjà prises à la première étape. Si $h_1(\xi) - z_1 > 0$ alors $y_1 > 0$ et $y_2 = 0$ (cf à la fonction objectif dans (1.24)) alors que si $h_1(\xi) - z_1 < 0$ alors $y_1 = 0$ et $y_2 > 0$. Donc, selon le sens de la violation il y a des pénalités (ou des primes) différentes, ce qui rend le recours "simple". En plus, observons que le recours simple est toujours complet, puisque on a toujours moyen de satisfaire les contraintes avec les variables y^+ et y^-.

En général, pour s'assurer que le coût de la seconde étape reste toujours raisonnable, i.e $|Q(x, \xi)| < +\infty$, il faut que :

1. le recours soit complet (W t.q $Q(x, \xi) < +\infty$) afin de pouvoir satisfaire les contraintes de la seconde étape à un coût fini (avant ou après la construction de l'ensemble des contraintes induites) ;
2. le recours soit suffisamment cher ($q^+ + q^- > 0$) pour assurer un gain limité. Si $q^+ + q^- < 0$, alors on aurait intérêt à violer la contrainte correspondant à ces pénalités pour gagner plus. De tels cas doivent être évités lors de la modélisation, car ils amènent à des solutions soit non-réalisable, soit non bornées.

Les modèles qui s'appuient sur le recours simple sont plus faciles à résoudre que les modèles généraux de recours, et des algorithmes spécifiques à leur structure ont été développés, qui donnent des solutions plus rapidement. Klen Haneveld et van der

Vlerk travaillent spécialement sur les problèmes de recours simple, en proposant des modélisations qui donnent des temps de calcul pour la résolution de tels problèmes (de recours en deux étapes avec des variables aléatoires discrètes) presque aussi petits que la version déterministe du problème (remplacer tous les termes aléatoires par leur moyenne).

1.4.2 Propriétés des programmes linéaires avec recours

Deux propriétés de la fonction de recours vont être présentées dans ce paragraphe : la convexité et la linéarité par morceaux. Ces deux propriétés nous seront utiles pour la description des méthodes de résolution.

On s'intéressera d'abord à la convexité de la fonction de recours. Considérons le problème de départ (1.24). L'espace des solutions du problème de recours est construit à travers des équations linéaires, il est donc convexe. De plus, si la fonction $\mathbb{E}\{Q(x,\xi)\}$ est convexe en x (on parle de minimisation) on a finalement un problème convexe. En faisant l'hypothèse que la fonction $Q(x,\xi)$ est convexe pour tout $\xi \in \Xi$, alors par la linéarité de l'espérance, il est évident que la fonction à minimiser est convexe :

Pour $\tilde{x} = \lambda x_1 + (1-\lambda)x_2 \quad \forall\, \xi \in \Xi$:

$$\begin{array}{rcll} Q(\tilde{x},\xi) & \leq & \lambda Q(x_1,\xi) + (1-\lambda)Q(x_2,\xi) & \Leftrightarrow \\ \mathbb{E}\{Q(\tilde{x},\xi)\} & \leq & \lambda\mathbb{E}\{Q(x_1,\xi)\} + (1-\lambda)\mathbb{E}\{Q(x_2,\xi)\} & \end{array} \tag{1.27}$$

et donc $c^T + \mathbb{E}\{Q(x,\xi)\}$ est une fonction à minimiser convexe. Dans le cas de fonctions non-linéaires, il est plus difficile d'assurer la convexité des problèmes.

Regardons plus précisément la fonction $Q(x,\xi)$ pour un $\bar{\xi}$ donné. Cette fonction est exprimée (cf à (1.24)) comme :

$$\begin{array}{ll} & Q(x) = \min q^T y \\ \text{s.c} & Wy = h(\bar{\xi}) - Tx \\ & y \geq 0 \end{array} \tag{1.28}$$

On va maintenant montrer la linéarité par morceaux de la fonction de recours. On se place dans le cas d'un recours fixe (W et q sont déterministes ; T pas nécessairement). On construit le dual :

$$\begin{array}{ll} & Q(x) = \max y^T(h(\bar{\xi}) - Tx) \\ \text{s.c} & W^T y \leq q \\ & u \geq 0 \end{array} \tag{1.29}$$

Le polyhèdre y : $W^T y \leq q$ est fermé et convexe ; par conséquent il a un nombre fini de points extrêmes. La fonction $Q(x)$ peut être reécrite comme suit :

$$Q(x) = \max_{y_i, i=1,\ldots,N} y_i^T(h(\bar{\xi}) - Tx) \tag{1.30}$$

et il est bien connu de la théorie de la programmation linéaire que cette fonction, représentée comme le maximum point à point d'un ensemble de fonctions affines, est une fonction linéaire par morceaux.

1.5 Méthodes hybrides

La modélisation d'un problème en utilisant l'approche de recours ou celle des contraintes probabilistes est toujours soumise aux critiques. Pour les contraintes probabilistes, on pourrait mettre en question la probabilité de satisfaction d'une contrainte quelconque, puisque une telle information est souvent indisponible. D'autre part, en utilisant un modèle de recours, on suppose être capable de mesurer la violation de n'importe quelle contrainte, et le plus important, d'imposer une pénalité pour cette violation. Or, comment pénaliserait-on, par exemple, la faillite d'une entreprise ? Bien entendu, on peut toujours, suivant le cas, préférer une modélisation à une autre, puisque il existe bien des applications, où les coûts des actions de recours sont connus ou bien la probabilité de la violation d'une contrainte peut être aisément déterminée.

On pourrait pourtant, pour certaines applications où ceci semble utile, combiner les approches précédemment exposées, en créant ainsi le modèle hybride suivant :

$$
\begin{aligned}
&\min\ h(x) + \mathbb{E}\left\{q(\xi - Tx)\right\} \\
&\textit{sous contraintes :} \\
&\mathbb{P}\{Tx \geq \xi\} \geq p \\
&\mathbb{E}\left\{\xi_i - T_i x | \xi_i - T_i x > 0\right\}\mathbb{P}\{\xi_i - T_i x > 0\} \leq d_i, \quad i{=}1, 2, \ldots, r \\
&Ax = b \\
&x \geq 0
\end{aligned}
\tag{1.31}
$$

Dans le programme (1.31), on exige :

1. de garantir que la violation de la contrainte $Tx \geq \xi$ ait lieu au plus dans $(1 - p)100\%$ des cas ;
2. de garder la moyenne de la violation de chaque contrainte violée en dessous de d_i ;
3. de respecter toutes les contraintes déterministes, imposées sur les décisions d'aujourd'hui, et enfin
4. parmi les solutions réalisables, de choisir celle qui minimise le coût, ainsi que l'espérance des pénalités.

1.6 Optimisation robuste

Dans la littérature les auteurs font souvent la distinction entre la programmation stochastique et la programmation robuste (par exemple cf à [51]). Nous décidons d'inclure les deux approches dans la même introduction, puisque les deux tiennent compte de l'incertitude en utilisant des outils de l'analyse et de la programmation convexe.

Le but de l'optimisation robuste est en général de trouver des solutions qui ne soient pas sensibles aux variations du système, elle sont donc robustes. La problèmatique de la robustesse emerge de la nécessité d'avoir une solution dont le coût ne sera jamais plus grand qu'un seuil donné quelque soit la réalisation des aléas. Ce qui est intéressant c'est que le coût de la robustesse n'est pas forcément très grand, ce qui signifie que pour le même coût (à un ε près) on peut passer d'une solution quelconque optimale à une solution robuste. Dans l'optimisation robuste on adopte la vision du pire des cas

puisque on minimise le coût du pire cas dans l'espace d'incertitude. Il existe les liens qui ont été mis en évidence entre cette approche et les modèles de Markowitz.

Dans le domaine de l'optimisation robuste on trouve différentes directions, comme par exemple les travaux de Mulvey (voir aussi Vanderbei et Zenios [51], Vladimirou et Zenios [75], Beraldi et coauteurs [8]) d'une part et de Ben-Tal et Nemirovski [5] d'autre part. Dans les premiers travaux, l'incertitude est décrite à travers un ensemble de scénarios, le but étant la minimisation d'une combinaison de l'espérance et des moments, alors que dans les seconds, l'aléa est considéré de façon moins précise : les paramètres incertains varient dans un espace d'incertitude $\mathcal{U}$ (notation employée souvent par Ben-Tal, Nemirovski et autres auteurs) qui peut constituer un hypercube ou un éllipsoïde. Si u_i sont les éléments de l'espace $\mathcal{U}$, une description possible de l'espace $\mathcal{U}$ pourrait être la suivante :

$$\mathcal{U} = \{u_1 + u_2 = 3,\ 0 \leq u_1 \leq 1\} \tag{1.32}$$

Dans le cas où la robustesse est considérée dans un espace de scénarios, la formulation du programme est similaire à celle d'un modèle de recours (cf au poaragraphe 1.4). Ce type d'approches est connu dans la littérature comme approche du pire scénario avec des pénalités. Il y a deux types de variables, comme les variables du premier et du second niveau ; maintenant ces variables s'appellent respectivement *design variables* et *control variables*. Le programme résultant est un modèle quadratique.

Dans les approches de la robustesse dans le sens de Ben-Tal et autres auteurs, toutes les contraintes sont considérées comme étant des contraintes *dures* (*"hard constraints"*). Cette approche est plus proche du contrôle optimal que de la programmation mathématique comme il est noté dans le travail de Ben-Tal et Nemirovski [6]. L'approche est orientée vers le pire scénario. Dans ce cadre, une solution optimale est celle issue du programme suivant :

$$\begin{array}{l} \min c^T x \\ \text{sous contraintes} \\ Ax \geq b \quad \forall\,(A,b) \in \mathcal{U} \end{array} \tag{1.33}$$

qui est appelé le programme robuste équivalent (*"robust counterpart"*). La matrice A et le vecteur b sont considérés comme incertains.

Dans certains travaux l'incertitude est colonne-à-colonne, dans ce cas le programme résultant est un programme linéaire mais très pessimiste. L'incertitude colonne-à-colonne signifie que les composantes de la variable aléatoire a_i qui se trouvent dans la matrice technologique A et qui correspondent à la colonne i appartiennent à un espace convexe, comme par exemple celui de (1.32). Il se peut que l'incertitude est ligne-à-ligne. Dans ce cas là, si l'espace d'incertitude est considéré comme étant un éllipsoïde, le problème appartient à la programmation conique quadratique.

L'espace d'incertitude, qu'il soit caractérisé comme un hyper-cube, un éllipsoïde ou une intersection d'éllipsoïde nous ramène toujours à un problème de programmation semi-infinie (c'est-à-dire que l'on dispose d'un nombre infini de variables ou de contraintes) qui est ensuite transformée en un problème de programmation conique. En général, le programme résultant n'est pas un programme linéaire.

1.7 Distribution de l'optimum

Ce qui nous a intéressé jusqu'à maintenant dans les problèmes, ce sont les décisions que l'on prend *aujourd'hui* qui doivent être optimales (dans un sens prédéterminé) et doivent permettre de faire face aux situations inconnues qui se produiront *demain*. Ce type de problème s'appellent *"here and now problems"*.

Il y a également une autre catégorie de problèmes. Imaginons que l'on soit capable de connaître la réalisation de la variable aléatoire pour *demain*. Alors on est également capable de trouver la solution optimale *d'aujourd'hui* qui correspond à ce problème. Ces problèmes s'appellent également *"wait and see problems"*.

Si j est le compteur des différentes réalisations des paramètres incertains, alors on a une solution optimale x_j^* qui correspond à la réalisation j. La question qui se pose dans un problème de distribution de l'optimum porte sur la distribution probabiliste de deux choses : (i) de la valeur de la solution optimale et (ii) de la solution optimale elle même.

Un autre problème qui pourrait également faire partie des problèmes de distribution de l'optimum est la stabilité de la base, c'est-à-dire que la probabilité pour qu'une base optimale reste insensible aux variations des paramètres aléatoires. Une base ro-

buste aux aléas signifie pratiquement que les mêmes ressources seront utilisées même si le futur est incertain. En revanche la participation de chaque ressource dépendra de la réalisation du paramètre incertain.

Les travaux les plus importants dans le domaine sont dus à Bereanu (surtout [11] et [10], mais aussi [9], [12]). D'autres travaux intéressants sont ceux de Dempster [28], de Tamer [71] de King [38] et pour la programmation en nombre entiers le travail de Rinnooy Kan [56].

1.8 Valeur espérée d'information parfaite (*Expected Value of Perfect Information – EVPI*)

Souvent dans les applications réelles on souhaite quantifier la *stochasticité* d'un problème particulier. Pour expliquer cela en mots, on dirait que l'on souhaite avoir une mesure de "combien notre problème est loin d'être déterministe".

En même temps, une autre question qui pourrait nous intéresser est la suivante : considérons un problème en deux étapes où on prend ajourd'hui des décisions qui doivent être valables pour toutes les réalisations de la variable aléatoire demain. On n'a aucun moyen de connaître la réalisation de demain. Supposons que l'on puisse payer pour obtneir une telle information. Combien serait-on disposé à céder de notre gain pour l'obtenir ?

La réponse aux deux questions précédemment posées est la valeur espérée d'information parfaite (EVPI). Si HN constitue la valeur de la fonction objectif après la résolution du problème *"here and now"* et WS la valeur optimale du problème *"wait and see"* (c'est-à-dire la moyenne des coûts issus de la résolution séparée de chaque scénario), alors la valeur espérée d'information parfaite pour un problème de minimisation sera la suivante :

$$\text{EVPI=HN-WS} \tag{1.34}$$

En d'autres termes le problème *"here and now"* est le problème avec les variables couplantes ou avec les contraintes de non-anticipativité. Le problème *"wait and see"* est le problème avec les contraintes de non-anticipativité relaxées. La différence de ces

deux problèmes constitue la valeur maximale que le décideur, confronté à un problème stochastique en deux étapes, serait disposé à payer pour obtenir l'information portant sur la réalisation qui se produira demain.

Notons que, généralement, on ne peut pas associer une mesure du type EVPI à chaque composante du vecteur aléatoire séparément. À titre d'exemple, pour deux composantes aléatoires indépendantes ξ_1 et ξ_2, on n'est pas capable d'estimer la valeur maximale à payer pour obtenir uniquement l'information sur la réalisation de ξ_1. Si on se propose comme problème stochastique de parier sur la somme des réalisations de deux dés que l'on lance, on parierait sur l'évènement de la plus forte probabilité : le nombre 7. Notons pourtant que l'on ne payerait rien du tout pour obtenir une information sur la réalisation du premier dé, puisque de toute façon, peu importe la réalisation du premier dé, le nombre 7 restera toujours notre choix.

1.9 Conclusions

Dans un monde déterministe, toute donnée du problème est connue avec certitude. Trouver l'optmimum d'un problème d'optimisation déterministe fait appel à un ensemble de méthodes de programmation mathématique bien étudiées avec de nombreuses applications. Lorsque des paramètres incertains apparaissent dans un problème d'optimisation, on ne se retrouve plus dans le domaine déterministe, on passe dans un monde stochastique.

La modélisation d'un problème stochastique n'est pas unique ni générique. La façon dont les données sont disponibles, la quantité et la qualité de l'information qui nous est donnée, ainsi que l'objectif que le décideur se pose, déterminent la représentation mathématique du problème. On a introduit dans le présent chapitre différentes manières de modéliser des problèmes stochastiques en observant le contexte dans lequel chaque modélisation peut être légitime.

Parmi les modèles les plus souvent rencontrés dans la littérature, on retrouve les modèles avec des contraintes probabilistes. Une ou plusieurs contraintes peuvent être violées avec une certaine probabilité qui représente d'une certaine manière la fiabilité du système à modéliser. Les contraintes probabilistes s'appliquent souvent dans des

problèmes d'ingénierié et de contrôle.

Les modèles robustes sont également une autre grande famille de méthodes que les auteurs distinguent souvent de la programmation stochastique. Il existe des travaux qui essaient de faire le lien entre les modèles de recours, la voie de modélisation la plus employée dans la programmation stochastique, et les modèles robustes. Un travail représentatif est attribué à Vladimirou et Zenios [75].

Les modèles de recours ont également été introduits dans le présent chapitre. Le principe de recours repose sur le principe des décisions irrévocables que l'on prend aujourd'hui et que l'on ne peut plus mettre en question ultérieurement. Les variables de recours jouent le rôle du levier nous permettant d'effectuer des actions correctives, une fois que l'aléa se sera produit. Dans ce travail on s'intéressera aux modèles de recours. Les modèles de recours se partagent en deux catégories : les modèles en deux étapes et les modèles multi-étapes. Dans le chapitre 2 on se focalisera sur les méthodes de résolution des modèles en deux étapes en présentant les méthodes principales de décomposition. La présentation des méthodes de résolution pour les problèmes multi-étapes est réportée au chapitre 3.

Chapitre 2

Modèles en deux étapes

2.1 Introduction

Après une étude introductoire à l'incertitude et la programmation stochastique, on choisit de se focaliser plus sur les modèles de recours en deux étapes. Dans ce chapitre on s'efforcera d'esquisser les méthodes les plus importantes et les plus couramment utilisées dans la littérrature pour la résolution des problèmes pouvant se modéliser en deux étapes.

Dans la suite on supposera que l'on dispose d'un ensemble de réalisations que l'on appelera également des *scénarios*. On considère que l'on a aucune information sur la manière dont ces scénarios ont été générés. Ils auraient pu être crées par l'opinion des experts ou suivant la fonction de répartition d'une variable aléatoire quelconque. Le problème de référence sera le programme linéaire suivant :

$$\begin{array}{ll} & \min c^T x + \mathcal{Q}(x) \\ \text{s.c.} & Ax = b \\ \text{avec} & \mathcal{Q}(x) = \sum_s p_s Q(x,s) \\ \text{s.c.} & W(s)y(s) = h(s) - T(s)x \end{array} \tag{2.1}$$

La fonction $Q(x,s)$ représente la valeur de la fonction issue du problème d'optimisation correspondant au scénario s étant donné une solution x, p_s est la probabilité d'ocurrence

du scénario s et $\mathcal{Q}(x)$ est l'espérance de cette fonction sur l'espace de scénarios qui s'appelle également *fonction de recours*.

Le problème (2.1) peut être écrit comme un grand problème linéaire (*deterministic equivalent*) ; or ses dimensions augmentent vite en fonction du nombre de scénarios. Lorsque le nombre de scénarios commence à être assez grand pour que la mémoire de la machine s'avère insuffisante pour contenir le modèle, on est obligé de faire appel à d'autres méthodes de résolution.

Dans ce qui suit on va décrire brièvement certaines méthodes de résolution, faisant partie des méthodes de décomposition les plus fréquemment utilisées, qui sont amplement employées dans les applications de modèles en deux étapes. Une courte discussion sur le cas ou certaines variables sont entières est également incluse dans le paragraphe 2.5. Le lecteur peut se référer à tout ouvrage de programmation stochastique ([37], [64], [18]) pour plus de détail sur les méthodes qui seront exposées, ainsi qu'à d'autres, plus spécifiques aux applications particulières, qui ont été éventuellement omises.

2.2 Méthode "L-Shaped"

En 1962 Benders [7] a proposé une méthode de décomposition consistant à partitionner les variables d'un programme linéaire en deux catégories. Son objectif a été de résoudre des problèmes de programmation mixte.

En s'ispirant de cette idée Wets [77] en 1966 a appliqué la décomposition de Benders sur les problèmes de recours en deux étapes. En 1969 Van Slyke et Wets [73] ont dévéloppé la première méthode de la programmation stochastique sous le nom *"L-Shaped"* destinée à résoudre les problèmes représentés en tant que modèles de recours en deux étapes.

L'idée de la méthode "L-Shaped" consiste donc à partager les variables en deux niveaux. Les variables du premier niveau sont les variables dont la valeur doit être fixée avant la réalisation des aléas et ne peut en aucun cas être modifiée suivant la réalisation de la variable aléatoire. De l'autre part, les variables du second niveau (les variables de recours) sont autorisées à dépendre des variables aléatoires et des variables du premier

niveau. La logique derrière cette règle est que l'on tient compte de toute information déjà disponible afin de prendre nos décision, mais en même temps, il est irréaliste de tenir compte des évènements inconnus, qui n'existent, au moment de la décision, que dans un espace probabiliste.

On crée deux problèmes complémentaires, le problème *maître* (*master*) et les problèmes *satellites* ou *sous-problèmes* ou *problèmes-esclaves*. Les problèmes satellites (un pour chaque scénario) réçoivent comme entrée les valeurs des variables du premier niveau et calculent les variables de recours propres à chaque scénario ainsi que le côut associé. Le problème maître comporte les variables du premier niveau et une description approximée de la fonction de recours dont la forme exacte n'est pas connue. Si une telle fonction était donnée de façon explicite, alors on connaitrait automatiquement l'impact de chaque solution du premier niveau au second. Le but est de construire cette fonction de recours à partir des solutions proposées au fur et à mesure par le problème maître et en ajoutant des coupes déduites de la résolution des problèmes-satellites. Les coupes se formeront en utilisant la valeur de la fonction objectif du problème-satellite avec un de ses sous-gradients (on parle des sous-gradients, puisque la focntion de recours est une fonction linéaire par morceaux, cf au paragraphe 1.4.2).

L'algorithme ne démarre qu'avec les contraintes du premier niveau :

$$\begin{aligned} & \min c^T x + \theta \\ \text{s.c.} \quad & Ax = b \\ & \theta \geq -M \\ & x \geq 0 \end{aligned} \tag{2.2}$$

θ est la variable qui donne l'approximation de la fonction réelle de recours. Si on n'y rajoute pas la contrainte supplémentaire $\theta \geq -M$ à la première itération de l'algorithme, il est évident que le problème sera non-bornée et une solution x ne sera pas obtenue. À chaque itération on injecte la variable x à chaque sous-problème du second niveau. On demande de chaque sous-problème deux informations : (i) la valeur de sa fonction objectif et (ii) le vecteur des variables duales optimales. Ces informations se regroupent en une coupe, constituant une combinaison linéaire de tous les problèmes-satellites.

Au moment où la fonction de recours $\mathcal{Q}(x)$ sera *"très proche"* de son approxima-

tion θ, la méthode s'arrête avec la précision souhaitée. Tant que ceci n'est pas le cas, l'algorithme procède en ajoutant des coupes dont l'existence et la validité est assurée par la théorie de la dualité (trouver une direction d'amélioration).

Au départ on résout le problème (2.2). On récupère la solution (x^*, θ) et on fait passer cette solution à chaque sous-problème :

$$\begin{array}{lll} & \textbf{\textit{Primal}} & \textbf{\textit{Dual}} \\ & \min q(\xi)^T y & \max \lambda^T(h(\xi) - Tx) \\ \text{s.c} & Wy \geq h(\xi) - Tx & \text{s.c} \quad \lambda W \leq q(\xi) \\ & y \in \mathcal{Y} & \lambda \geq 0 \end{array} \tag{2.3}$$

Si lors d'une itération et pour un x^* donné le problème dual n'est pas borné, ce qui signifie que le primal est non-réalisable, on rajoute dans le problème maître la coupe de réalisabilité suivante :

$$\hat{\lambda}^T T x \geq \hat{\lambda}^T h \tag{2.4}$$

Cette contrainte est satisfaite également pour la solution x^* qui rendait le problème non-réalisable. De cette manière, on arrive à imposer implicitement le recours relativement complet (cf à la définition 1.3 au paragraphe 1.4.1). À noter que $\hat{\lambda}$ est une direction extrème du cone polaire correspondant à la matrice W et elle est calculée pour un point donné x^* à partir du problème suivant :

$$\begin{array}{ll} & \max \lambda^T(h(\xi) - Tx^*) \\ \text{s.c.} & \lambda^T W \leq 0 \\ & \|\lambda\| \leq 1 \end{array} \tag{2.5}$$

La norme sur λ pourrait être soit la norme $\|\cdot\|_2$ soit la norme $\|\cdot\|_1$ si on souhaite rester en linéaire.

Si tous les problèmes admettent une solution, on essayera de *faire mieux* en obtenant une direction d'amélioration de la fonction réelle de recours (ceci pour chaque scénario séparément) et tracer ainsi le plan sécant qui approxime le mieux la fonction de recours au point courant x^*. Pour tracer cet *hyperplan* on a besoin (i) de la valeur

de la fonction objectif $f_i(x^*)$ pour chaque sous-problème i auquel on a passé la solution x^*, ainsi que (ii) le vecteur dual optimal λ_i^* du sous-problème i.

Si on se restreint au sous-problème i (correspondant au scénario i), l'hyperplan d'appui au point x^* est donné par la formule suivante :

$$Q_i(x) = f_i(x^*) - \lambda_i^{*T} T_i(x - x^*) \tag{2.6}$$

On note bien que par dualité (2.6) est équivalent à (2.7) :

$$Q_i(x) = \lambda_i^{*T}(h_i - T_i x) \tag{2.7}$$

La fonction de recours est la combinaison linéaire de la fonction de chaque sous-problème différent. On a donc :

$$Q(x^*) = \sum_i p_i Q_i(x^*) \tag{2.8}$$

où p_i est la probabilité d'ocurrence du scénario i qui pondère la partie $Q_i(x^*)$ correspondant au scénario i.

La fonction $Q(x)$ étant convexe, le support se trouve géométriquement en dessous du graphe de la fonction. La coupe à rajouter dans le maître, qui regroupe toute information de tout sous-problème est la suivante :

$$\theta \geq \sum_i p_i(f_i - \lambda_i^{*T} T_i(x - x^*)) \quad \Leftrightarrow \tag{2.9}$$

$$\theta \geq \sum_i p_i \lambda_i^{*T}(h_i - T_i x) \tag{2.10}$$

Les coupes du type (2.4) sont appelées *coupes de réalisabilité* et celles du type (2.9) (ou (2.10)) *coupes d'optimalité*. À chaque itération on rajoute au problème maître soit une coupe de réalisabilité, soit une coupe d'optimalité. Au cours des itérations, θ sera le maximum point à point d'une famille d'inégalités qui approximeront la fonction de recours. Si on arrive à trouver un θ^* dont la valeur vaudra $Q(x^*)$, alors on aura trouvé l'optimum, comme θ représente une minorante de la fonction réelle de recours

$Q(x)$. Pratiquement on s'arrête lorsque la différence relative entre θ et $Q(x)$ est plus petite que ε.

La figure 1 montre le flux d'information entre le problème maître et les sous-problèmes (correspondant aux différents scénarios). Seulement les coupes d'optimalité ont été considérées, mais il en va de même pour les coupes de réalisabilité.

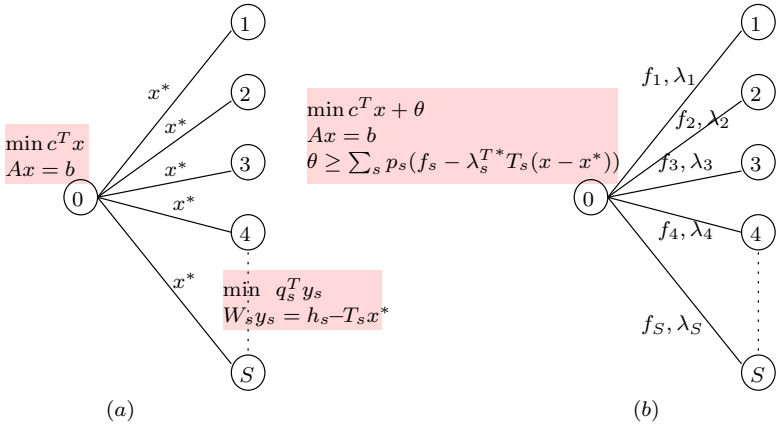


Fig. 1: Flux d'information du maître aux sous-problèmes pendant la première itération : (a) les solutions x^* sont transférées du maître à tout sous-problème, (b) la valeur de la fonction f_s et le vecteur dual optimal λ_s sont envoyés de tout sous-problème au maître ; à partir ces informations une nouvelle coupe agrégée se rajoute au maître.

La figure 2 montre géométriquement les étapes de l'algorithme "L-Shaped". On a commencé d'abord par les coupes de réalisabilité qui sont dessinées à gauche et à droite de la fonction. Tout en bas se trouve la contrainte $\theta \geq -M$ où $-M$ est une borne inférieure connue de la fonction de recours (ou un très grand nombre négatif). À chaque itération de l'algorithme, après avoir résolu plusieurs programmes linéaires, un pour chaque satellite, on résout le problème-maître en lui rajoutant une coupe d'optimalité de plus (dans la figure on considère que toute coupe de réalisabilité a été construite). À la fin de la méthode, la famille de coupes rajoutées approxime aussi bien que souhaité (par le critère d'arrêt) la fonction de recours.

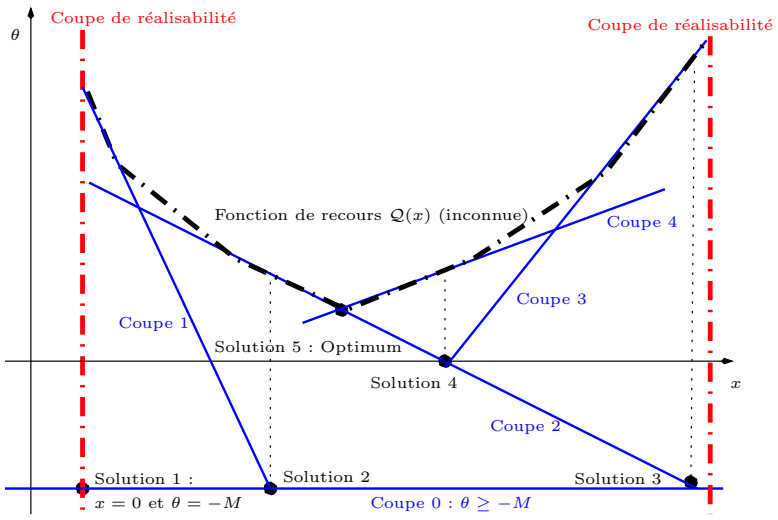


Fig. 2: Déroulement de la méthode "L-Shaped". À chaque instant une coupe d'optimalité est rajoutée. Au début, il n'y a que les coupes de réalisabilité.

2.2.1 "L-Shaped" en version multi-coupes

En 1988 Birge et Louveaux [19] ont proposé une variante de la méthode "L-Shaped" en remplaçant la seule coupe d'optimalité qui regroupe toutes les informations de chaque scénario par plusieurs coupes (une par scénario), introduisant ainsi la version *multi-coupes* de la méthode "L-Shaped". On n'a plus un seul terme θ qui décrit la fonction de recours, mais plusieurs θ_i, un pour chaque scénario i. Les coupes que l'on rajoute au maître à chaque itération sont donc les suivantes :

$$\theta_i \geq \lambda_i^{*T}(h_i - T_i x^*), \quad \forall i \tag{2.11}$$

La fonction objectif du problème maître ne comporte plus un seul terme θ mais une combinaison linéaire de θ_i. À la k-ième itération, le problème maître ressemble à

(2.12)

$$\begin{array}{ll} & \min c^T + \sum_s p_s \theta_s \\ \text{s.c.} & Ax = b \\ & \theta_s \geq \lambda_s^{*T}(h_s - T_s x^{i*}), \quad \forall s,\, i{\leq}k \\ & x \geq 0 \end{array} \tag{2.12}$$

La mémoire requise est bien entendu plus importante, mais la méthode exploite bien l'information déjà recueillie des précédentes itérations et donc globalement le temps de calcul est en pratique réduit. Ruszczyński [61] cite pourtant qu'il avait auparavant observé qu'une agrégation éventuelle des coupes pourrait ralentir jusqu'à 5 fois la convergence. Il suffit de considérer un problème avec des millions de scénarios (réalisations de la variable aléatoire à la seconde étape) et d'imaginer qu'à chaque itération *tous les scénarios* renverront leurs coupes, une par une, au problème-maître. Il y aura un moment où le temps de calcul pour la résolution du maître sera très lent.

La méthode "L-Shaped" converge que ce soit en version multi-coupes ou en version coupes agrégées. La convergence de la méthode a été prouvée par Van Slyke et Wets ([73]), mais ceci seulement pour les coupes basiques. Les coupes basiques sont les coupes dont les coefficients portant sur les multiplicateurs de Lagrange sont choisis parmi un ensemble fini qui ne dépend pas de la solution courante x^k (à l'itération k) : l'ensemble de solutions de base réalisables du problème dual (2.5). Pour les coupes non-basiques la convergence a été démontrée seulement en 2003 par Ruszczyński [64].

2.3 Décomposition régularisée

La méthode "L-Shaped" présente deux inconvénients importants :

1. Les coupes à chaque itération que l'on ajoute s'aditionnent aux coupes existantes et par conséquent le nombre de coupes dans le problème maître peut augmenter sans pouvoir maitriser sa taille. Surtout en version multi-coupes, quand le nombre de scénarios devient important, les dimensions du problème-maître en termes de contraintes augmentent très vite.
2. La méthode "L-Shaped" ne peut pas exploiter sufisammment bien une bonne solution. Au cours de l'algorithme, même si à l'itération k une bonne solution x_k

est repérée, elle peut être remplacée par une autre, x_{k+1}, dont le coût approximé peut être meilleur, sans que son coût réel le soit. La figure 3 montre un exemple.

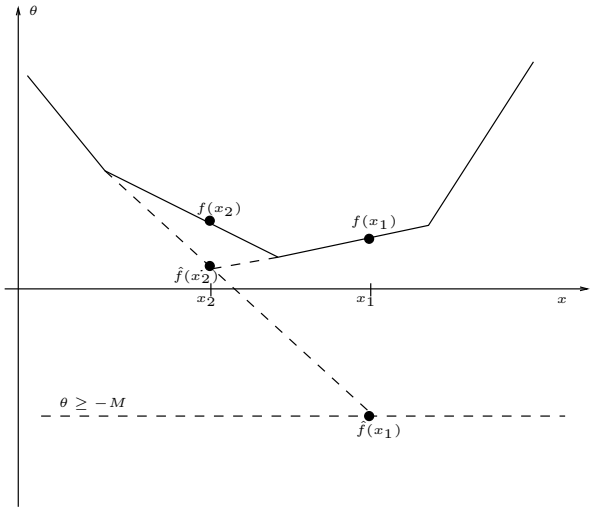


Fig. 3: La méthode "L-Shaped" n'exploite pas bien une bonne solution. La x_1 bien que meilleure en termes de coût réel ($f(x_1) < f(x_2)$) est remplacée par x_2 qui a un *meilleur* coût approximé ($\hat{f}(x_2) > \hat{f}(x_1)$).

Pour remédier à ces problèmes, Ruszczyński [59] a proposé la décomposition *régularisée*. En introduisant un terme quadratique dans la fonction objectif du problème maître, on arrive à contrôler la distance euclidienne entre la nouvelle solution et une *solution de référence* en stabilisant ainsi les itérations de la méthode. L'algorithme peut démarrer en utilisant une bonne solution de départ comme solution de référence. S'il n'y pas moyen d'avoir une bonne solution de départ, l'algorithme peut commencer avec la première solution proposée par "L-Shaped". La question qui se pose maintenant est comment mettre à jour cette solution de référence.

À l'itération k, on résout le programme suivant :

$$\begin{array}{ll} & \min \frac{1}{2\sigma}||x - z_{k-1}||^2 + c^T x + \sum_s p_s \theta_s \\ \text{s.c} & \{x \in X\} \cap \{\theta \in \Theta_k\} \end{array} \tag{2.13}$$

où z_{k-1} est la solution de référence qui a été choisie à l'issue de l'itération précédente. Les ensembles X et Θ_k donnent de façon succinte le domaine de définition du vecteur

(x, θ). On rappelle que θ est donné en tant que le maximum point à point d'une famille d'inégalités qui approximent la fonction de recours. L'indice k porte sur le fait qu'à chaque itération les inégalités décrivant la fonction de recours augmentent en nombre.

Le programme (2.13) nous fournit à chaque itération k une solution x_k, étant donné une solution de référence z_{k-1}. Cette solution x_k est ensuite passée à chaque scénario séparément pour calculer le coût réel $f_i(x_k)$ pour ce scénario i, comme dans la méthode "L-Shaped". Lorsque tous les scénarios seront traités, afin de choisir la nouvelle solution de référence z_k pour la prochaine itération, on compare les trois quantités suivantes :

- $F(x_k) = c^T x_k + \sum_{i=1}^{S} p_i f_i(x_k)$: Le coût *réel* de la fonction de recours, calculé pour la solution x_k.
- $\hat{F}_k(x_k) = c^T x_k + \sum_{i=1}^{S} p_i \theta_i$: Le coût *approximé* de la fonction de recours, calculé pour la solution x_k.
- $F(z_{k-1}) = c^T z_{k-1} + \sum_{i=1}^{S} p_i f_i(z_{k-1})$: Le coût *réel* de la fonction de recours, calculé pour la solution de référence z_{k-1}.

Le terme $f_i(x_k)$ représente le coût *réel* du scénario i pour la solution x_k, alors que θ_i représente le coût *approximé* du même scénario. La fonction d'approximation $\hat{F}_k(x_k)$ de la solution x_k est indicée sur les itérations puisque elle change d'une itération à l'autre. La figure 4 fait apparaître ces trois quantités.

À chaque itération, on décide si la solution de référence reste invariée ou si elle est remplacée par la x_k. La nouvelle solution x_k passe alors le test suivant :

- si l'amélioration effective $(F(z_{k-1}) - F(x_k))$ est relativement importante en comparaison avec l'amélioration estimée $(F(z_{k-1}) - \hat{F}_(x_k))$ et
- si le point x_k se trouve à un point anguleux de $\hat{F}$,

alors la solution remplace la solution de référence. Sinon, la solution de référence reste inchangée pour l'itération suivante. L'introduction de la solution de référence stabilise d'une certaine manière le problème maître puisque elle ne permet pas de sauts inutiles.

Quant aux coupes de réalisabilité et d'optimalité, il n'y a pas de différence par rapport à la méthode "L-Shaped".

Les étapes de l'algorithme dans sa version multi-coupes qui est la plus performante (cf à [64], paragraphe 3.1) se résument ci-dessous.

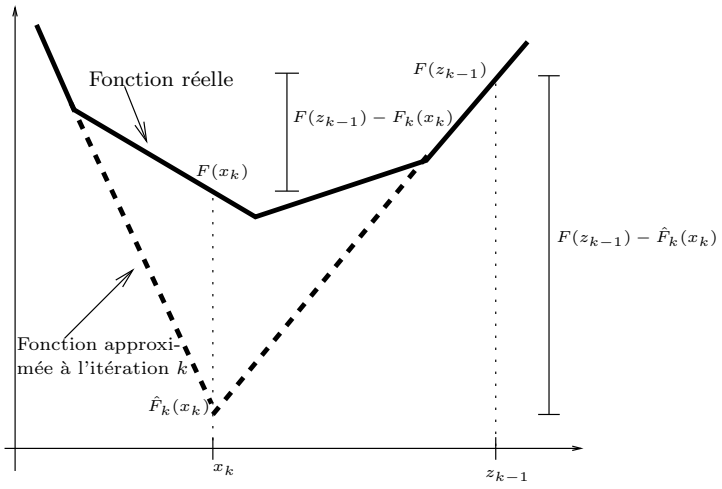


Fig. 4: Les trois quantités importantes pour se décider sur la mise à jour de la solution de référence z_k à l'itération k dans le cadre de la méthode de décomposition régularisée.

ALGORITHME DE LA DÉCOMPOSITION RÉGULARISÉE

Étape 0. Poser k=0. On considère avoir une solution x_0.

Étape 1. Pour tout scénario s=1, ..., S résoudre le sous-problème (primal) (2.3) pour x=x_k.

– Si le coût du scénario s est infini, alors construire une coupe de réalisabilité au problème-maître, comme dans la méthode "L-Shaped".

– Si le coût du scénario s est fini et il est différent (supérieur) de son approximation alors construire une coupe d'optimalité au problème-maître, comme dans la méthode "L-Shaped".

Étape 2. Si k=1 ou si

$$F(x_k) \leq (1-\gamma)F(z_{k-1}) + \gamma F_k(x_k)$$

alors poser z_k=x_k ; sinon, poser z_k=z_{k-1}.

Étape 3. Résoudre le programme du maître (2.13) tel qu'il est à l'itération k (c'est-à-dire avec les coupes d'optimalité et de réalisabilité déjà construites). S'il est non-réalisabe, alors arrêter,le problème d'origine est non-réalisable ; sinon, stocker le vecteur de solution (x_{k+1}, θ_{k+1}) et poser $F_k(x_{k+1})$=$c^T x_{k+1} + \sum_{s=1}^{S} p_s \theta_{k+1,s}$.

Étape 4. Si $F_k(x_{k+1})$=$F(z_k)$ alors arrêter ; z_k est la solution optimale.

Étape 5. Enlever quelques coupes d'optimalité et de réalisabilité dont les multiplicateurs Lagrange après la résolution du (2.13) sont nuls. Poser $k \to k+1$, retourner à l'Étape 1.

À l'issue de l'étape 1, en ayant à disposition les termes $f_s(x_k)$ et θ_s (c'est-à-dire les coûts réel et approximé) pour chaque scénario s on construit les quantités $F(x_k)$ le coût actuel du problème-maître et $\hat{F}_k(x_k)$ le coût approximé du problème-maître. À la deuxième étape on compare le coût actuel $F(x_k)$ avec le le coût approximé et le coût réel de la solution de référence à l'itération précédente. Le coefficient γ est choisi par l'utilisateur et il donne une sorte de compromis : si γ=1 par exemple on remplace toujours la solution de référence z_k par la solution courante x_k ; si en revanche γ=0, alors on choisit d'échanger la solution de référence contre la x_k seulement si son coût réel est meilleur. Enfin, le coefficient σ peut lui aussi être controlé en temps réel de façon à ce qu'il baisse si la différence $F(x_k) - F(z_{k-1})$ est petite et qu'il augmente si $F(x_k) > F(z_k)$.

La stabilisation de la méthode avec l'introduction de la notion de la solution de référence et du terme quadratique peut avoir comme effet la réduction du nombre d'itérations ; elle évite donc l'augmentation du nombre de coupes au problème-maître. De plus, à l'étape 5, on efface un certain nombre de coupes qui ne sont pas actives à la solution optimale. Malheureusement, il n'a pas moyen de savoir si les coupes qui ont été effacées seront utiles ultérieurement dans l'algorithme. La suppression de coupes est donc une tâche heuristique.

2.4 Décomposition stochastique

Si on se restreint aux coupes basiques, on obtient une méthode exacte. Pourtant les différentes bases pour les solutions duales peuvent être extrêmement nombreuses ce qui rendrait la résolution du dual fastidieuse. En 1997 Higle et Sen [36] ont proposé l'idée de chercher les solutions des sous-problèmes non pas parmi toutes les bases duales, mais parmi une partie d'entre elles déterminée par échantillonnage statistique. L'optimisation alors faite au second niveau n'est pas une optimisation exacte mais approchée. On souligne cépendant qu'une telle idée est légitime et applicable si on a la certitude que les sous-problèmes admettent toujours une solution pour n'importe quelle solution du premier niveau en faisant l'hypothèse qu'il n'y a pas d'incertitude dans la fonction objectif. On reprend la formulation du primal et du dual comme dans les for-

mules (2.3). Le polyhèdre du dual avec une telle hypothèse prise en compte est défini comme l'ensemble de λ tels que :

$$\begin{aligned} \lambda^T W &\leq q \\ \lambda &\geq 0 \end{aligned} \tag{2.14}$$

Les x et ξ (la solution du premier niveau et la réalisation de la variable aléatoire respectivement) n'interviennent pas dans la définition de ce polyhèdre. Ceci implique que s'il existe un seul x et un seul ξ pour lequel (2.14) est réalisable, alors la réalisabilité du problème est assurée pour n'importe quelle solution x et n'importe quelle réalisation de la variable aléatoire ξ.

Au départ on démarre avec une solution du premier niveau. On la fait passer aux sous-problèmes et on calcule les variables duales. À chaque itération, on génère un scénario, on résout le programme associé à ce scénario et on rajoute la solution duale optimale dans l'ensemble de bases dont on dispose jusqu'à ce point de l'algorithme. Pour ajouter une coupe au problème-maître, il nous faut les valeurs duales des autres scénarios (ceux qui ont été générés précédemment) portant sur la solution du problème-maître. Or, au lieu de résoudre tous les sous-problèmes (associés à ces scénarios) en ayant comme espace de solutions toutes les valeurs duales du polyhèdre (2.14), on les résout parmi les valeurs duales déjà collectionnées. Une telle stratégie rend d'un côté le calcul très rapide, mais de l'autre sous-optimal. Mathématiquement, on résout alors le problème suivant :

$$\max_{\lambda} \{\lambda^T(\xi_j - T(\xi_j)x_k) \mid \lambda \in \mathcal{V}_k\} \tag{2.15}$$

où $\mathcal{V}_k$ est l'ensemble des bases réalisables à l'itération k. Dès lors, afin de calculer une borne inférieure *statistique* de la fonction de recours on calcule la moyenne suivante :

$$N^{-1} \sum_{j=1}^{N} \lambda^T(\xi_j - T(\xi_j)x_k)$$

Au départ on suppose disposer d'un ensemble de scénarios potentiellement infini (variable aléatoire continue). À la première itération, la fonction de recours est

représentée par un support qui porte sur un seul scénario. À la deuxième itération, un autre scénario sera généré. La première coupe – ajoutée au maître lors de la première itération – est mise à jour et la deuxième coupe sera rajoutée au maître en représentant cette fois-ci deux scénarios. En continuant ainsi, à l'itération k, le $k^{\text{ème}}$ scénario sera généré, la coupe rajoutée au maître comportera des coefficients (valeurs duales et fonction objectif) des tous les scénarios, alors qu'en même temps les coupes générées précédemments seront mises à jour.

L'algorithme est formellement décrit ci-dessous. Le terme ξ désigne une réalisation d'une variable alétoire continue et il est équivalent à un scénario.

On essaye alors de mettre à jour les coupes approximées auparavant suivant l'information obtenue au fur et à mesure des itérations. Comme la méthode est une méthode stochastique, le critère de terminaison n'est pas exact mais statistique.

Dans le même travail [36], les auteurs identifient la difficulté de l'algorithme d'approximer de mieux en mieux – à travers la mise à jour des anciennes coupes – la fonction de recours. Pour remédier à ce problème ils proposent d'introduire une solution de référence. La mise à jour de cette solution de référence est faite à l'issue de chaque itération. Elle est soit remplacée par la solution actuelle, soit elle reste inchangée. L'idée est proche de la décomposition régularisée.

ALGORITHME DE LA DÉCOMPOSITION STOCHASTIQUE

Étape 0. Initialiser : k=0, $\mathcal{V}_k$=$\emptyset$, ξ_0=$\mathbb{E}[\xi]$. On considère avoir une solution du premier niveau x_0.

Étape 1. $k \leftarrow k+1$. Générer de manière aléatoire et indépendante un scénario ξ_k de la distribution F_ξ.

Étape 2. Résoudre le sous-problème portant à ce scénario

$$\min q^T y \quad \text{s.c} \quad Wy = \xi_k - Tx_k$$

pour obtenir une solution duale optimale $\lambda(x_k, \xi_k)$. Mettre à jour l'ensemble de bases duales $\mathcal{V}_k \leftarrow \mathcal{V}_{k-1} \cup \{\lambda(x_k, \xi_k)\}$.

Étape 3. Dans cette étape on calcule les coefficients de la nouvelle coupe et on met à jour les coefficients des coupes générées jusqu'à ce stade (itération k). L'indice μ_t^k signifie que le terme μ appartenant à la coupe générée à l'itération t est à nouveau mis à jour à l'itération k.

a Calculer les coefficients de la nouvelle coupe :

$$\alpha_k^k + (\beta_k^k + c)x \equiv cx + \frac{1}{k}\sum_{t=1}^{k} \lambda_t^k (\xi_k - Tx)$$

Chaque λ_t^k est calculé en résolvant le problème :

$$\lambda_t^k = \arg\max\{\lambda(\xi_t - Tx_k) \,|\, \lambda \in \mathcal{V}_k\}$$

b Mettre à jour les coupes précédemment générées :

$$\alpha_t^k \leftarrow \frac{k-1}{k}\alpha_t^{k-1}, \quad \beta_t^k \leftarrow \frac{k-1}{k}\beta_t^{k-1}, \quad t{=}1, ..., k-1.$$

Étape 4. Résoudre le problème maître avec la nouvelle coupe rajoutée et les coupes mises à jour à l'Étape 3. Obtenir une solution x_{k+1}. Retourner à l'Étape 1.

2.5 Programmation mixte

La version de la méthode "L-Shaped" où les variables du premier niveau sont purement binaires a été présentée par Laporte et Louveaux en 1993 [45]. Le choix des variables du premier niveau se fait par branch & bound alors que les coupes au second niveau (d'optimalité et de réalisabilité) se calculent comme avant. Si pourtant une variable du premier niveau n'est pas binaire, alors la méthode ne marche plus.

Lorsque le second niveau contient des variables qui doivent être entières les méthodes des plans sécants échouent pour une raison principale : la fonction de recours n'est plus convexe ni continue. La figure 5 montre un cas simple d'une telle fonction de recours. Dans cette figure la fonction de recours est semi-continue inférieurement. Ceci n'est pas particulier à certaines fonctions de recours mais c'est un résultat général lorsque $\mathbb{E}[\xi] < \infty$ [67].

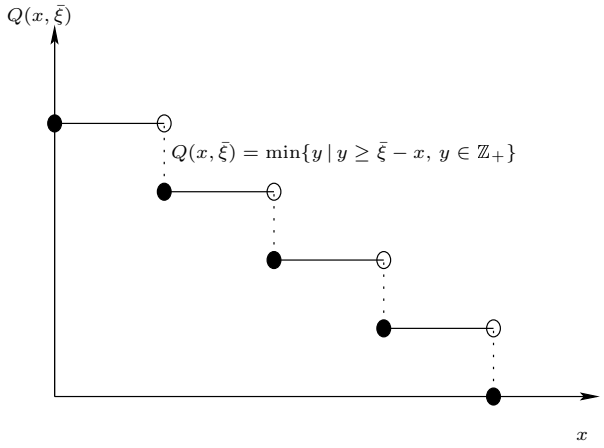


Fig. 5: Fonction de recours non-convexe et discontinue. Cela se produit lorsque quelques variables de recours sont entières.

La méthode qui permettrait d'avoir des variables mixtes au premier ainsi qu'au second niveau a été développée en 1998 par Caroe et Tind [24] mais les auteurs recourent à la programmation non-linéaire afin d'approximer la fonction de recours.

D'après l'étude bibliographique de Sahinidis en 2004 [66], la programmation stochastique en nombres entiers est de plus en plus étudiée du point de vue de la convexi-

fication. D'après l'auteur, Higle et Sen ont proposé des algorithmes liées à l'idée de *"lift-and-project"* (Balas, Ceria et Cornuéjols [2]) dans le contexte de la méthode "L-Shaped". Cépendant, il n'y a pas encore d'implémentations de ces nouveaux algorithmes.

Pour une présentation complète des méthodes sur la programmation stochastique en nombres entiers, on se réfère à Klein Haneveld et van der Vlerk [40].

2.6 Conclusions

Les problèmes en deux étapes avec recours présentent en général deux difficultés importantes : (i) la taille des problèmes, qui augmente vite avec le nombre de scénarios et (ii) le calcul de la fonction de recours. On procède à la discrétisation de la variable alétoire afin de pouvoir se ramener à la résolution d'un problème linéaire (sinon, à la place de la moyenne $\sum_i p_i \theta_i$ on aurait à calculer une intégrale).

Le nombre de scénarios peut néanmoins défavoriser la convergence ou augmenter considérablement la taille du problème. La décomposition stochastique peut pourtant traiter un nombre potentiellement inifini de scénarios, en discrétisant aussi finement que souhaité la variable aléatoire. Des techniques d'échantillonnage ont été développées [42] dans l'objectif de réduire la taille des problèmes en second niveau en donnant des estimations statistiques sur la fonction de recours.

Dans l'ensemble, la littérature est assez riche concernant des méthodes qui abordent des problèmes en deux étapes. Surtout pour le recours simple il existe plusieurs travaux ([80] [78]) qui établissent des résultats pour les problèmes linéaires continus, aussi bien que pour les problèmes quadratiques [57] ainsi qu'en nombres entiers [41].

Dans le chapitre suivant, les difficultés que l'on a identifiées chez les problèmes en deux étapes se multiplient, puisque on étend l'étude en plusieurs périodes temporelles en rendant la complexité des problèmes beaucoup plus importante. Certaines méthodes qui existent pour la résolution des modèles en deux étapes n'ont pas été mentionnées dans le présent chapitre, puisque elles peuvent être généralisées au cas de modèles multi-étapes. On les a pour cette raison reportées au chapitre suivant, où on introduira les modèles multi-étapes et on étudiera leurs méthodes de résolution.

Chapitre 3

Modèles multi-étapes

3.1 Introduction

Les modèles multi-étapes sont l'extension des modèles en deux étapes. On se restreindra dans ce qui suit aux problèmes où l'optimisation se fait sur un espace de scénarios discret. On considère alors plusieurs étapes successives de décision qui correspondent le plus souvent aux pas de temps. On précise que dans la suite, le terme *étape* sera équivalent au terme *période*. À chaque instant où une décision doit être prise, le décideur peut utiliser toute l'information déjà disponible jusqu'à ce moment ainsi que la connaissance de toutes les décisions déjà prises.

Dans les modèles multi-étapes on travaille souvent sur un arbre de scénarios comme celui de la figure 1. Cela suppose implicitement que la discrétisation des variables aléatoires se fasse de telle sorte à ce que tout l'univers des évènements possibles puisse être décrit par cet ensemble de scénarios. Dans la suite, sauf mention contraire, on supposera que le nombre de scénarios à notre disposition est fini.

Il y a un aspect crucial qui doit être correctement représenté dans les modèles multi-étapes : la *non-anticipativité*. La différence de la programmation dynamique déterministe par rapport à la programmation dynamique stochastique est le fait que dans le premier cas le décideur a une information parfaite d'ici jusqu'à la fin de l'horizon d'étude. Dans un problème de gestion de stocks où la demande est dynamique

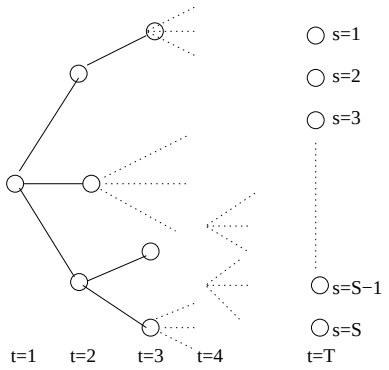


Fig. 1: Un arbre de scénarios : T pas de temps, S scénarios.

mais déterministe, la quantité que l'on décide de mettre en stock le moment t peut tout légitimement dépendre du niveau de la demande au moment $t+k$. Le planning se fait de telle manière que l'on puisse anticiper dès le moment t une augmentation de la demande au moment $t+k$.

Or, dans la programmation dynamique stochastique où on optimise dans un espace de scénarios, la notion du scénario n'a un sens qu'une fois la fin de l'horizon atteinte. À chaque pas de temps il y a une partie de l'information qui se révèle au fur et à mesure. Or le scénario représente une évolution possible depuis le début jusqu'à la fin de l'horizon. C'est seulement lorsque l'on sera arrivé à la fin de l'horizon d'étude que l'on pourra dire que le scénario s est l'enchaînement des évènements $\{\omega_{1s}, \omega_{2s} \dots \omega_{ts}\}$ qui se sont produits à chaque pas de temps. Schématiquement, au moment $t{=}k{\neq}T$ de l'arbre de la figure 1 on ne peut pas dire dans quel scénario on se trouve. La notion de scénario apparaît à l'instant T. On dit souvent que dans un modèle multi-étapes il existe un enchaînement des décisions et des observations qui suit la loi : *(décision)* $\rightsquigarrow$ (*observation*) $\rightsquigarrow \dots \rightsquigarrow$ *(décision)* $\rightsquigarrow$ (*observation*).

Il existe deux voies pour modéliser la non-anticipativité. La première est immédiate : on adopte la vision d'arbre de de scénarios. On reprend la figure 1, on enlève la numérotation sur les feuilles (qui correspond à la numérotation des scénarios) et on numérote tous les nœuds en commençant par la racine. Vu du côté de la programmation dynamique, à chaque nœud on minimise la fonction de Bellman qui est maintenant une

espérance sur l'espace de différents scénarios pouvant se produire à partir du nœud courant. Mathématiquement, cette vision est représentée par le programme linéaire suivant (les termes génériques f_i sont des produits $c_i^T x^i$) :

$$\begin{array}{ll} & \min f^1(x^1) + \sum_{i\in\mathcal{N}\setminus\{1\}} p^i f^i(x^i) \\ \text{sc} & T^i x^{\alpha(i)} + W^i x^i = h^i, \quad i \in \mathcal{N}\setminus\{1\} \\ & W^1 x^1 = h^1 \\ & x^i \in \mathcal{X}^i \end{array} \tag{3.1}$$

$\mathcal{N}$ est l'ensemble de nœuds dans l'arbre et $\alpha(i)$ désigne le prédécesseur direct du nœud i.

La deuxième vision de la non-anticipativité est la voie explicite. On crée autant de séquences d'évolution possibles que de scénarios. Supposons que l'on doit prendre une décision au moment t'. Supposons également que pour deux scénarios s_1, s_2, qui partagent le même passé jusqu'au moment t', on prend deux décisions différentes. Ceci signifierait que la décision du moment t' dépend de la réalisation qui ne s'est pas encore produite. Or, on n'est pas censé connaître déjà cette réalisation. Nos décisions doivent faire face à tout scénario possible à partir de l'instant t'. Afin d'éviter de prendre des décisions qui supposent connue les réalisations ultérieures, on introduit les contraintes dites de *non-anticipativité*.

On dispose maintenant des scénarios parallèles dont les variables se couplent suivant leur position sur l'arbre. La figure 2 présente un exemple de 9 scénarios dispersés en 4 pas de temps, dont les variables se couplent suivant la structure de l'arbre affiché à gauche. Les scénarios parallèles avec des contraintes de non-anticipativité consituent une vision duale de l'arbre de scénarios. À titre alors d'exemple, les variables de décision du premier au quatrième scénario à l'instant t=2 doivent absolument avoir la même valeur, afin de respecter la condition de la non-anticipativité. Cette vision est représentée par la formulation (3.2).

$$\begin{array}{ll} & \min \sum_s p^s f^s(x^s) \\ \text{sc} & W^s x^s = h^s \quad \forall\, s \\ & H x^s = 0 \\ & x^s \in \mathcal{X}^s \end{array} \tag{3.2}$$

où les contraintes Hx^s=0 représentent les contraintes de non-anticipativité (x_1=x_2, x_2=x_3, ...).

Une solution du problème anticipatif n'est pas une solution applicable ; elle donne pourtant une borne inférieure sur la fonction de coût, puisque pour l'obtenir on a oublié un certain nombre de contraintes.

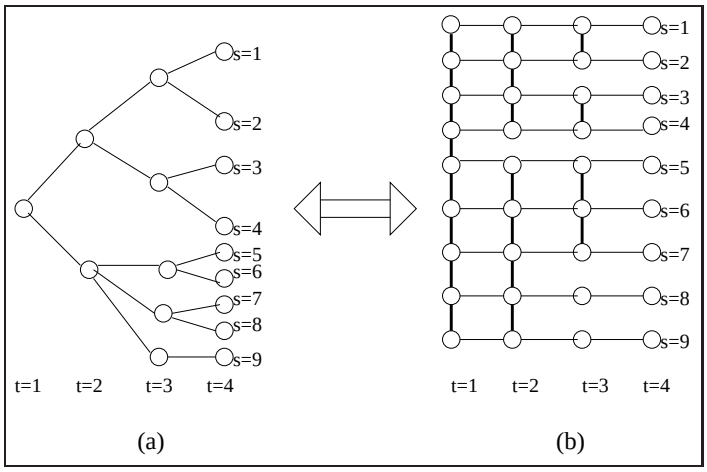


Fig. 2: Vision de l'arbre du point de vue des contraintes de non-anticipativité.

Comme on l'a déjà vu au chapitre précédent, la taille des problèmes de recours en deux étapes augmente linéairement avec le nombre de scénarios. Or, quand on passe aux problèmes multi-étapes, les choses sont différentes. La taille des problème augmente exponentiellement en fonction du nombre d'étapes de décision. En restant modeste, on considère un problème en 5 étapes avec 10 réalisations de la variable aléatoire à chaque étape ; tout de suite on passe à un nombre de scénarios qui atteint les 100.000. Les travaux [34] ont résolu des problèmes qui comportaient 13 périodes et presque 5 millions de scénarios ont été résolu. Les auteurs indiquent qu'à leur connaissance ceci

était un record à cette époque (2000). Il s'agissait d'un problème d'optimisation sur la gestion de l'actif et du passif pour un organisme de retraite hollandais. La méthode utilisée a été la décomposition imbriquée présentée ci-après et la résolution des sous-problèmes linéaires correspondant aux nœuds a été effectuée avec des méthodes de points intérieurs. Birge en 2003 [17] souligne que le plus grand problème traité jusqu'à ce jour a été un problème de 4 étapes et de plus de 30 million de scénarios. En 2003 [22] on rencontre un problème dans le domaine de la finance (selection d'actifs) qui comportait 9 étapes et 5.8 millions de scénarios. La méthode de résolution a été une méthode de points intérieurs.

Pour attaquer ce type de problèmes il faudrait prendre en compte la structure particulière sous-jacente et développer des méthodes spécifiques qui rendrait soit une grande partie du problème plus facile à résoudre, soit tout le problème décomposable. Plusieurs auteurs ont travaillé sur la décomposition de modèles multi-étapes, afin de paralléliser et ainsi accélérer la résolution. La décomposition peut se faire soit par nœuds soit par scénarios, suivant la modélisation adoptée.

3.2 Modélisation du flux d'information

Rappelons la vision d'un arbre de scénarios parallèles de la figure 2b. On souhaite décrire mathématiquement le couplage des variables à différents niveaux. Rappelons que les décisions aux moments discrets $t \in \mathbb{N}$ ne peuvent dépendre que de l'information disponible depuis l'instant initial jusqu'à l'instant t où la décision doit être prise.

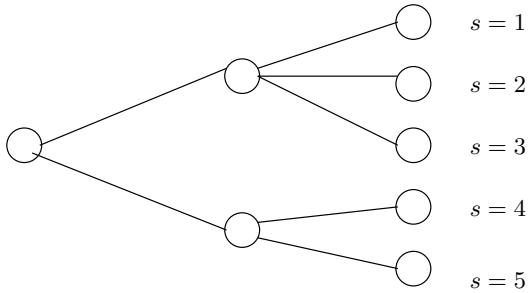
Tous les états possibles du vecteur de la variable aléatoire sont représentées par l'ensemble Ω dont les élements sont désignés par ω. En ayant défini cet espace d'évènements possibles, il faut définir différentes partitions possibles aussi bien qu'une mesure qui évaluera chaque évènement. Cette mesure est la probabilité $\mathbb{P}$ et toute partition possible est la $\sigma-$algèbre[1] $\mathcal{F}$ de l'espace Ω. Cet ensemble Ω muni d'une $\sigma-$algèbre $\mathcal{F}$ est désormais appelé *espace mesurable*, la probabilité $\mathbb{P}$ étant sa mesure. En notant qu'une condition doit être vérifiée pour presque tout $\omega \in \Omega$ (on note a.s : *almost surely*)

[1] Rappel : σ-algèbre $\mathcal{F}$ d'un ensemble Ω est une collection non-vide d'ensembles qui est fermée pour le complémentaire et pour les unions finies.

on entend que la condition doit être valable pour tous ces évènements ω qui ont une mesure non-nulle dans l'espace mesurable où ils appartiennent (cela revient à exclure tout évènement d'une probabilité de réalisation nulle).

Dans les modèles multi-étapes, on considère une séquence des $\sigma-$algèbres qui sont générées itérativement $\mathcal{F}_0 \supset \mathcal{F}_2 \supset ... \supset \mathcal{F}_T$ (cf à la figure 3). $\mathcal{F}_t \supset \mathcal{F}_{t+1}$ signifie que les ensembles générés par $\mathcal{F}_{t+1}$ sont des sous-ensembles des ceux générés par $\mathcal{F}_t$. Ceci modélise le flux d'information qui doit être pris en compte lors de la décision au moment t. Plus précisément l'aspect modelisé est le filtrage de l'information lors du passage de l'instant $t-1$ à l'instant t. On définit pour chaque instant t (le pas de temps des décisions et/ou des réalisations des aléas) les vecteurs de décisions $x_t \in X_t$ qui correspondent à des applications mesurables par rapport à la $\mathcal{F}_t$. On note bien que $x_t(\omega)$ doit absolument être $\mathcal{F}_{t-1}$-mesurable, à savoir, constant pour tout élément dans chacun des sous-ensembles générés au moment $t-1$ par la $\sigma-$algèbre $\mathcal{F}_{t-1}$. L'exigence de la mesurabilité du vecteur $x_t(\omega)$ correspond aux contraintes de non-anticipativité.

En ayant défini le contexte dans lequel on se positionne, on décrirera brièvement les principales méthodes proposées pour résoudre les modèles multi-étapes.



$$\mathcal{F}_0 = \{1,2,3,4,5\} \quad \mathcal{F}_1 = \begin{matrix}\{1,2,3\}\\ \{4,5\}\end{matrix} \quad \mathcal{F}_2 = \begin{matrix}\{1\}\\ \{2\}\\ \{3\}\\ \{4\}\\ \{5\}\end{matrix}$$

Fig. 3: Exemple d'un arbre de scénarios où on fait apparaître le flux d'information : on génère une filtration à chaque période t $\mathcal{F}_0 \supset \mathcal{F}_1 \supset \mathcal{F}_2$. Les éléments de $\mathcal{F}_t$ sont des scénarios s.

3.3 Décomposition imbriquée

La méthode de référence pour la résolution des problèmes multi-étapes due à Birge [15] est l'extension de la méthode "L-Shaped" utilisée dans les modèles avec recours en deux étapes. Elle s'appelle *décomposition imbriquée*(cf aussi à [64]) ; dans la bibliographie elle est aussi appelé *"L-Shaped" imbriquée* ou encore *méthode de programmation dynamique stochastique duale* (par exemple [53]) .

La méthode s'applique sur un arbre de scénarios couplés comme il a été expliqué dans le paragraphe 3.1. Chaque nœud de l'arbre contient une réalisation des paramètres aléatoires intervenant dans le problème. La modélisation mathématique suit celle du programme (3.1) dont la matrice a la forme suivante :

$$\begin{bmatrix} W^1 & & & & & & \\ T^2 & W^2 & & & & & \\ & T^3 & W^3 & & & & \\ & & & \ddots & & & \\ & & & & T^{N-1} & W^{N-1} & \\ & & & & & T^N & W^N \end{bmatrix} \tag{3.3}$$

Le couplage par blocs de variables représente la dynamique du système, il suit la structure de l'arbre de scénarios et rend le problème difficile à résoudre. En observant la forme de la matrice (3.3), une première idée serait de considérer chaque bloc de lignes comme un problème de recours en deux étapes et lui appliquer la méthode "L-Shaped" comme décrite auparavant.

La décomposition se fera donc au niveau des nœuds. Dans le cadre de la méthode, un programme linéaire est associé à chacun de nœuds ; ce programme, à part les contraintes locales qui ne contiennent que des variables propres à ce nœud, contient également un nombre augmentant de coupes qui approximent la fonction de Bellman. L'algorithme de décomposition imbriquée consiste à parcourir tous les nœuds de l'arbre ; en chaque nœud, récupérer les solutions du nœud prédécesseur, récupérer les coupes de réalisabilité et d'optimalité des nœuds successeurs et résoudre le problème suivant :

$$
\begin{array}{ll}
 & \min \left\{ c_i^T x_i + \theta_i \right\} \\
\text{s.c} & A_i x_i = b_i \\
 & W_i(\xi) x_i = h_i(\xi) - T_i(\xi) x_{\alpha(i)} \\
 & \sum_{j \in \mathcal{S}(i)} p_{ij} (a_{kj} + g_{kj}^T x_i) \leq \theta_i, \quad k \in \mathcal{J}_{obj}^j \\
 & \beta_j + r_j^T x_i \leq 0, \quad j \in \mathcal{S}(i) \\
 & x_i \geq 0
\end{array}
\tag{3.4}
$$

où $\alpha(i)$ désigne le nœud-père du nœud i, tandis que $\mathcal{S}_(i)$ l'ensemble de successeurs du nœud i. La probabilité conditionnelle de se trouver au nœud j en provenance du nœud i est notée p_{ij}. $\mathcal{J}_{obj}$ est l'ensemble d'indices portant chacun sur une coupe d'optimalité. Dans une hypothèse de recours (relativement) complet, où il n'y aurait pas de coupes de réalisabilité, l'indice k serait égal à l'indice des itérations.

À chaque itération on effectue deux passages sur l'arbre de scénarios : un passage en partant du nœud-racine et en avançant vers les feuilles, et un passage en provenance des feuilles à destination de la racine. Les figures 4 et 5 illustrent la méthode pendant le passage vers l'avant (figure 4) et vers l'arrière (figure 5) respectivement lors de la première itération.

Lors du passage vers l'avant, on visite tous les nœuds séparément et on résout le programme linéaire (3.4) correspondant au nœud associé. Remarqons que lors de la première itération les contraintes portant sur les coupes de réalisabilité et d'optimalité ne seront pas présentes. Le terme θ sera borné par un grand nombre négatif ($\theta \geq -M$) pour assurer que chaque problème ait une solution d'un coût fini. Si le problème (autre que le problème du nœud-racine) est non-réalisable, alors on ajoute au problème appartenant au nœud-père une coupe de réalisabilité ((3.4) quatrième groupe de contraintes). Cette coupe est construite en récupérant la direction duale du problème non-réalisable. Si la non-réalisabilité a été constatée sur le problème du nœud-racine, l'algorithme s'arrête : en effet dans ce cas le problème est globalement non-réalisable. Au moment où le nœud cosidéré admettra une solution optimale (c'est-à-dire au moment où il n'y aura plus de coupes de réalisabilité à ajouter à l'itération considérée), on passe à son nœud-fils. Le nœud-fils récuperera la solution optimale de son nœud-père, et à son tour, il proposera une solution (sa solution optimale) à son nœud-fils. Le passage

vers l'avant continue ainsi.

Lorsque tous les nœuds seront considérés, on change de direction[2] de traitement de nœuds et on commence le passage vers l'arrière. En récupérant la solution duale optimale, ainsi que la valeur optimale de la fonction de coût du nœud-fils, on construit une coupe d'optimalité au nœud-père ((3.4) troisième groupe de contraintes).

Le parcours des nœuds sur l'arbre, comme il a été décrit ci-dessus, est le protocole *"Fast Forward Fast Backward"*. Il y a plusieurs façons de parcourir l'arbre qui ont été proposées dans la littérature. Birge [15] a implémenté le protocole *"Fast Forward"*. Il consite à passer des nœuds de la période t aux nœuds de la période $t+1$ seulement si tous les nœuds contiennent des solutions optimales. Gassmann [33] a proposé le protocole *"Fast Backward"* qui consiste à remonter de la période t à la période $t-1$ seulement si tous les nœuds contiennent des solutions optimales. Parmi les trois protocoles, le *"Fast Forward Fast Backward"* a été montré comme étant le plus efficace ([21] et [33]).

[2] On peut également changer de direction lors du passage vers l'avant, lorsque on construit des coupes de réalisabilité, mais on considère que ceci fait également partie du passage vers l'avant.

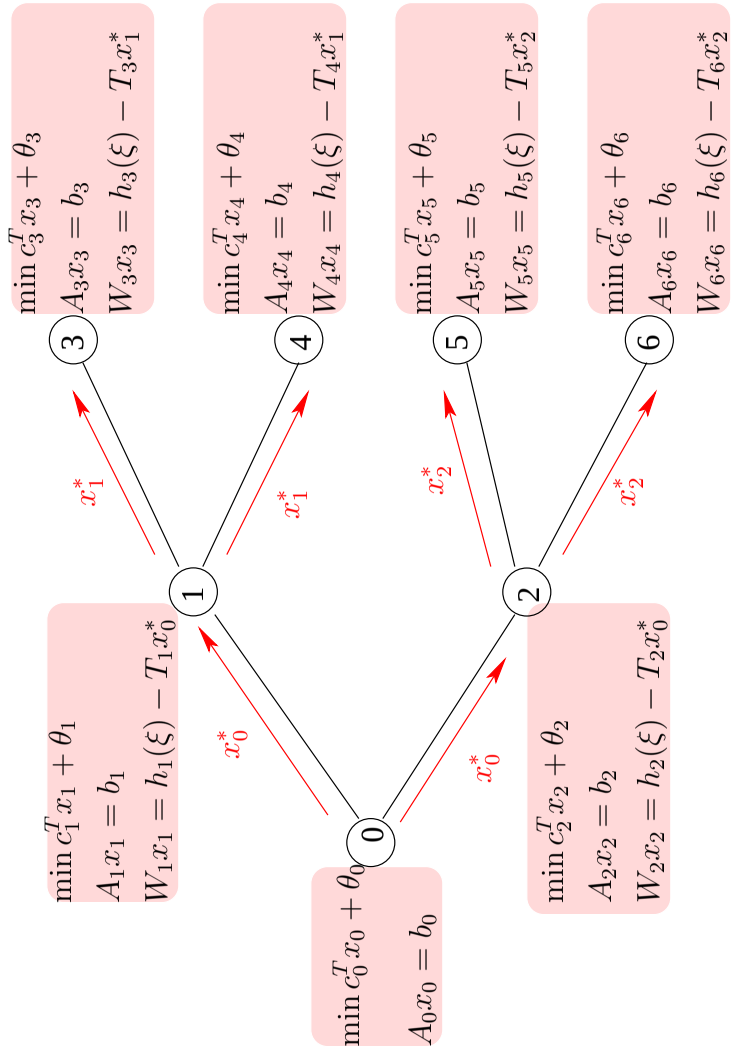


Fig. 4: Passage vers l'avant durant la première itération de la décomposition imbriquée.

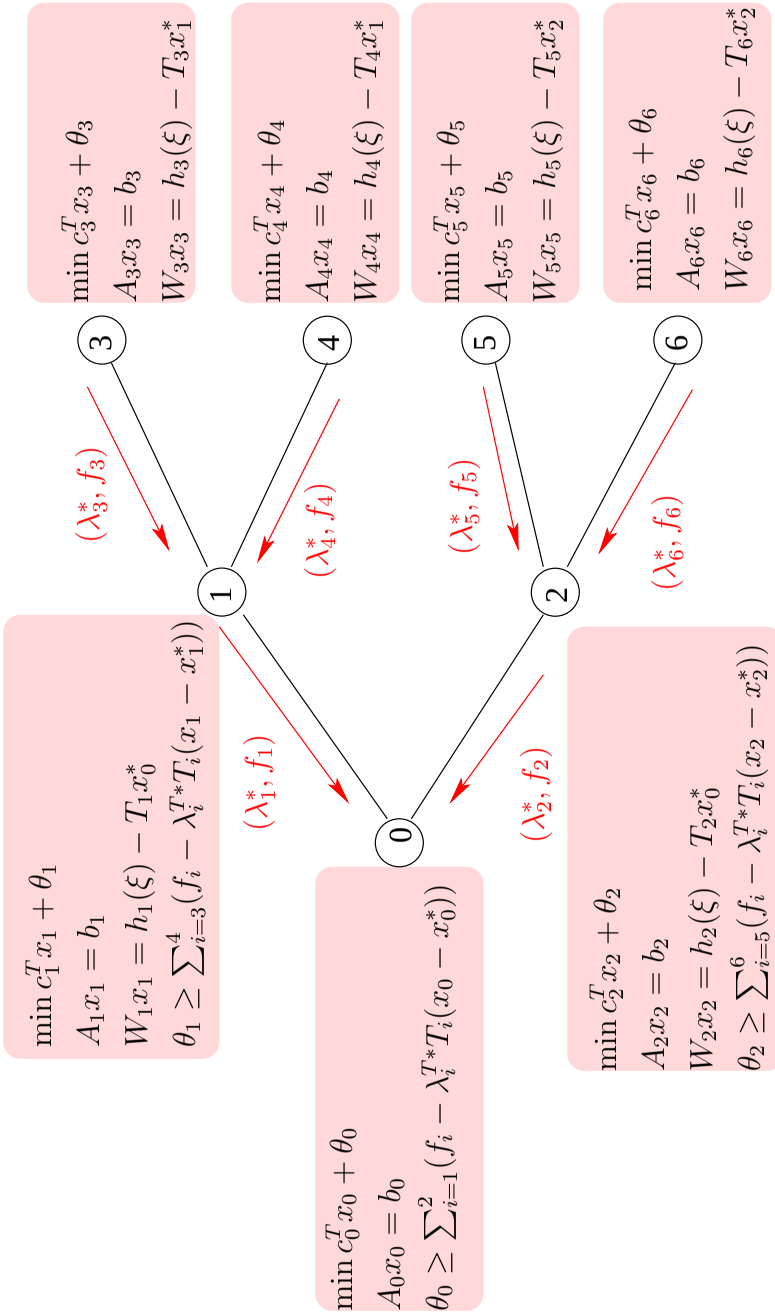


Fig. 5: Passage vers l'arrière durant la première itération de la décomposition imbriquée.

On appelera les problèmes associés à chaque nœud de l'arbre des *sous-problèmes*. L'ordre dans lequel on traite les sous-problèmes associés à chacun des nœuds n'est pas spécialement important. Néanmois, il n'y a aucune raison de s'ocuper du sous-problème du nœud i si on n'a aucune nouvelle solution $x_{\alpha(i)}$ et aucune nouvelle coupe provenant d'au moins un sous-problème des nœuds successeurs ($j \in \mathcal{S}(i)$). D'autre part, si au cours des itérations, une solution a été mise à jour, une des étapes suivantes sera forcément de résoudre les sous-problèmes des nœuds successeurs. Si une nouvelle coupe a été produite, une des étapes suivantes devrait être de résoudre le sous-problème du nœud-père. L'algorithme s'arrête lorsque toutes les fonctions de recours de tous les nœuds sont suffisamment bien approximés, ou si le coût réel de toutes les solutions proposée à tous les nœuds de l'arbre est suffisamment proche du coût approximé au premier nœud. Dans tous les cas où un de critères d'arrêt est vérifié, la solution à chaque nœud est considérée comme étant la solution optimale. L'arbre contient à chacun de ses nœuds la politique optimale qui devra être appliquée pour arriver au coût associé au nœud-racine. L'algorithme converge de façon finie (cf à [64] par exemple).

La décomposition imbriquée existe aussi en sa version *régularisée* comme celle-ci a été décrite pour les modèles en deux étapes (cf au paragraphe 2.3). Rappelons que dans le cas de deux étapes la solution de référence est remplacée par la solution actuelle suivant un critère qui porte sur la valeur réelle et aproximée de la fonction de recours. Dans un sous-problème multi-étapes chaque nœud inclut une estimation de la fonction de recours (fonction *"cost-to-go"*) à partir de ce nœud jusqu'à la fin de l'horizon. La solution de référence porte juste sur le nœud-racine de l'arbre et donc les valeurs réelles et approximées de la fonction de recours sont essentiellement les valeur réelle et approximée de la fonction *"cost-to-go"* du nœud-racine.

Une autre amélioration de la méthode consisterait à réduire le nombre de coupes qui augmentent vite dans tous les nœuds. On pourrait partager certaines coupes qui sont identiques à certains nœuds, surtout dans le cas où les scénarios sont issus d'une variable aléatoire dont le support est borné (Ξ=$[\xi_1, \xi_2]$) et les étapes sont indépendantes. Dans ce cas simplifié, si les sous-problèmes dans deux nœuds sont identiques, une résolution d'un des nœuds fournirait les coupes nécessaires pour tous les autres.

D'autres améliorations sont également envisageables, comme par exemple des

bases partagées entre plusieurs scénarios ou encore une solution issue d'un scénario qui servira de solution initiale (*"hotstart"*) pour les autres.

3.4 L'algorithme de couverture progressive (*Progressive hedging*)

La méthode de couverture progressive proposée par Rockafellar et Wets [58] a ouvert la voie vers une nouvelle approche de traitement des problèmes stochastiques aves recours, celle de relaxation avec langrangien augmenté. La méthode est appliquée sur la modélisation par scénarios (cf au paragraphe 3.1) et donc la relaxation s'entend sur les contraintes de non-anticipativité.

On considère un ensemble de scénarios $\mathcal{S}=\{1,2,...,S\}$. Le problème associé à chaque scénario est un problème dynamique, comme le (3.5) :

$$\begin{array}{ll} & \min \sum_t c_t x_t \\ \text{s.c} & A_{t-1}x_{t-1} + B_t x_t = b_t \quad \forall t{=}1,...,T \\ & x_t \geq 0 \quad \forall t{=}0,...,T \end{array} \tag{3.5}$$

Soit deux scénarios s_1 et s_2 distincts qui partagent le même passé pendant tout instant $t \leq \bar{t}$. Les variables – qui leurs sont associées – $x_{\bar{t}}^{s_1}$ et $x_{\bar{t}}^{s_2}$ représentent les décisions prises à l'instant $\bar{t}$. Pour respecter la condition de non-anticipativité, il faut exiger que les variables associées à ces deux scénarios aient la même valeur pour tout instant $t \leq \bar{t}$.

Afin de définir ceci plus formellement, considérons les sous-ensembles $\mathcal{B}_i(t)$ (i est le compteur de différents sous-ensembles à l'instant t) générés par les σ-algèbres $\mathcal{F}_t$ à chaque instant t. Chaque sous-ensembe contient des scénarios qui partagent le même historique au moment t (filtrage itératif d'information : cf au paragraphe 3.2). Un exemple est donné à la figure 6. On pourrait également définir l'opération inverse $\mathcal{B}_s^{-1}(t)$ qui renvoie le groupe de contraintes de l'instant t dans lequel est contenu le scénario s. Pour l'exemple de la figure 6, $\mathcal{B}_3^{-1}(1){=}2$, puisque le scénario 3 appartient à $\mathcal{B}_2(1)$.

On définit les trois ensembles suivants :

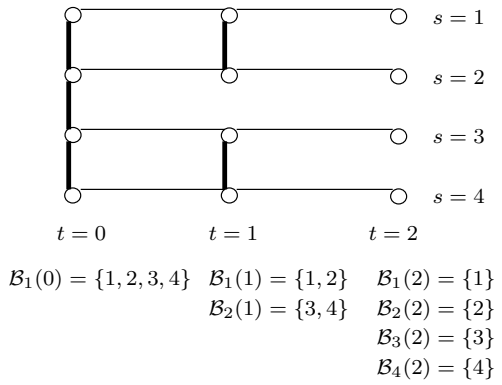


Fig. 6: Un exemple où apparaissent les différents groupes de scénarios qui partagent le même historique pour tout instant t. Tout sous-ensemble $\mathcal{B}_i(t)$ est généré de la σ-algèbre $\mathcal{F}_t$.

1. L'ensemble de décisions *implémentables* (*politiques*) qui satisfont les contraintes de non-anticipativité. Ces décisions ne respectent pas forcément les autres contraintes du problème. Notons cet ensemble $\mathcal{N}$.

$$\mathcal{N} := \{x_t^s \in \mathbb{R}^n \,|\, x_t^{s_1} = x_t^{s_2} \quad \forall s_1 \neq s_2 \quad \text{t.q} \quad \{s_1, s_2\} \in \mathcal{B}_i(t) \,\forall\, i, t\}$$

2. L'ensemble de décisions *admissibles* qui appartiennent dans l'ensemble X_s. Cet ensemble contient des décisions qui ne sont pas nécessairement non-anticipatives (elles dépendent des scénarios). Notons cet ensemble $\mathcal{C}$ (on suit la notation d'origine).

$$\mathcal{C} := \{x_t^s \in X_t^s \quad \forall\, s, t\}$$

3. Une décision qui appartient à la fois aux ensembles $\mathcal{N}$ et $\mathcal{C}$ est une décision *réalisable* et c'est elle que l'on s'efforcera de trouver.

   Dès lors, le problème se formule de manière compacte comme suit :

$$\min \sum_{s \in S} \sum_{t} p_{t,s} f(x_t^s) \quad \text{sous contraintes} \quad x_t^s \in \mathcal{N} \cap \mathcal{C} \tag{3.6}$$

où $p_{t,s}$ représente la probabilité de se retrouver à l'instant t au scénario s.

Il est bien connu que ce qui empêche la décomposabilité d'un programme multi-étapes comme le (3.6) est les contraintes de non-anticipativité. On relaxe cette famille de contraintes, on les ramène dans la fonction objectif de façon lagrangienne et on augmente la fonction avec un terme quadratique qui controlera l'écart entre la nouvelle solution x_t^s et une solution de référence $\bar{x}_t^i$, avec $i{=}\mathcal{B}_s^{-1}(t)$. La solution de référence est une solution commune aux variables de chaque groupe non-anticipatif. À titre d'exemple, à la figure 6, à la période $t{=}0$ il n'y a qu'une seule solution commune, à la période $t{=}1$ il y en deux, alors qu'à la période $t{=}2$ il y en a quatre.

Si on ecrivait les contraintes de non-anticipativité portant sur la figure 6, on aurait le système suivant[3] :

Contrainte				Multiplicateur
x_0^1	$=$	x_0^2	$\rightarrow$	λ_0^1
x_0^2	$=$	x_0^3	$\rightarrow$	λ_0^2
x_0^3	$=$	x_0^4	$\rightarrow$	λ_0^3
x_1^1	$=$	x_1^2	$\rightarrow$	λ_1^1
x_1^3	$=$	x_1^4	$\rightarrow$	λ_1^2

La forme générique des contraintes de non-anticipativité ainsi que de leurs multiplicateurs est la suivante :

$$\forall s \neq s' \in \mathcal{B}_t^{-1}(s): \quad x_t^s = x_t^{s'} \quad \rightarrow \quad \lambda_t^j \quad \forall j \in \mathcal{J}_t$$

L'ensemble $\mathcal{J}_t$ comporte les compteurs des contraintes de non-anticipativité à chaque intant t et pour tous les différents groupes de contraintes $\mathcal{B}_s(t)$. Par exemple, $\mathcal{J}_0 = \{1,2,3\}$ et $\mathcal{J}_1{=}\{1,2\}$.

Au cours de l'algorithme, on aura besoin de calculer à chaque itération une politique implementable, c'est-à-dire une décision $\bar{x}_t^i$ commune pour tous les scénarios $s \in \mathcal{B}_i(t)$. Cette décision est calculée ainsi :

$$\bar{x}_t^i = \sum_{s \in \mathcal{B}_i^{-1}(t)} p_s x_t^s \quad \forall i,\, t \tag{3.7}$$

[3]Le symbole "$\rightarrow$" devrait être interprété comme "*correspond à*" ou "*lui est associé*".

Remarquons qu'il s'agit d'une sorte de moyenne[4] pour chaque groupe de variables non-anticipatives.

Après la relaxation et l'ajout du terme quadratique, on obtient une famille de problèmes (un problème par scénario) comme suit :

$$\min \sum_t c_t^s x_t^s + \sum_{\substack{s' \neq s \\ s' \in \mathcal{B}_s(t)}} (x_t^s - x_t^{s'})\lambda_t^j + \frac{1}{2\rho}\|x_t^s - \bar{x}_t^i\|^2 \quad \forall s \tag{3.8}$$

où $j \in \mathcal{J}_t$ est l'ensemble de compteurs de contraintes de non-anticipativité et $i{=}\mathcal{B}_s^{-1}(t)$.

On est prêt à décrire formellement l'algorithme de couverture progressive. Toutes les quantités introduites auparavant qui changent de valeur d'une itération à l'autre seront indicées par k, le compteur des itérations. Pour permettre une écriture compacte on remplace $\mathcal{B}_s(t)$ par i.

ALGORITHME DE COUVERTURE PROGRESSIVE

Étape 0. Poser $k{=}0$. On suppose disposer des solutions $x_t^{s,k} \in \mathcal{C}$, ainsi que d'un vecteur de pénalités $\lambda_t^{s,k}$.

Étape 1. Calculer la solution de référence (politique implémentable) $\bar{x}_t^{i,k} \in \mathcal{N}$ par la formule (3.7).

Étape 2. Calculer la politique admissible suivante, $x_t^{s,k+1}$, en résolvant le programme (3.8) pour tous les scénarios.

Étape 3. Mettre à jour les multiplicateurs suivant la règle :

$$\lambda_t^{i,k+1} = \lambda_t^{i,k} + \alpha\|x_t^{s,k} - \bar{x}_t^{i,k}\|$$

À l'Étape 0 on pourrait commencer avec les solutions optimales pour chaque scénario séparément. Le vecteur de multiplicateurs peut être par exemple nul. À l'Étape 1, on obtient des solutions implémentables mais pas forcément admissibles. En revanche,

[4] Il n'est pas nécessaire d'après les auteurs que le terme p_s s'interprète comme une probabilité. Cela pourrait également être l'importance que l'algorithme accorde au scénario s.

à l'Étape 2, on obtient des solutions admissibles, pas nécessairement implémentables. Si à n'importe quel moment on souhaite arrêter l'algorithme, la solution à proposer sera la solution candidate (implémentable) calculée à l'Étape 1.

Les séquences générés sont les $\{x_t^{i,k}\}$ et $\{\lambda_t^{i,k}\}$. Pour que $\bar{x}^k \to x^*$ et $\lambda^k \to \lambda^*$ il est montré suffisant et nécessaire que les sous-problèmes aient des solutions ; en d'autres termes que le recours soit relativement complet (cf à la définition 1.3 au paragraphe 1.4.1).

Le rôle du paramètre α est délicat. Une valeur élevée encouragerait la séquence $\{x^k\}$ à converger plus rapidement mais elle empecherait la convergence de la séquence $\{\lambda_t^{i,k}\}$.

3.5 Méthode d'approximation quadratique diagonale (*DQA*)

La méthode de la *couverture progressive* [58] a introduit deux idées intéressantes : la relaxation des contraintes de non-anticipativité et l'utilisation de lagrangiens augmentés dans les modèles multi-étapes.

La méthode d'approximation quadratique diagonale ([50]) s'inscrit dans un cadre général des problèmes qui comportent un certain nombre de contraintes couplantes (contraintes qui couplent certaines variables), mais dans la suite on va considérer l'application de la méthode dans la programmation stochastique. Soit alors le problème suivant :

$$\begin{array}{ll} & \min \sum_s f_s(x_s) \\ \text{sc} & x_s \in \mathcal{X}_s \\ & Qx = 0 \end{array} \tag{3.9}$$

Les contraintes de positivité sont incluses dans le polyhèdre $\mathcal{X}_s$. Le dernier groupe de contraintes représente les contraintes de non-anticipativité, où x représente le vecteur de variables incluant toute variable x_s de tous les scénarios. On forme le lagrangien augmenté en relaxant les dernières contraintes (soit λ le vecteur de multiplicateurs) :

$$\min_{x^s \in \mathcal{X}^s} \sum_s f^s(x^s) + \lambda^T Q x + \tfrac{1}{2}\rho \|Qx\|^2 \tag{3.10}$$

ρ étant un terme de pénalité choisi par l'utilisateur et changé éventuellement au cours de l'algorithme. Observons que le terme $\|Qx\|^2$ du programme (3.10) détruit la décomposabilité par scénario, en introduisant des produits entre les variables non-anticipatives appartenant à différents scénarios.

Pour mettre en clair les idées principales de la méthode, on choisit d'étudier un exemple. Supposons que les scénarios sont disposés sur un arbre comme montré dans la figure 7.

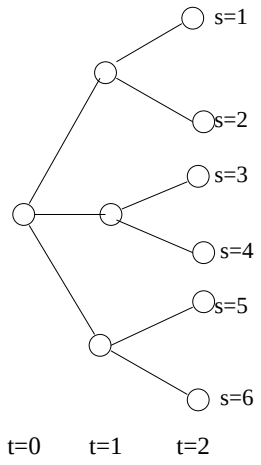


Fig. 7: Un exemple d'arbre de scénarios pour l'explication de la méthode DQA.

La variable x_t^s correspond à la décision prise au moment t suivant le scénario s. En observant le flux d'information comme celui-ci est affiché sur la figure 7, on introduit les contraintes de non-anticipativité suivantes :

$$\begin{array}{c} \text{Période 0} \\ \hline x_0^1 = x_0^2 \\ x_0^2 = x_0^3 \\ x_0^3 = x_0^4 \\ x_0^4 = x_0^5 \\ x_0^5 = x_0^6 \end{array} \qquad \begin{array}{c} \text{Période 1} \\ \hline x_1^1 = x_1^2 \\ x_1^3 = x_1^4 \\ x_1^5 = x_1^6 \end{array} \tag{3.11}$$

On écrit la fonction duale pour cet exemple (fonction objectif du programme (3.10)) :

$$\begin{aligned}\Lambda(x,\lambda) = &\sum_{s,t} c_t^s x_t^s \\ &+ \sum_{s=0}^{s=5}(x_0^s - x_0^{s+1})\lambda_0^s + (x_1^1 - x_0^2)\lambda_1^1 + (x_1^3 - x_0^4)\lambda_1^2 + (x_1^5 - x_0^6)\lambda_1^3 \\ &+ \frac{1}{2}\rho[\sum_{s=1}^{s=5}(x_0^s - x_0^{s+1})^2 + (x_1^1 - x_1^2)^2 + (x_1^3 - x_1^4)^2 + (x_1^5 - x_1^6)^2]\end{aligned} \tag{3.12}$$

Il est clair que cette fonction n'est pas décomposable par scénarios. On s'occupera de cet aspect ultérieurement. Notre préoccupation pour le moment est la minimisation de la fonction (3.12), qui est une fonction quadratique faisant intervenir des variables de différents scénarios. Si on dispose d'un vecteur de multiplicateurs λ, on peut résoudre la (3.12), malgré le fait qu'un tel calcul peut être fastidieux (surtout dans le cas où on a beaucoup de scénarios).

L'algorithme pour la recherche de multiplicateurs est le suivant :

ALGORITHME SUR LE CALCUL DE MULTIPLICATEURS

Étape 0. Poser k=0. On suppose disposer d'un vecteur de pénalités λ^k, au départ ce vecteur peut être nul.

Étape 1. Résoudre le programme suivant :

$$\min \Lambda(x,\lambda) \quad \text{(voir fonction (3.12))} \quad \text{s.c} \quad x \in \mathcal{X} \tag{3.13}$$

Soit $x^k = (x_1^k, x_2^k, ..., x_S^k)$ la solution obtenue.

Étape 2. Si les contraintes de non-anticipativité sont vérifiées(concrètement Qx^k=0), alors arrêter. Sinon, poser :

$$\lambda^{k+1} = \lambda^k + r(Qx^k)$$

Mettre à jour le compteur des itérations : $k \leftarrow k+1$. Retourner à l'Étape 1.

La solution x^k obtenue à l'issue de l'Étape 1 est le vecteur de solution pour tous

les scénarios s=1, ..., S à l'itération k. Chaque composant x_s est à son tour un vecteur $(x_1, ...x_T)$ pour tous les instants t=1, ..., T où T est le nombre de périodes d'étude.

L'écriture Qx=0 constitue une manière condensée pour exprimer les contraintes de non-anticipativité. Qx est donc d'un vecteur de type $(x_i - x_j, x_j - x_k, ...)$ avec autant de lignes que les contraintes de non-anticipativité (la dimension du vecteur λ).

On se focalise maintenant sur l'Étape 1 de l'algorithme. On souhaite résoudre le programme (3.13). Le programme dans une telle forme n'est pas séparable par scénarios. Ce qui rend la décomposabilité impossible, à ce stade, est l'existence des produits scalaires entre les variables de différents scénarios $x_i x_j$. Par exemple, à l'instant t=1 les scénarios s=1 et s=2 se couplent. Si on essaye de décomposer la (3.12) par scénarios on aura pour le scénario s=1 :

$$\Lambda_1 \quad = \quad \sum_t c_t^1 x_t^1 + \sum_t \lambda_t^1 x_t^1 + \tfrac{1}{2}\rho (x_1^1)^2 - \rho x_1^1 x_1^2 + \tfrac{1}{2}\rho (x_2^1)^2 - \rho x_2^1 x_2^2 \tag{3.14}$$

où on repère tout de suite le produit couplant $x_2^1 x_2^2$.

L'approximation proposée dans l'algorithme DQA se réfère au produit $x_2^1 x_2^2$ de l'équation(3.14) et elle est la suivante : avec une certaine erreur, on peut remplacer le produit $x_i x_j$ par $x_i \tilde{x}_j + \tilde{x}_i x_j - \tilde{x}_i \tilde{x}_j$. Rappelons que les variables x_i et x_j devraient avoir la même valeur (exigence de non-anticipativité). La valeur $\tilde{x}_i$ est une solution de référence, et plus proche elle est de la valeur que les deux variables x_i, x_j devraient avoir, plus l'erreur est petite. L'erreur de l'approximation est d'ordre $\mathcal{O}(\|x - \bar{x}\|^2)$. Les auteurs cite [69] d'avoir introduit et utilisé pour la première fois cette approximation quadratique.

Avec cette approximation, la fonction duale (3.14) (pour le scénario $s = 1$) se tranforme en une fonction qui ne contient aucun terme provenant de scénario autre que s=1 ; la (3.12) devient désormais décomposable.

$$\Lambda_1 \quad = \quad \sum_t (c_t^1 + \lambda_t^1) x_t^1 + \tfrac{1}{2}\rho (x_1^1)^2 - \rho x_1^1 \tilde{x}_1^1 + \tfrac{1}{2}\rho (x_2^1)^2 - \rho x_2^1 \tilde{x}_1^2 \tag{3.15}$$

où $\tilde{x}_1^1 = \tilde{x}_1^2$, par exemple, est la valeur que prendraient les variables couplées x_1^1 et x_1^2 à la première période si on arrêtait l'algorithme à ce point.

Les approximations $\tilde{x}$ se mettent à jour en appliquant la règle suivante, dont la convergence est montrée si le pas $\tau \in (0,1)$ est suffisamment petit. On supprime les indices pour permettre une notation plus légère.

$$\tilde{x}^{k+1}=\tilde{x}^k + \tau(x^k - \tilde{x}^k) \tag{3.16}$$

Le moment de terminaison de l'algorithme est le moment où les contraintes de non-anticipativité seront satisfaites. Si pourtant on souhaite arrêter avant, le critère de terminaison devient le suivant :

$$\|Q_s(x^k - \tilde{x}^k)\| \leq \varepsilon \quad \forall\, s$$

On donne la description formelle de l'algorithme portant sur le calcul de solutions candidates $\tilde{x}_i$. Cet algorithme est imbriqué dans l'algorithme de calcul de multiplicateurs et il est contenu dans la première étape.

Algorithme sur le calcul de solutions candidates

Étape 0. Poser m=0. On suppose disposer des solutions $\tilde{x}^{k,m}$, ainsi que d'un vecteur de pénalités $\lambda^{k,m}$.

Étape 1.

a. Résoudre (3.15) pour chaque scénario s=1, ..., S afin d'obtenir $x^{k,m}$.

b. Si $\|Q_(x_i^{k,m} - \tilde{x}_i^{k,m})\| \leq \varepsilon$ pour toute contrainte de non-anticipativité i, alors aller à l'Étape 2. Sinon, poser

$$\tilde{x}^{k,m+1} = \tilde{x}^{k,m} + \tau(x^{k,m} - \tilde{x}^{k,m})$$

Poser $m \leftarrow m+1$. Retourner à l'Étape 1b.

Étape 2. Calculer la politique suivante, $x^{k,m+1}$, en résolvant le programme (3.8).

La structure de la matrice Q, comportant les contraintes de non-anticipativité, est affichée dans le tableau 8 pour le cas de l'exemple (3.12) (cf à la figure 7). Cette

s=1		s=2		s=3		s=4		s=5		s=6	
1		-1									
		1		-1							
				1		-1					
						1		-1			
								1		-1	
	1		-1								
					1		-1				
									1		-1

Tab. 8: Matrice des contraintes de *non-anticipativité* pour l'exemple de la figure 7.

matrice se trouve désormais dans la fonction objectif de façon décomposée (cf à (3.10)). Il est important d'étudier la forme que prend la Hessienne pour chaque scénario. On remarque que les colonnes de ce tableau pour chaque scénario sont orthogonales. Pour l'exemple que l'on est en train de traiter, les Hessiennes pour chaque scénario sont les suivantes :

$$H_1 = H_6 = \rho \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad H_2 = H_3 = H_4 = H_5 = \rho \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$$

En général, la Hessienne pour chaque scénario est toujours donnée par $H_i{=}\rho Q_i^T Q_i$ et elle est une matrice diagonale dont les élements diagonaux sont 0, 1 ou 2 avec les 0 correspondant aux variables du dernier niveau (qui n'entrent pas dans les contraintes de non-anticipativité). Cette structure facilite beaucoup l'utilisation des méthodes de points intérieurs.

3.6 Décomposition avec lagrangien augmenté

Ruszczyński [62] a étudié le problème de contraintes couplantes dans sa généralité et il a proposé une décomposition avec lagrangien augmenté. Il a constaté que la com-

plexité du problème est fortement liée au nombre de sous-matrices qui interviennent dans le couplage. Dans l'exemple de la figure 9, où les régions hachurées dans chaque sous-matrice représentent des éléments non-nuls, le nombre maximum de voisins est de 1, puisque au plus deux blocs de sous-matrices sont impliqués dans une contrainte donnée. Pour expliquer brièvement l'idée, on va considérer le problème général suivant. L désigne le nombre de blocs (sous-matrices) différents.

$$\begin{aligned} &\min \sum_{i=1}^{L} f_i(x_i) \\ &\sum_{i=1}^{L} A_i x_i = b \\ &x_i \in \mathcal{X}_i, \quad i{=}1, 2, \ldots, L \end{aligned} \tag{3.17}$$

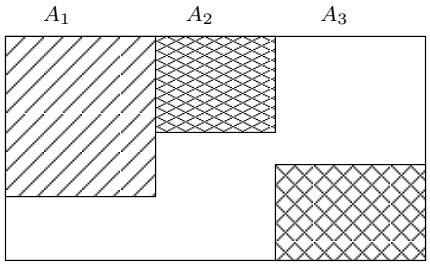


Fig. 9: Sous-matrices adjacentes. Il n'y a au plus que deux sous-matrices qui se couplent à la fois, une structure souvent rencontrée aux problèmes multi-étapes avec recours.

Le lagrangien augmenté pour un seul bloc i de contraintes est le suivant :

$$\begin{aligned} \Lambda_i(x,\lambda) =& \sum_{i=1}^{L} f_i(x_i) + \langle \lambda, b - \sum_{i=1}^{L} A_i x_i \rangle + \frac{1}{2}\,\rho \left\| b - \sum_{i=1}^{L} A_i x_i \right\|^2 \\ =& \langle b, \lambda \rangle + \sum_{i=1}^{L} (f_i(x_i) - \langle A_i^T \lambda, x_i \rangle) + \frac{1}{2}\,\rho \left\| b - A_i x_i + \sum_{\substack{j=1 \\ j \neq i}}^{L} A_j x_j \right\|^2 \end{aligned} \tag{3.18}$$

où ρ est un paramètre de pénalité. Le dual se forme ainsi :

$$\max_{\lambda \in \mathbb{R}^m} \inf_{x_i \in \mathcal{X}} \Lambda(x,\lambda), \quad \mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_L \tag{3.19}$$

Le problème (3.19) n'est pas séparable par bloc. Remarquons que le term $\sum_{j\neq i} A_j x_j$ est celui qui fait intevenir des variables x_i et x_j des blocs différents et il constitue par conséquent la source du couplage. Ruszczyński propose de remplacer ce terme par une estimation $\tilde{x}$ qui sera améliorée à chaque itération. Cette forme permet ainsi de résoudre les problèmes 3.20 en parallèle, chacun correspondant à la sous-matrice A_i. Cette méthode est appelée *algorithme jacobien non-linéaire*.

$$\min_{x\in\mathcal{X}_i} f_i(x_i) + \langle \lambda, b - A_i x_i \rangle + \frac{1}{2}\,\rho \left\| b - A_i x_i - \sum_{\substack{j=1 \\ j\neq i}}^{L} A_j \tilde{x}_j \right\|^2 \tag{3.20}$$

La stratégie de résolution du problème initial (3.17) est la suivante. Pour un vecteur λ fixé, on résout chaque sous-problème (3.18) pour récupérer la solution x_i ; ensuite, on génère un autre multiplicateur λ suivant la formule (3.21) et on répète la procédure dite *méthode de multiplicateurs*. Par ailleurs, la méthode employée pour le calcul des nouveaux $\tilde{x}_j$ est similaire et suit la formule (3.22). La méthode Jacobienne constitue une partie de la méthode de multiplicateurs. Il existe alors dans la méthode deux compteurs d'itérations : le r correspond aux itérations de la méthode Jacobienne et le k s'incrémente chaque fois qu'un nouveau multiplicateur se génère. Les deux méthodes (Jacobienne et de multiplicateurs) exploitent le fait que la différence algébrique de deux variables qui devraient être égales constitue un sous-gradient du problème, et dans ce sens sont similaires.

$$\lambda^{k+1} = \lambda^k + \rho(b - Ax^k) \tag{3.21}$$

$$x^{k+1} = x^k + \tau(x_i^{k,r} - \tilde{x}_i^{k,r}) \tag{3.22}$$

Le critère de terminaison de la méthode Jacobienne est la condition $A_i x_i^{k,s} = A_i \tilde{x}^{k,s}$ pour tout sous-problème i. Il se trouve que le choix du coefficient τ est déterminant pour le bon déroulement de l'algorithme. Ruszczyński montre que sa valeur doit être contenue dans l'interval $0 \leq \tau \leq N^{-1}(1-\mu)$, où N est le nombre maximal de blocs couplants dans la matrice de contraintes (N=1 dans l'exemple de la figure 9) et $0 \leq \mu < 1$ est

un paramètre de l'algorithme. Dans ce cas, la décomposition lagrangienne augmentée converge et la convergence est finie. La preuve et la vitesse de convergence sont données dans [62]. Le critère de terminaison de la méthode de multiplicateurs est la condition que la solution issue des sous-problèmes x^k satisfasse les contraintes $Ax^k{=}b$.

Deux ans plus tard Ruszczyński [63] applique la méthode générique décrite ci-dessus (utilisée pour des problèmes où des couplages de sous-matrices apparaîssent dans la matrice de contraintes) sur les problèmes modélisés en multi-étapes. Il effectue deux types de décomposition : par scénarios et par nœuds. Il traite alors les modèles de deux manières, comme celles-ci ont été décrites au paragraphe 3.1.

D'une part, dans la décomposition par scénarios les contraintes de non-anticipativité se ramènent dans la fonction objectif, augmentée par un terme quadratique multiplié par une pénalité ρ suivant le cadre général du lagrangien augmenté. La démarche suivie est très proche du travail original de Rockafellar et Wets [58]. La différence se trouve dans la façon dont les solutions candidates $\tilde{x}$ se calculent. On rappelle que dans [58] le nouveau $\tilde{x}$ est la somme pondérée (une sorte de moyenne) de toute solution de variables appartenant dans le même faisceau de scénario à un instant donné, alors que dans [63] les nouveaux $\tilde{x}$ se calculent suivant la méthode jacobienne non-linéaire.

D'autre part, la décomposition par nœuds s'effectue sur la base du problème (3.1) (cf à page 55). Les expressions couplantes $T^i x^{\alpha(i)} + W^i x^i - h^i$ multipliées par le vecteur des multiplicateurs lagrangiens, sont integrées dans la fonction augmentée en obtenant ainsi le problème (3.23).

$$\begin{aligned}\Lambda(x,\lambda) = \sum_{i\in\mathcal{N}} p^i f^i(x^i) + \sum_{i\in\mathcal{N}} p^i \langle \lambda^i, h^i - T^i x^{\alpha(i)} - W^i x^i \rangle \\ + \frac{1}{2}\rho \sum_{i\in\mathcal{N}} p^i \left\| h^i - T^i x^{\alpha(i)} - W^i x^i \right\|^2\end{aligned} \tag{3.23}$$

où on a inclus le premier nœud $i{=}1$ dans l'ensemble $\mathcal{N}$ dont la probabilité d'occurence p^1 est de 1. On décompose le (3.23) en sous-problèmes (3.24).

$$\begin{aligned}\Lambda(x^i,\tilde{x},\lambda) &= f^i(x^i) - \langle T_i^T\lambda^i + \sum_{s\in\mathcal{S}^(i)} p^{s|i}T_s^T\lambda^s,\, x^i\rangle \\ &\quad + \frac{1}{2}\rho\left\|h^i - T^i\tilde{x}^{\alpha(i)} - W^i x^i\right\|^2 \\ &\quad + \frac{1}{2}\rho\sum_{s\in\mathcal{S}^(i)} p^{s|i}\left\|h^s - T^s x^i - W^s\tilde{x}^s\right\|^2\end{aligned} \tag{3.24}$$

Le choix des λ à chaque itération se fait comme décrit auparavant, alors que la mise à jour de l'estimation $\tilde{x}$ s'effectue comme dans la méthode DQA (cf au paragraphe 3.5).

Les points forts de la méthode semblent être la simplicité de la mise à jour des multiplicateurs et la convergence dans les modèles multi-étapes (deux blocs impliqués à chaque couplage) qui est améliorée par rapport à d'autres problèmes où la densité de contraintes couplées est supérieure.

3.7 Recours itéré

La difficulté d'une résolution exacte de grands problèmes multi-étapes a incité Balasubramanian et Grossmann [3] à proposer une approximation passant par le problème en deux-étapes.

On décrit brièvement la méthode : on est au moment t=0 et on relaxe toutes les contraintes de non-anticipativité à partir de l' instant t=1 jusqu'à la fin de l'horizon d'étude. On résout ainsi un problème en deux étapes. La première étape est le nœud n=0 et la seconde étape est tous les scénarios découplés à partir de ce nœud là. On garde la solution du premier niveau.

On passe ainsi à la période t=1. À cette période là, on visite tous les nœuds un par un. Pour chaque nœud n se trouvant en période t=1, on a un nouveau sous-arbre dont la racine est le nœud courant n. On relaxe encore la non-anticipativité en résolvant à nouveau un problème en deux étapes. On continue ainsi jusqu'au moment où on sera arrivé à la période $T-1$.

Une partie de la méthode est présentée dans la figure 10. L'arc dans la figure pointe vers la partie de l'arbre qui sera *"parallélisée"* i.e la partie dont les contraintes

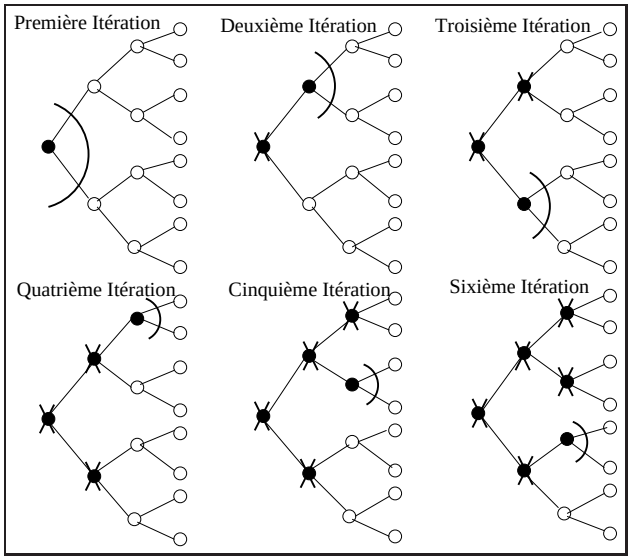


Fig. 10: Méthode d'approximation des modèles multi-étapes par les modèles en deux étapes. Déroulement de la méthode dans un problème à 4 périodes. L'arc à chaque nœud signifie la relaxation des contraintes de non-anticipativité à partir de ce point jusqu'aux feuilles de l'arbre.

de non-anticipativité seront relaxées. Cela revient à dire que tous les scénarios sont découplés, en créant autant de variables par nœuds que les scénarios tranversant le nœud courant. Chaque fois que l'on parcourt un nœud, on le barre de la liste ce qui signifie que la complexité de la méthode augmente linéairement avec le nombre de nœuds. Il ne faut pourtant pas oublier que la méthode de recours en deux étapes peut déjà demander un temps de calcul important si les scénarios sont nombreux. En tout cas le temps de calcul reste largement en dessous du temps de calcul qui aurait été nécessaire pour résoudre le problème multi-étapes dans sa totalité.

Il faudrait remarquer que la méthode proposée calcule les variables du premier niveau lors de la première itération et il n'y plus moyen d'y remedier ultérieurement. L'algorithme continue avec les autres nœuds mais ceci ne met pas en cause les décisions du premier niveau. La seule raison pour laquelle on continue au dela de la première itération est pour mieux évaluer le coût de la décision déjà prise au premier niveau, et

non pas pour améliorer cette décision au cours de l'algorithme.

Ce qui fait d'une approche multi-étapes une approche plus fiable, est la possibilité de revenir implicitement dans le passé et de changer des solutions déjà fixées afin de les rendre plus flexibles vis-à-vis des scénarios potentiellement tordus apparus ultérieurement dans l'horizon d'étude. Ceci n'est pas possible avec une méthode de type recours itéré. En d'autres termes, si on ne s'intéresse qu'à la solution du premier niveau, la méthode du recours itéré revient à être la résolution d'un problème en deux étapes.

Le problème que les auteurs ont résolu a été un problème de programmation mixte. Comme le nombre de variables de recours a été important, les auteurs ont préféré une méthode approximée plutôt que d'adapter une méthode existante dans la programmation stochastique linéaire qu'elle puisse aborder la non-convexité (éventuellement une méthode branch & bound). Les résultats qu'ils présentent comparent différentes approches, dont une est la méthode déterministe avec un horizon glissant. Au lieu d'approximer à chaque nœud le problème par un modèle en deux étapes, on l'approxime par un modèle déterministe (où la demande par exemple est donnée en moyenne, et non pas à travers de différents scénarios). L'autre approche était l'approche exacte c'est-à-dire le modèle réel multi-étapes. Les résultats portent sur un problème en six étapes et 3^5=243 scénarios. Le traitement du problème en tant que multi-étapes a demandé un temps de calcul qui s'est approché de 20 heures. La version de la méthode avec le recours a tourné en une vingtaine de secondes, alors que sa variante en déterministe a eu juste besoin de 3 secondes. Cépendant, les deux méthodes approchées n'étaient pas loin de la solution optimale donnée par le modèle multi-étapes. Malheureusement, cela n'est pas un résultat qui puisse être généralisé, puisque il dépend fortement du problème particulier à résoudre.

Dans le problème étudié par les auteurs l'aléa apparaissait dans le second membre, alors que la fonction objectif était déterministe. Cela revient à dire que s'il existe une solution au premier niveau qui est rélisable pour tous les sous-problèmes du second niveau, alors toutes les solutions du premier niveau sont réalisables pour les problèmes de la deuxième étape. Cela est vrai puisque les polyhèdres des sous-problèmes sont tous identiques et ne dépendent pas donc de l'aléa. On fait toutefois remarquer que

la méthode de recours itéré n'est applicable que si l'hypothèse de recours complet est vérifiée.

3.8 Autres méthodes de décomposition

Dans la plupart des méthodes de décomposition, l'optimisation se fait en considérant l'ensemble de scénarios, en rendant ainsi la taille des problèmes très grand. La décomposition imbriquée, comme méthode de référence dans les modèles multi-étapes, résout à chaque étape tous les sous-problèmes correspondant à chaque nœud de l'arbre. Le résultat est que les coupes produites par la méthode sont plus serrées, plus exactes mais le temps pour les produire augmente considérablement avec la taille de l'arbre . D'autres méthodes échantillonent l'espace de scénarios, comme par exemple la décomposition stochastique de Higle et Sen [35]. Elles sont plus performantes au niveau du temps de calcul mais la qualité de coupes n'est pas aussi bonne, puisque il est probable d'avoir des coupes non valides.

Chen et Powell [26] ont proposé la méthode dite *l'algorithme des plans coupants et de l'échantillonage partiel* (CUPPS) qui se positionne au milieu de la décomposition imbriquée et de la décomposition stochastique. La forte hypothèse d'indépendance des variables aléatoires à travers les étapes se fait ce qui simplifie considérablement le modèle (cf à l'annexe A, page 167). Les coupes sont valides, mais pas exactes, et par conséquent pas suffisamment serrées (cf à la figure 11). Pourtant le temps de calcul est nettement amélioré par rapport à la méthode de décomposition imbriquée.

Ce qui réduit le temps de calcul est que dans la méthode CUPPS un seul nœud par étape est visité, alors que dans la décomposition imbriquée chaque nœud à chaque étape est visité. Brièvement, l'algorithme démarre de l'étape 0 et le problème initial est résolu :

$$\begin{aligned} &\min_{x_1 \in \mathcal{X}} c^T x_1 + z_1 \\ &z_1 \geq -M \end{aligned} \tag{3.25}$$

Dans la première itération, comme le coût approximé de la fonction de recours est inconnu, il est remplacé par un grand nombre négatif d'où l'inégalité $z_1 \geq -M, M > 0$.

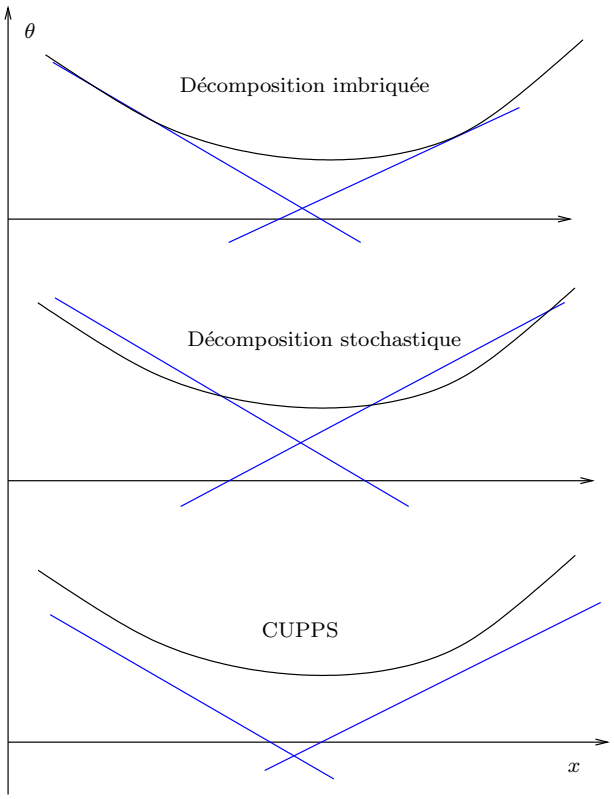


Fig. 11: Comparaison de la qualité des coupes d'optimalité générées dans chacune des trois méthodes. Décomposition imbriquée : coupes serrées, mais temps de calcul important. Décomposition stochastique : coupes non-exactes, mais très rapide à générer. CUPPS : coupes valides, non-serrées, mais rapide à générer.

La solution obtenue x_1 va passer au problème de l'étape suivante. On est toujours dans la première itération, où à chaque étape on tire aléatoirement une réalisation de la variable aléatoire parmi celles qui nous sont proposées et on résout le programme suivant pour une réalisation ξ et une solution x_{t-1} du programme qui correspond à l'étape précédente :

$$\begin{aligned} &\min_{x_t \in \mathcal{X}_t} c^T x_t + z_t \\ &Wx = h(\xi) - Tx_{t-1} \\ &z_t \geq -M \end{aligned} \tag{3.26}$$

Après la résolution de ce problème, la solution x_t passe au problème suivant ; des coupes (3.27) sont génerées pour être rajoutées au problème du nœud-père.

$$a_{t,k+1} + g_{t,k+1}^T x_{t-1} \leq z_t \tag{3.27}$$

où z_t est le maximum d'une famille de fonctions affines qui sont les hyperplans d'appui de la fonction de recours, comme il a déjà été expliqué.

À toute itération k et pour chaque étape t on résout le programme suivant :

$$\begin{aligned} &\min_{x_t \in \mathcal{X}_t} c^T x_t + z_t \\ &Wx = h(\xi) - Tx_{t-1} \\ &a_{t,k+1} + g_{t,k+1}^T x_{t-1} \leq z_t \end{aligned} \tag{3.28}$$

Donc à chaque itération, on parcourt une par une l'ensemble d'étapes t=1, ..., T, on utilise la solution de l'étape $t-1$ qui représente une constante au second membre du problème à l'étape t et on génère une coupe d'optimalité que l'on passe au problème précédent. Lorsque aucune coupe n'est produite (car l'estimation z_t coïncidera avec la fonction réelle de coût) pour tout instant t, on aura terminé l'algorithme.

Les trois conditions suivantes sont imposées pour la convergence de la méthode CUPPS. La convergence à la limite de l'algorithme est montrée mais elle n'est pas finie :

- L'espace de scénarios (réalisations de la variable aléatoire) à chaque étape est fini et discret et les réalisations sont indépendantes.
- On fait l'hypothèse du recours rélativement complet.
- Les aléas n'infuencent que le second membre (toutes les matrices intervenant dans les contraintes et dans la fonction objectif ne contiennent pas d'aléas).

3.9 Méthodes de points intérieurs

La difficulté de résolution des problèmes multi-étapes est essentiellement due à la taille des modèles qui augmente avec le nombre de étapes de façon exponentielle. Dans les méthodes décrites précédemment une décomposabilité (par nœud ou par scénarios) est possible, mais les itérations nécessaires pour une solution suffisamment proche d'une solution implémentable (qui satisfasse les contraintes de non-anticipativité par exemple) peuvent être nombreuses. Les méthodes de points intérieurs sont de plus en plus utilisées pour deux raisons : en exploitant la structure de la matrice du problème, elles permettent des calculs par blocs et le nombre d'itérations nécessaires pour converger reste relativement faible (typiquement une vingtaine d'itérations suffissent). De plus, comme l'ont observé de nombreux auteurs, le nombre d'itérations jusqu'à convergence ne dépend pratiquement pas de la dimension du programme.

On s'efforcera dans la suite d'esquisser l'idée des méthodes de points intérieurs sans aucune ambition d'entrer dans les détails. Considérons un programme linéaire qui se formule sous forme standard comme suit :

$$(P) \quad \begin{array}{c} \min c^T x \\ \text{s.c } Ax = b \\ x \geq 0 \end{array} \tag{3.29}$$

On construit son dual qui après avoir rajouté les variables d'écart s, s'écrit :

$$(D) \quad \begin{array}{c} \max b^T y \\ \text{s.c } A^T y + s = c \\ s \geq 0 \\ y \in \mathbb{R}^n \end{array} \tag{3.30}$$

La première étape des méthodes de points intérieurs du type *"primal-dual"* consiste à écrire les conditions d'optimalité. On cherche alors des vecteurs x, y et s de dimensions appropriées pour que les conditions suivantes soient vérifiées :

$$\begin{array}{ll} Ax = b & \text{(réalisabilité du primal)} \\ A^T y + s = c & \text{(réalisabilité du dual)} \\ b^T y - c^T x = 0 & \text{(pas de saut de dualité)} \\ x \geq 0, \quad s \geq 0 & \end{array} \tag{3.31}$$

Dans cette formulation, un algorithme de points intérieurs efficace devrait exploiter au maximum à la fois l'information primale et duale. Le système homogène (3.32) doit être résolu :

$$\begin{array}{l} Ax - b\tau = 0 \\ A^T y + s - c\tau = 0 \\ b^T y - c^T x + k = 0 \\ x \geq 0, s \geq 0, t \geq 0, k \geq 0 \end{array} \tag{3.32}$$

Une solution de (3.32) qui aura $\tau^* > 0$ ainsi que k^*=0 sera une solution optimale du problème initial. Le vecteur optimal sera $(\frac{x^*}{\tau}, \frac{y^*}{\tau}, \frac{s^*}{\tau}, \frac{k^*}{\tau})$. Par ailleurs, une solution de (3.32) satisfera forcément $x^T s + \tau k$=0, puisque il faut à la fois que $x_i s_i$=0 $\forall\, i$ et τk=0.

On entre dans la phase où la méthode de Newton est utilisée pour trouver une direction d'amélioration. En général, si $f_1(x, y, s, \tau)$ désigne le membre de gauche de la première équation du système (3.32), $f_2(x, y, s, \tau)$ la suivante et $f_3((x, y, s, \tau)$ la dernière, le rôle de la méthode de Newton est de déterminer une direction (dite aussi : *correction*) $(\delta_x, \delta_y, \delta_s, \delta_\tau)$ au vecteur (x, y, s, τ) de tel sorte que $f_i(x^0 + \delta_x, y^0 + \delta_y, s^0 + \delta_s, \tau^0 + \delta_\tau)$=0, $\quad i$=1, 2, 3. Pour estimer cette direction, on procède à un développement de Taylor d'ordre 1 en toutes les variables pour chacune des fonctions et on obtient donc $f_i + \frac{\partial f_i^0}{\partial x}\delta_x + \frac{\partial f_i^0}{\partial y}\delta_y + \frac{\partial f_i^0}{\partial s}\delta_s + \frac{\partial f_i^0}{\partial \tau}\delta_\tau$=0. Après avoir obtenu une direction issue du problème (3.34) on se déplace dans cette direction en choisissant un pas α.

$$
\begin{aligned}
y^{'} &= y + \alpha\delta_y > 0 \\
x^{'} &= x + \alpha\delta_x > 0 \\
s^{'} &= s + \alpha\delta_s > 0 \\
\tau^{'} &= \tau + \alpha\delta_\tau > 0 \\
k^{'} &= k + \alpha\delta_k > 0
\end{aligned}
\tag{3.33}
$$

et on s'arrête à l'itération où un certain critère de précision sera satisfait. Un des sujets importants pour la vitesse de l'algorithme est le choix du pas de déplacement α.

Indépendamment de la méthode de points intérieurs choisie, le problème de recherche de la direction de Newton passe par la résolution d'un système du type ADA^T=B où D est une matrice diagonale qui dépend de la variante de points intérieurs (primal, dual, primal-dual etc). Pour la résolution d'un tel système il est souvent préférable d'utiliser la factorisation Cholesky. La forme donc de la matrice ADA^T joue un rôle capital sur le temps de calcul. Modéliser un problème stochastique multi-étapes de telle manière à ce que sa structure résulte à une factorisation rapide constitue un défi. Lustig et coauteurs [47] montrent comment une bonne modélisation de contraintes de non-anticipativité peut favoriser la factorisation, contrairement à la formulation standard. Les figures 12 et 13 montrent la différence dans les deux modélisations employés dans [47].

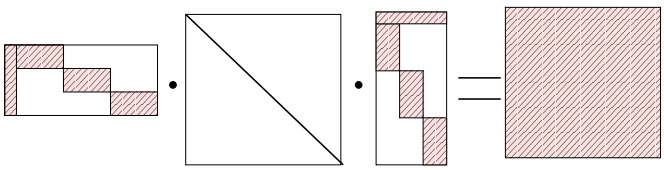


Fig. 12: Multiplication des matrices. De gauche à droite : A, D, A^T, ADA^T. La matrice résultante étant dense, la factorisation Cholesky sera exigeante en termes de temps de calcul.

Xu et Hung [79] ont utilisé un autre système linéaire pour la recherche de la correction $(\delta_x, \delta_y, \delta_s, \delta_\tau)$ dans le cadre la leur méthode dite *"homogeneous self-dual embedding"*. À partir d'une solution *"intérieure"* $(x > 0, s > 0, \tau > 0, k > 0)$, on cherche

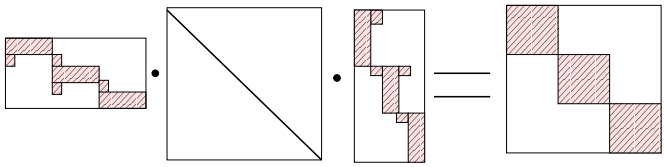


Fig. 13: Multiplication des matrices. De gauche à droite : A, D, A^T, ADA^T. La matrice résultante étant creuse, la factorisation Cholesky sera rapide.

une direction δ dans le cadre du *chemin central analytique*. Ce problème se formule comme suit :

$$\begin{aligned} A\delta_x - b\delta_\tau &= \eta(\tau b - Ax) \\ A^T\delta_y + \delta_s - \delta_c\tau &= -\eta(\tau c - A^T y - s) \\ b^T\delta_y - \delta_c^T x + \delta_k &= \eta(k + c^T x - b^T y) \\ S\delta_x + X\delta_s &= \gamma\mu e - Xs \\ k\delta_\tau + \tau\delta_k &= \gamma\mu - \tau k \end{aligned} \tag{3.34}$$

où η et γ sont deux paramètres dans l'intervalle $[0, 1]$ et μ est donné par $\mu = (x^T s + \tau k)/(n + 1)$ où n est le nœud courant. X et S sont les matrices qui n'ont que de x et de s dans leur diagonale respectivement. En posant η=1 et γ=on obtient la méthode de Newton normale.

Dans [14] les auteurs ont décomposé le problème de la recherche de direction en exploitant la faible densité de la matrice de contraintes. La méthode est adaptée pour des problèmes en deux étapes, mais pas aux problèmes avec des couplages de blocs de contraintes.

Sur les algorithmes de ce type, les travaux [13] ainsi que [14] sont parmi les plus représentatifs. D'autres algorithmes de points intérieurs se trouvent dans [23] ainsi que des tentatives de parallélisation pour les problèmes multi-étapes avec des résultats très prometteurs dans [22].

3.10 Généralités sur les approches par échantillonnage

Aborder un problème multi-étapes issu d'une application réelle a toujours été d'une difficulté considérable. La discrétisation fine des distributions de probabilité des paramètres aléatoires, conduit à beaucoup de scénarios, donc à des arbres de grandes tailles. On décide de conclure ce chapitre avec une brève discussion sur l'échantillonnage mis en œuvre sur les problèmes multi-étapes.

Une approche souvent employée est l'échantillonnage Monte Carlo. La distribution réelle du paramètre aléatoire considéré est échantillonnée et ce qui en résulte est un problème d'une taille plus abordable dont on cherche la valeur optimale.

Celle-ci est connue comme une approche d'échantillonnage dite *"extérieure"*, puisque une fois l'échantillonnage obtenu, le problème devient un problème déterministe de taille moyenne que l'on résout avec un algorithme de notre choix. L'approche est connue comme *approximation moyenne de l'échantillonnage (SAA)* et elle a été présentée dans [42]. Des résultats numériques issus de différents problèmes stochastiques, qui comportent également des variables entières, sont contenus dans [74]. Les problèmes traités ont été des modèles en deux étapes et la méthode de résolution a été la méthode "L-Shaped".

Il existe également des approches dites *intérieures* qui effectuent un processus d'échantillonnage au sein de la méthode employée. Par exemple la méthode de la décomposition stochastique de Higle et Sen [36] est une approche qui emploie l'échantillonnage intérieur au sein de la méthode "L-Shaped" (cf au paragraphe 2.4). La méthode CUPPS décrite précédemment l'est aussi.

Pour les problèmes de recours à deux étapes l'étude sur la validité des bornes obtenues est bien développée en particulier par Shapiro et des résultats indiquant le nombre de scénarios qu'il faut générer pour un niveau de confiance donné sont bien établis (par exemple [64]). Par exemple il est montré que pour un problème en deux étapes comportant un nombre potentiellement infini de scénarios, si on génère un nombre de scénarios *suffisant* et on résout le problème en deux étapes résultant, l'estimateur sur la fonction objectif est valide et consistant[5].

[5]Rappel : Un estimateur est dit consistant si le biais et la variance tendent vers 0 quand le nombre

Dans le domaine des modèles multi-étapes on dispose de très peu de résultats de ce type, car on ne peut pas simplement étendre l'analyse faite pour le cas de deux étapes. Si on tire aléatoirement des scénarios, l'ensemble de scénarios échantillonnés ne forme pas forcément un arbre qui branche à chaque étape ; ceci ne capte pas toute la difficulté d'un problème multi-étapes. La méthode SAA donne toujours une borne inférieure valide, mais souvent cette borne est très loin de l'optimum, elle n'est pas nécessairement consistante (cf [64], paragraphe 6.1).

Pour avoir une borne inférieure valide et consistante sur les problèmes multi-étapes, il faut effectuer un échantillonnage dit *conditionnel*. On procède ainsi : à l'instant $t = 1$ on génère N_1 tirages de la variable aléatoire, soit ξ_2^i, $i = 1, ..., N_1$. On a ainsi N_1 réalisations du vecteur aléatoire ξ_2 à la deuxième période ($t = 2$). Ensuite, pour toute réalisation $i \in \{1, ..., N_1\}$ on génère un autre échantillon ξ_3^{ij}, $j \in \{1, ..., N_2\}$ de la distribution conditionnelle de ξ_3 de l'étape-3 étant donné la réalisation ξ_2^i. On continue ainsi aux étapes ultérieures.

En appliquant l'échantillonnage conditionnel, on obtient une borne inférieure valide et consistante (cf également à [64], paragraphe 6.2). Par ailleurs, pour obtenir une borne supérieure statistiquement valide, il faudra calculer une politique réalisable, c'est-à-dire résoudre un problème multi-étapes. C'est sur cette politique réalisable qu'une borne supérieure sera calculée. On ne dispose pas actuellement de la théorie permettant de déterminer à priori des tailles d'échantillons qu'il faut avoir pour une précision donnée. Plus de détails concernant des résultats théoriques sur l'échantillonnage pour les modèles multi-étapes sont contenus dans [68].

3.11 Conclusions

On a étudié les principales méthodes de résolution des problèmes multi-étapes. Les problèmes qui nous ont intéressés comportent des variables aléatoires discrétisées ou échantillonnées d'une variable continue. Le flux d'information se modélise à travers un arbre de scénarios. La catégorisation de méthodes traitant des modèles multi-étapes change d'un auteur à l'autre. En général ou pourrait distinguer les méthodes directes

d'observations tend vers l'infini.

qui agissent sur la structure de la matrice du grand problème déterministe (cf au pararaphe 3.9) et des méthodes de décomposition.

Les modèles multi-étapes avec discrétisation de la variables aléatoire admettent deux modélisations possibles : (i) avec des scénarios parallèles et des contraintes de non-anticipativité explicitement imposées et (ii) avec des scénarios couplés sur un arbre. Suivant la voie de modélisation on applique la méthode appropriée, qui est soit la relaxation des contraintes de non-anticipativité (première modélisation), soit l'approximation de la fonction de recours (fonction *"cost-to-go"*) à chaque nœud de l'arbre (deuxième modélisation).

Or, la fonction de recours est une fonction qui inclut des moyennes et des minimisations (ou maximisations) imbriqués, c'est-à-dire $\min(\mathbb{E}[\min(\mathbb{E}...)])$, où à chaque fois que l'on calcule la moyenne, l'opération est faite sur l'ensemble de cas possibles qui pourrait se réaliser à partir de l'instant courant. Le calcul de la fonction de recours est l'aspect qui complique le plus la procédure et c'est pour cette raison que des méthodes d'approximation de la fonction de recours sont employées, en utilisant des informations duales.

Sen [70] explique et cite brièvement les grands défis de la programmation stochastique (qui reposent principalement sur les modèles multi-étapes) en évoquant en particulier le besoin de couplage des méthodes de décomposition avec des méthodes d'échantillonnage, le but étant toujours de traiter des instances plus grandes. Dans le chapitre 5 nous serons amenés à proposer un tel couplage : une méthode de décomposition sera choisie et une procédure d'échantillonage sera employée au sein de cette méthode. C'est l'approche de l'échantillonnage intérieur comme défini au paragraphe 3.10.

Dans le chapitre 4 on va décrire le problème spécifique qui nous intéressera. Ce problème est un problème de dimensionnement de capacités où un grand nombre de scénarios doit être traité. La taille du problème rend les approches directes impossibles, puisque on ne dispose pas d'assez de mémoire en machine pour faire contenir le modèle mathématique. Dans le chapitre 5 on présentera des nouvelles mises en œuvre de la décomposition imbriquée qui constituera notre méthode de résolution pour le problème de dimensionnement et des résultats seront présentés.

Chapitre 4

Application au dimensionnement de capacités et à la gestion de stocks et de flux stochastiques

4.1 Introduction

Le marché gazier en France est ouvert à la concurrence depuis juillet 2004. EDF, comme tout opérateur qui manifèste un intérêt d'entrer dans le marché, doit souscrire un certain nombre de contrats avec divers acteurs qui l'engagent sur différents horizons de temps que chaque acteur impose. Ces contrats sont les suivants :

- Des contrats d'approvisionnement avec les producteurs qui injectent du gaz dans le réseau.
- Des contrats d'acheminement avec les opérateurs de transport qui acheminent le gaz vers le clients ou le stock.
- Des contrats de stockage avec les opérateurs de stockage pour stocker/déstocker du gaz aux périodes souhaitées.

Pour bien mettre en évidence les aspects structurants du problème, on introduit la notion de la *zone d'équilibrage*. La zone d'équilibrage est une entité d'abord conceptuelle, et en même temps géographique. Elle pourrait être perçue comme un ensemble de

points *d'entrée* et *de sortie* de gaz. Elle est appelée ainsi car l'expéditeur (par exemple EDF) est tenu de s'efforcer de garder l'équilibre entre le gaz qui entre et le gaz qui sort à travers ces points. Toute la description qui suivra aura comme point de départ la zone d'équilibrage.

Le gaz arrive des pays producteurs et entre dans le territoire français par certains points répartis dans le pays. Chaque point est rattaché à une seule zone d'équilibrage. L'approvisionnement en gaz peut se faire par trois moyens que l'on détaillera au paragraphe 4.4 : (a) contrats d'approvisionnement avec les producteurs, (b) achat auprès de marchés gaziers (par exemple à Zeebrugge) (*forward*) et (c) achat au marché spot du jour pour le lendemain (Zeebrugge, points d'échange de gaz).

Ensuite via un contrat d'acheminement on peut acheminer du gaz soit au client final, soit aux points de stockage, soit aux points d'échange du gaz (PEG), soit à une autre zone d'équilibrage. L'opérateur de transport est le propriétaire de la zone d'équilibrage sur laquelle il fonctionne. Les zones d'équilibrage, appartenant à différents opérateurs de transports (GdF Transport, Total), sont interconnectées, donnant la possibilité à l'expéditeur (EDF ou autre) de fournir du gaz provenant d'une zone d'équilibrage à une autre à travers les liaisons entre les zones.

Il est également possible de stocker tout ou partie de la quantité acheminée dans des points de stockage prévus à cet effet. La quantité stockée sera utilisée ultérieurement, lorsque ceci sera jugé nécessaire par EDF.

Le marché gazier constitue un autre moyen d'approvisionnement de gaz, lorsque pour différentes raisons la source principale d'approvisionnement devient très chère à utiliser. C'est aussi au marché spot qu'EDF ou un autre expéditeur, pourrait en vendre des quantités excedentaires.

Le rôle d'EDF dans un tel contexte est un rôle purement contractuel. EDF ne produit pas ni n'achemine de gaz ; EDF n'est pas propriétaire de points de stockage. Son rôle devient donc celui du pilotage des différents acteurs de telle manière à ce que les flux du gaz depuis le point d'entrée jusqu'au client final se fasse tout en respectant les règles contractuelles auxquelles EDF s'est engagé.

Les contrats passés auprès des producteurs sont typiquement des contrats qui portent sur un horizon long qui s'étend de trois à cinq ans. Les contrats d'achemine-

ment avec les opérateurs de transport ont une période de validité plus courte, mensuelle, semestrielle ou annuelle. EDF peut empiler des contrats d'achemeniment, et ainsi réduire le risque d'une mauvaise prévision initiale sur la demande clientèle ou une sous-estimation de son portefeuille de clients. À titre d'exemple, EDF souscrit un contrat portant sur la capacité de livraison de α MWh/jour qui sera valable pendant un an. La *capacité de livraison* correspond à la quantité journalière maximale en MWh qu'EDF peut demander à son opérateur de transport de tranférer aux clients finaux. Trois mois plus tard, on s'aperçoit que la demande clientèle est plus forte que prévu. EDF peut passer un autre contrat de β MWh/jour sur une période par exemple d'un an. Ces β MWh/jour vont se rajouter à α pendant neuf mois et augmenter donc la capacité d'acheminement pendant ladite période de α MWh/jour à $(\alpha+\beta)$ MWh/jour.

EDF est obligé de programmer en rythme quotidien les quantités à livrer chaque jour pour le lendemain et est tenu de faire connaître ce planning au transporteur. Ces quantités doivent respecter les capacités souscrites. Quand on parle d'acheminement, on entend principalement le transfert du gaz d'un point à un autre. Comme on l'a dit auparavant, un point dans la zone d'équilibrage peut correspondre à :

- un point d'entrée, où le gaz entre dans la zone d'équilibrage provenant des producteurs ;
- un point de livraison à destination des clients finaux ;
- un point PEG (point d'échange de gaz) où on échange du gaz avec d'autres opérateurs ;
- un point de stockage où on fait entrer et sortir du gaz.
- un point qui relie la zone avec une autre zone d'équilibrage.

Aux deux derniers points, du gaz est à la fois entré et sorti de la zone d'équilibrage. Du gaz peut être expédié à une autre zone d'équilibrage ; il peut également provenir d'une autre zone. En un PEG (points d'échange du gaz), EDF pourrait vendre à des tiers une partie du gaz qu'elle a achetée ; il est aussi possible qu'EDF s'achète du gaz auprès des PEG.

EDF a le droit de stocker ou déstocker autant de gaz que souhaité, à condition que celà soit conforme avec les capacités souscrites et que cela respecte le profil d'injection et de soutirage prévu dans les contrats. Les profils d'injection et de soutirage, dont

on parlera dans le paragraphe 4.6, sont assez contraignants et jouent donc un rôle important dans le dimensionnement du stock.

L'objectif pour EDF est d'avoir un outil d'aide à la décision lui permettant :

- de passer de bons contrats avec les differents acteurs, et
- étant engagé à respecter ces conrats, d'effectuer un planning sur un horizon (glissant) de N périodes d'étude. Une période d'étude est entendue comme étant un jour, un mois, un semestre ou un an, suivant le pas de temps choisi.

On distingue alors deux phases dans cette procédure : le dimensionnement et l'ordonnancement. Lors de l'ordonnancement, EDF s'efforce de faire la meilleure planification possible, compte tenu les contrats passés sans pouvoir revenir en arrière et les modifier. Deux cas extrêmes devraient être évités :

- Sous-dimensionnement : Il se peut que lors du dimensionnement des capacités, EDF n'ait pas pris en compte la possibilité d'avoir une forte demande, en ayant souscrit des contrats tels que la capacité d'approvisionnement s'avère finalement insuffisante ou la capacité de stockage soit trop basse. Le cas échéant EDF s'obligera soit d'acheter du gaz en utilisant d'autres moyens qui lui couteront beacoup plus cher, soit à faire des déséquilibres et payer donc les pénalités que cela entraînerait.
- Surdimensionnement : En revanche, si les capacités sont surdimensionnées, il est évident qu'EDF devra payer pour des contrats largement au dessus de ses besoins, et donc pour des capacités inutilisées.

Entre ces deux cas extrêmes, il y a une infinité de cas possibles, certains favorables, d'autres moins favorables, pouvant tous se produire. Il est clair qu'il faut trouver les capacités à souscrire conduisant à des coûts et à des risques minimaux.

4.2 Les différentes sources d'aléas

Lors de la modélisation du problème il y aura deux phases distinctes : le dimensionnement et l'ordonnancement. Cependant la seconde partie s'étend sur un horizon de plusieurs périodes et ceci rend le problème dynamique. Comme on le verra par la suite, le problème est également un problème linéaire puisque l'espace de solutions sera

décrite par un polyhèdre, la fonction objectif ne contenant, elle aussi, que des termes linéaires. Le problème appartient enfin à la catégorie des problèmes de la programmation (linéaire) dynamique *stochastique* à cause de la présence des paramètres incertains. Dans la suite on énumerera tous les paramètres du problème soumis aux aléas. Au paragraphe 4.7 du présent chapitre, on extraira une version simplifiée de ce problème, ne prenant en compte qu'une partie des aléas. Il y aura également d'autres aspects qui seront simplifiés lors de la résolution du problème, mais dans l'immédiat on considérera la forme complète du problème et les simplifications vont suivre ultérieurement.

On énumère les paramètres du problème soumis à l'incertitude et tout de suite on les détaillera :

- La demande du client final.
- Les prix d'approvisionnement du gaz.
- Les défaillances qui peuvent se produire soit au niveau des approvisionnements soit au niveau de l'acheminement.

Demande

La demande clientèle est dépendante à la fois de l'aléa climatique et du portefeuille de clients. Le portefuille de clients est proportionnel à la part du marché d'EDF. Pour la plupart des clients, on ne dispose pas de données sur leur consommation quotidienne. Il n'y a que 0.002% de clients qui sont télé-relevés. Pour cette catégorie de clients, on dispose d'un historique de consommations journalières.

Les autres clients qui ne sont pas télé-relevés sont affectés à différents profils suivant leur consommation. Il y a des clients dont la consommation est fortement dépendante de la température et d'autres qui y sont légèrement corrélés (utilisation du gaz pour leur processus industriel). À partir d'une fonction de tranfert on arrive à convertir la température du jour en consommation en gaz pour chaque profil de clients.

Prix d'approvisionnement

Il y trois types de contrats d'approvisionnement différents qui seront détaillés au paragraphe 4.4 : les contrats à long terme, les contrats du marché gazier du type

"*forward*" et les contrats du "*marché spot*". Tous les trois types sont plus ou moins soumis aux aléas.

Défaillance de fournisseurs

Les producteurs de gaz risquent d'être défaillants pendant une certaine période de temps. EDF est censé pouvoir répondre à la demande de ses clients, en utilisant d'autres moyens d'approvisionnement (par exemple le stock).

Il est important à ce stade de noter qu'il est interdit de suspendre la fourniture du gaz vers un client, celui-ci ne pouvant pas changer rapidement la source d'énergie. EDF ne peut donc pas refuser du gaz aux clients, ce qui la forcera à chercher à tout mettre en œuvre pour satisfaire la demande, même si cela inclut des surcoûts considérables.

Les opérateurs de transport peuvent, eux aussi, pour leurs propres raisons (par exemple travaux de maintenance) réduire pendant une période annoncée à l'avance la capacité d'acheminement sur certaines parties du réseau. EDF devra également pouvoir faire face à de telles situations.

4.3 Structure du réseau

Le réseau d'un opérateur de transport est constitué de zones d'équilibrage et de points qui y sont rattachés. Chaque point appartient à une zone d'équilibrage et une seule. Un opérateur de transport est interconnecté à d'autres opérateurs de transport et à des opérateurs de stockage. Les stockages sont représentés par des points de stockage qui regroupent plusieurs sites de stockage physiques et qui sont reliés aux points élémentaires via des points d'interface transport/stockage (pits). Les interconnexions entre opérateurs différents sont représentées par des points " frontières ".

Au sein d'une zone d'équilibrage, on trouve les points suivants :

- Points de livraison : ils permettent de livrer du gaz à des clients finaux.
- Points d'entrée : ils permettent de faire entrer du gaz fourni par les producteurs dans la zone d'équilibrage.

- Points de liaison : ils permettent de faire passer du gaz d'une zone d'équilibrage d'un opérateur à une autre de ses zones d'équilibrage, à condition que ces deux zones soient connectées entre elles.
- Points d'interface transport/stockage (PITS) : ils permettent de faire entrer (resp. sortir) du gaz en provenance (resp. à destination) d'un site de stockage ;
- Point d'échange de gaz (PEG) : ils permettent d'échanger du gaz avec d'autres expéditeurs présents sur la zone d'équilibrage.

Tout ce qui a été décrit ci-dessous est affiché dans la figure1. D'une certaine manière, les différents points constituent une sorte d'interface entre l'opérateur de transport – propriétaire de la zone d'équilibrage – et les autres acteurs participants (producteurs, autres opérateurs, autres expéditeurs).

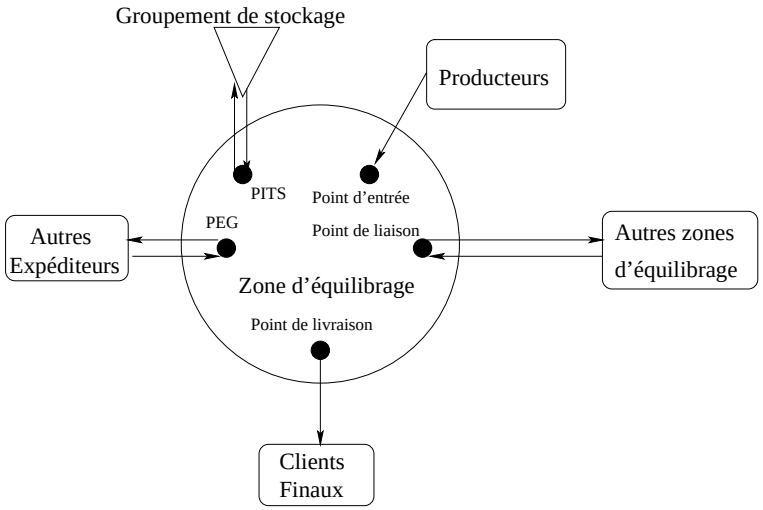


Fig. 1: Structure du réseau gazier.

Afin qu'un expéditeur puisse entrer dans le marché gazier, il aura besoin de passer des contrats avec trois types d'acteurs, comme on l'a dit auparavant : les producteurs pour la fourniture du gaz, les opérateurs de transport pour l'acheminement dans le réseau français, et les opérateurs de stockage pour entreposer du gaz aux périodes souhaitées. Dans la suite on va décrire les spécificités des contrats de chacun de ces acteurs.

4.4 Approvisionnement

Afin de pouvoir s'approvisionner en gaz pour le faire acheminer dans le territoire français, EDF doit passer des contrats avec un ou plusieurs producteurs sur un horizon de temps à définir. Il y en a de trois types qui seront détaillés par la suite : les contrats à long terme, le marché gazier avec des contrats du type forward et les contrats aux prix spot ; parmi les trois, les plus importants sont les premiers.

Contrats à long terme

Parmi les facteurs déterminants d'un contrat d'approvisionnement à long terme on distingue :

- *les points d'entrée* qui sont les points rattachés à une zone d'équilibrage où le producteur d'engage à mettre du gaz à disposition d'EDF.
- *la capacité du gaz réservée sur toute la période d'étude (**PCQ**)* et sa flexibilité. Il s'agit de la quantité maximale en MWh qu'EDF s'engage à acheter auprès du producteur sur la période de contrat. La flexibilité est entendue comme l'intervalle $[PCQ^{min}, PCQ^{max}]$. La quantité qu'EDF utilisera tout au cours de la période de validité du contrat sera bornés dans cet intervalle.
- *la capacité journalière du gaz réservée* et ses flexibilités. La flexibilité est entendue comme l'intervalle entre deux capacités : la capacité journalière minimale et la capacité journalière maximale.

EDF n'est pas tout à fait libre à choisir à la fois PCQ et la DCQ. En effet, à partir de la première, suivant le contrat, on en déduit la seconde. Le facteur de conversion de la PCQ en DCQ est connu sous le nom *nombre de jours*. La relation suivante relie les deux capacités :

$$\mathbf{DCQ} = \frac{\mathbf{PCQ}}{\mathbf{nJ}}$$

où **nJ** représente le *nombre de jours*.

Le facteur **nJ** est proposé par le producteur et ceci fait partie du contrat. À titre d'exemple, en choisissant un contrat qui propose **nJ**=70, alors pour une capacité annuelle souscrite à **PCQ**=700 MWh cela signifierait que l'on se permettrait de faire entrer chaque jour jusqu'à **DCQ**=10 MWh dans le réseau ; si on choisissait **nJ**=350 on

serait autorisé de ne faire entrer qu'au maximum **DCQ**=2 MWh. Un facteur **nJ** plus petit rend le planning plus flexible.

Quant aux coûts d'approvisionnement, EDF doit payer un terme proportionnel à la quantité de gaz (prix de la molécule) que fait entrer le producteur au point d'approvisionnement. Le prix de la molécule est indexé sur plusieurs facteurs, qui ne sont pas directement liés à l'aléa climatique. Le prix du pétrole, le prix du gasoil, le taux d'échange entre l'euro et le dollar sont parmi les coefficients qui déterminent le coût d'approvisionnement pour ce type de contrat.

L'achat est du type *"take or pay"*, à savoir : même si durant la période contractuelle on fait entrer moins que $\mathbf{PCQ}^{min}$, on sera obligé de payer pour toute la capacité minimale ($\mathbf{PCQ}^{min}$) que l'on a souscrite. Il n'est pourtant pas permis de dépasser la capacité maximale $\mathbf{PCQ}^{max}$. Il en va exactement de même pour les quantités journalières **DCQ**.

Contrats de marché gazier et contrats spot

Il est possible d'acheter du gaz dans les marchés gaziers à l'avance de deux manières différentes avec les contrats *"forward"* et avec les contrats *"spot"*.

Dans les contrats *"forward"* on réserve au jour j_0 des capacités pour un jour j_n ultérieur. Dans ce type de contrats on n'a pas de flexibilité et on est obligé de s'approvisionner de toute la quantité souscrite (*ruban*).

D'autre part il y a les contrats spot, où on achète du gaz aujourd'hui pour le lendemain. Les prix spot sont liés à la température. Lorsque l'on passe un contrat spot, on souscrit une certaine capacité que l'on ne se permet pas de dépasser. Il n'y a aucune notion de flexibilité. L'intérêt d'un tel contrat devient évident si le prix sur le marché se trouve aux niveaux favorables pour l'expéditeur.

4.5 Acheminement

Dans un contrat d'acheminement, EDF doit souscrire les capacités suivantes :

- *Capacité journalière d'entrée* qui est la quantité maximale d'énergie que le transporteur s'engage à enlever chaque jour en un point d'entrée donné.
- *Capacité journalière de liaison entre zones*, qui est la quantité maximale d'énergie que le transporteur s'engage à transférer chaque jour d'une zone d'équilibrage à une autre. Entre deux zones d'équilibrage il y deux capacités de liaison, une pour chaque sens.

Chaque jour, EDF doit communiquer au transporteur les trois quantités suivantes dont elle prévoit d'avoir besoin :

i. Quantité à enlever à chaque point d'entrée.
ii. Quantité à livrer à l'ensemble des points de livraison par zone d'équilibrage.
iii. Quantité qu'EDF prévoit de livrer ou de recevoir aux points de liaison de chaque zone d'équilibrage.
iv. Quantité qu'EDF prévoit d'injecter dans le stock et de soutirer depuis le stock.

La tarification de l'acheminement est basée sur (i) l'achat de capacité et (ii) l'équilibrage. L'idée de l'équilibrage porte sur la différence entre ce qui a été injecté dans le réseau et ce qui en a été soutiré. Si cette quantité est en dehors d'un certain intervalle de tolérance, elle est soit vendue au transporteur en cas d'excédent, soit achetée par EDF en cas de déficit à des prix pénalisants pour EDF.

Par ailleurs, le déséquilibre toléré alimente un autre compte d'écart, lui-même borné, dont les excédents/déficits doivent aussi être vendus/achetés à des prix pénalisants pour EDF, mais ceci à la fin du mois. À la fin donc du mois, ce compte est remis à zéro par un apurement mensuel, aux tarifs moins pénalisants cette fois pour EDF. Les tolérances dépendent de la capacité souscrite et forment une fonction concave qui ressemble à celle de la figure 2.

Sommairement, EDF paye à l'opérateur un coût de transport qui comporte les termes suivants, tous linéaires :

- Terme de capacité d'entrée.
- Terme de capacité de liaison entre zones.

En matière de pénalités, les seuls compléments de prix à payer sont liés aux déséquilibres dans les zones. On indique qu'aussi bien les déséquilibres journaliers que les déséquilibres cumulés sont pénalisés.

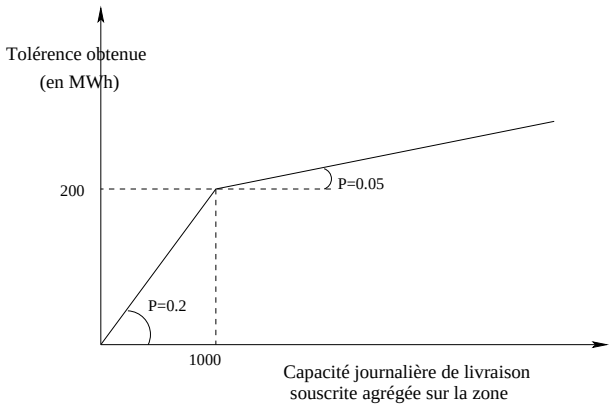


Fig. 2: Tolérances sur les bilans d'écarts.

4.6 Stockage

EDF doit s'engager sur un horizon donné à maintenir une capacité nominale de stockage qui représente la quantité maximale que l'on peut entreposer dans le stock. Dans un second temps, en choisissant le groupement géographique de stockage, on s'engage implicitement sur la capacité journalière de soutirage/injection et sur le stock minimal/maximal.

On met l'accent sur l'impossibilité de la part d' EDF d'injecter/soutirer du gaz au delà des limites contractées. Contrairement à cela, le dépassement des capacités de stockage est permis, bien que pénalisant.

Les restrictions du contrat de stockage portent sur les quantités à injecter et à soutirer et le niveau de stock. la quantité de gaz en stock doit être comprise chaque pas de temps entre deux bornes (cf à la figure 3).

Le principe de la limitation de l'injection et du soutirage est basé sur le fait qu'il n'est pas possible de soutirer de grandes quantités aux moments où le stock est déjà presque vide, et inversement, il n'est pas permis d'injecter beaucoup de gaz quand le niveau de stock est proche de sa capacité maximale. En bref, la capacité d'injection (c'est-à-dire la quantité maximale qu'EDF peut faire entrer dans le stock un jour donné) est une fonction de la capacité de stockage, ainsi que du niveau de stock actuel. Il en

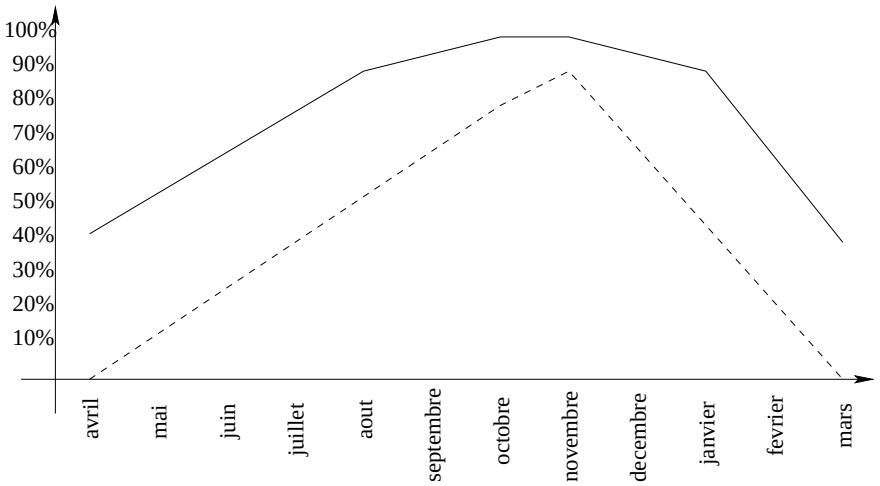


Fig. 3: Le niveau de stock maximal (—) et le niveau de stock minimal (- - -) pour chaque mois de l'année et pour un groupement spécifique.

va de même pour la capacité de soutirage.

Le facteur qui va multiplier la capacité de stockage pour déduire la capacité d'injection (soutirage) porte le nom *facteur de réduction*. La figure 4 montre deux profils, un d'injection et un de soutirage pour un groupement de stockage spécifique. La forme des courbes est similaire pour les autres groupements. Une simplification déjà faite porte sur la convexité (pour les profils d'injection) et la concavité (pout les profils de soutirage) des courbes. Dans la réalité, il existe des groupements où la convextité (resp. concavité) se perd au delà de 90% du niveau de stock.

La tarification prend en compte les termes suivants :

– un terme lié à la capacité de stockage,
– un terme proportionnel à la capacité de soutirage,
– un terme proportionnel à la capacité d'injection,
– un terme proportionnel à la quantité soutirée depuis le stock, et
– un terme proportionnel à la quantité injectée dans le stock.

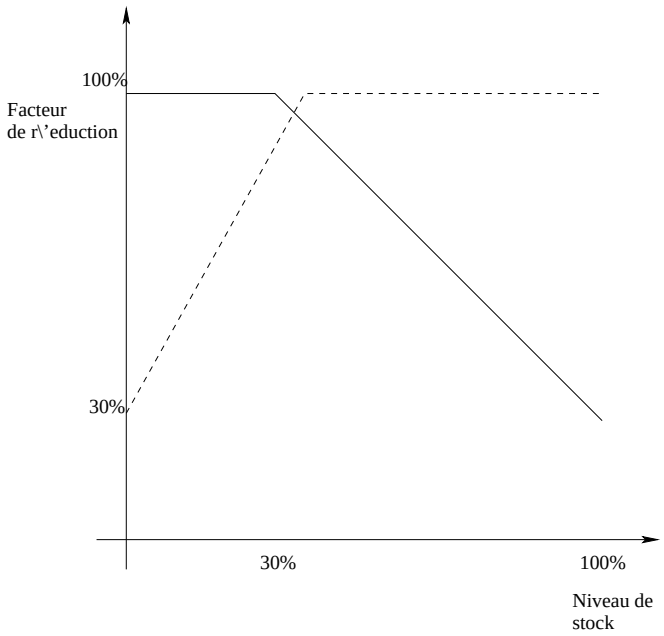


Fig. 4: Profil d'injection (- - -) et de soutirage (—) pour un groupement de stockage.

4.7 Une version simplifiée du problème

Les dimensions du problème décrit dans les paragraphes précédents peuvent vite dépasser les limites de mémoire auquelles sont soumises les logiciels d'optimisation (du type CPLEX). Afin d'étudier la structure du problème ainsi que le comportement des méthodes de résolution, on en a extrait une version simplifiée, tout en s'efforcant de préserver la structure et les caractéristiques particulières.

Dans la suite on va alors considérer deux zones d'équilibrage, deux opérateurs de transport, deux contrats d'approvisionnement et un point de livraison à chaque zone d'équilibrage qui correspond aux clients finaux. L'agrégation de la demande se fait alors au niveau de la zone d'équilibrage. Par conséquent la demande sera indicée dans la suite par zone d'équilibrage.

Les aléas du problème se réduisent à la demande clientèle et aux prix spots. Le portefeuille de clients se modélise à travers les scénarios de demande, c'est-à-dire qu'il est inclus de manière indirecte dans les scénarios de demande. Comme la demande

est agregée par zone d'équilibrage, il est normal que ces deux aspects incertains se confondent. La défaillance des producteurs n'a pas été prise en compte et donc on considère qu'ils sont tout le temps opérationnels.

Chaque zone d'équilibrage comporte un seul point d'enlèvement auquel les deux *producteurs* peuvent injecter du gaz dans le réseau gazier. Comme on n'a qu'un seul point d'enlèvement par zone d'équilibrage, le point d'entrée équivaut à la zone d'équilibrage.

Chaque zone d'équilibrage comporte un seul point de stockage, ainsi qu'un seul point de livraison. Les PEG (points d'échange du gaz) n'ont pas été pris en compte. Comme on n'a qu'un seul point de stockage par zone d'équilibrage, l'indice sur le groupement peut être remplacé par l'indice sur la zone d'équilibrage.

Si pour diverses raisons les ressources d'une zone d'équilibrage (contrats d'approvisionnement ou stockage) ne peuvent pas être utilisées pour couvrir la demande, EDF dispose du droit de faire entrer du gaz en provenance de l'autre zone d'équilibrage à travers la liaison entre zones. Cette liaison a une certaine capacité qui ne doit pas être dépassée.

Il n'y a pas de coût fixe de souscription de capacité d'approvisionnement ni de stockage. Dans la réalité il y en a un pour le stockage. On ne le prend pas en compte puisque le coût est marginal, ceci permettant d'éviter l'utilisation des variables binaires dans la modélisation liées à la décision du type "souscription ou pas au tel contrat".

Seuls les contrats d'approvisionnement à long terme ont été pris en compte et les contrats spot pour l'achat/vente du gaz du jour au lendemain. Les prix des premiers sont déterministes, alors que les prix des seconds sont donnés à travers des scénarios.

On est obligé de souscrire des capacités d'approvisionnement, d'acheminement et de stockage, comme elles ont été décrites aux paragraphes précédents.

L'empilement de contrats d'acheminement n'a pas été pris en compte. Le contrat sur la capacité d'acheminement a la même durée de validité que le contrat d'approvisionnement. La capacité d'approvisionnement est entendu comme la capacité d'enlèvement au point d'entrée. La quantité maximale qu'EDF peut faire entrer dans la zone d'équilibrage est la même que la quantité maximale qu'EDF (par l'intermédiaire d'un opérateur de transport) peut enlever chaque jour du point d'entrée rattaché à cette zone. À par-

tir de maintenant, lorsque on se refère à la capacité d'acheminement on entendra la capacité de liaison entre les zones, puisque la capacité de livraison équivaut à la demande clientèle (qui doit être satisfaite à 100%) et la capacité d'entrée est désormais la capacité d'approvisionnement.

Le déséquilibre dans les zones est interdit et les pénalités ne s'appliquent pas. La capacité minimale d'approvisionnement peut être violée, sachant qu'EDF paiera de toutes façons pour cette capacité même si la quantité équivalente n'a pas été consommée.

4.8 Modèle déterministe

La modélisation qui va suivre se basera sur la version simplifiée du problème décrit. On introduira d'abord le système de notation que l'on suivra, on expliquera ensuite toutes les contraintes une par une et enfin la fonction économique sera formulée. Il s'agira d'un programme linéaire dynamique stochastique où l'aléa apparaît à la fois dans le second membre et dans la fonction objectif. (Si la défaillance des producteurs était prise en compte, la matrice technologique ne serait plus une matrice déterministe).

•• Système de notation

- C : désignera toutes les capacités.
- Q : désignera toutes les quantités.
- γ : suivi de la capacité/quantité désignera le coût de cette capacité/quantité.
- A : désignera la capacité/quantité liée à l'approvisionnement.
- L : désignera la capacité/quantité liée à l'acheminement (liaison).
- S : désignera la capacité/quantité liée au stockage.
- in : indicera tout ce qui entre dans le réseau (en provenance des points d'entrée ou des points de stockage).

– out : indicera tout ce qui sort du réseau (à destination des clients ou des points de stockage).

•• Indices

– t : période de temps

– z : zone d'équilibrage

– ca : contrat d'approvisionnement

On fait remarquer qu'il n'est plus besoin d'indicer le groupement de stockage, puisque il n'y en qu'un seul par zone d'équilibrage et donc l'indice z sur le stock indique le groupement appartenant à cette zone. Il en va de même pour le point d'entrée (un par zone) et le transporteur qui achemine du gaz dans la zone (il n'y a qu'un seul opérateur par zone – différents opérateurs à différentes zones).

•• Approvisionnement

– $\mathbf{CA}_{z,ca}$: Capacité d'approvisionnement nominale réservée dans le contrat ca au point d'entrée associé à la zone z.

– $\mathbf{QA}_{z,ca,t}$: Quantité approvisionnée suivant le contrat ca au point d'entrée associé à la zone z en période t.

– $\mathbf{QAC}_{z,ca,t}$: Quantité cumulée d'approvisionnement suivant le contrat ca au point d'entrée associé à la zone z en période t.

– $\mathbf{SPin}_{z,t}$: Quantité approvisionnée depuis le marché spot au point d'entrée associé à la zone z en période t.

– $\mathbf{SPout}_{z,t}$: Quantité de gaz en zone z vendue au marché spot en période t.

– $\mathbf{nJ}_{z,ca}$: Le facteur (*nombre de jours*) qui divise la capacité d'approvisionnement dans la zone z suivant le contrat ca afin d'obtenir la capacité journalière.

- $\mathbf{fp}^{max}_{z,ca}$: Le facteur de flexibilité qui multiplie la capacité d'approvisionnement nominale dans la zone z suivant le contrat ca afin d'obtenir la capacité maximale d'approvisionnement.
- $\mathbf{fp}^{min}_{z,ca}$: Le facteur de flexibilité qui multiplie la capacité d'approvisionnement nominale dans la zone z suivant le contrat ca afin d'obtenir la capacité minimale d'approvisionnement.
- $\mathbf{fj}^{max}_{z,ca}$: Le facteur de flexibilité dans la zone z suivant le contrat ca qui en multipliant la capacité journalière maximale nominale permet de définir la capacité journalière maximale.
- $\mathbf{fj}^{min}_{z,ca}$: Le facteur de flexibilité dans la zone z suivant le contrat ca qui en multipliant la capacité journalière maximale nominale permet de définir la capacité journalière minimale.
- $\gamma\mathbf{QA}_{z,ca,t}$: Coût unitaire de quantité d'approvisionnement fourni dans le cadre du contrat ca au point d'entrée associé à la zone z en période t.
- $\gamma\mathbf{SPin}_{z,t}$: Coût unitaire d'achat de la molécule dans le marché spot pour appovisionner la zone z en période t.
- $\gamma\mathbf{SPout}_{z,t}$: Coût unitaire de vente de la molécule dans le marché spot depuis la zone z en période t.

•• Acheminement

- $\mathbf{D}_{z,t}$: Demande clientèle aggregée de la zone d'équilibrage z en période t.
- $\mathbf{CL}_{z_1,z_2}$: Capacité journalière de liaison de la zone z_1 à destination de z_2.
- $\mathbf{QL}_{z_1,z_2,t}$: Quantité transférée de la zone z_1 vers la zone z_2 en période t.

– $\gamma\mathbf{CL}_{z_1,z_2}$: Coût unitaire de souscription de capacité de liaison de la zone z_1 vers z_2.

•• Stockage

– $\mathbf{CS}_z$: Capacité nominale de stockage au groupement lié à la zone z.

– $\mathbf{QS}_{z,t}$: Niveau de stock au groupement lié à la zone z en période t.

– $\mathbf{QSin}_{z,t}$: Quantité soutirée depuis le stock au groupement lié à la zone z en période t.

– $\mathbf{QSout}_{z,t}$: Quantité injectée dans le stock au groupement lié à la zone z en période t.

– $\mathbf{nJin}_z$: Le facteur qui divise la capacité de stockage afin d'obtenir la capacité maximale d'injection dans le groupement lié à la zone z.

– $\mathbf{nJout}_z$: Le facteur qui divise la capacité de stockage afin d'obtenir la capacité maximale de soutirage dans le groupement lié à la zone z.

– $\gamma\mathbf{CS}_z$: Coût unitaire de réservation de capacité de stockage au groupement lié à la zone z.

– $\mathbf{NSmin}_{zt}$: Le coefficient qui multiplie la capacité nominale de stockage pour obtenir la capacité de stockage minimale en période t au groupement lié à la zone z.

– $\mathbf{NSmax}_{zt}$: Le coefficient qui multiplie la capacité nominale de stockage pour obtenir la capacité de stockage maximale en période t au groupement lié à la zone z.

Satisfaction de la demande

$$\begin{aligned}\textstyle\sum_{ca} \mathbf{QA}_{z,ca,t} + \mathbf{QL}_{z',z} - \mathbf{QL}_{z,z'} + \mathbf{QSin}_{z,t} - \mathbf{QSout}_{z,t} + \\ + \mathbf{SPin}_{z,t} - \mathbf{SPout}_{z,t} = \mathbf{D}_{z,t} \qquad \forall\ z,\ t\end{aligned} \tag{4.1}$$

Mise à jour du stock

$$\mathbf{QS}_{z,t-1} + \mathbf{QSout}_{z,t-1} - \mathbf{QSin}_{z,t-1} = \mathbf{QS}_{z,t} \qquad \forall\ z,\, t > 1 \tag{4.2}$$

Mise à jour du bilan de la quantité d'approvisionnement cumulée

$$\mathbf{QAC}_{z,ca,t-1} + \mathbf{QA}_{z,ca,t-1} = \mathbf{QAC}_{z,ca,t} \qquad \forall\ z,\, t > 1 \tag{4.3}$$

Respect de capacité journalière d'approvisionnement maximale

$$\mathbf{QA}_{z,ca,t} \leq \mathbf{CA}_{z,ca} \frac{\mathbf{fAj}^{max}_{z,ca}}{\mathbf{nJ}_{z,ca}} \qquad \forall\ z,\, ca,\, t \tag{4.4}$$

Capacité d'approvisionnement journalière minimale - Payer même sans avoir consommé

$$\begin{aligned} \mathbf{QAP}_{z,ca,t} &\geq \mathbf{QA}_{z,ca,t} && \forall\ z,\, ca,\, t \\ \mathbf{QAP}_{z,ca,t} &\geq \mathbf{CA}_{z,ca} \tfrac{\mathbf{fAj}^{min}_{z,ca}}{\mathbf{nJ}_{z,ca}} && \forall\ z,\, ca,\, t \end{aligned} \tag{4.5}$$

Respect de capacité d'approvisionnement maximale

$$\mathbf{QAC}_{z,ca,T} \leq \mathbf{CA}_{z,ca} \mathbf{fAp}^{max}_{z,ca} \qquad \forall\ z,\, ca \tag{4.6}$$

Respect de capacité d'acheminement (de liaison)

$$\mathbf{QL}_{z_1,z_2,t} \leq \mathbf{CL}_{z_1,z_2} \qquad \forall\ z_1,\, z_2,\, t \tag{4.7}$$

Respect de la quantité dans le stock

$$\mathbf{QSout}_{z,t} \leq \mathbf{QS}_{z,t} \qquad \forall\ z,\, t \tag{4.8}$$

Respect de capacité de stockage maximale et minimale

$$\mathbf{CS}_z \mathbf{NSmin}_{zt} \leq \mathbf{QS}_{z,t} \leq \mathbf{CS}_z \mathbf{NSmax}_{zt} \qquad \forall\ z,\, t \tag{4.9}$$

Respect de capacité maximale d'injection

$$\mathbf{QSin}_{zt} \leq \alpha_{in} \mathbf{CS}_z + \beta_{in} \mathbf{QS}_{z,t} \qquad \forall\ z,\, t \tag{4.10}$$

Respect de capacité maximale de soutirage

$$\mathbf{QSout}_{zt} \leq \alpha_{out}\mathbf{CS}_z + \beta_{out}\mathbf{QS}_{z,t} \qquad \forall\ z,\ t \tag{4.11}$$

On précise que les coefficients α_i et β_i sont employé pour décrire les fonctions linéaires par morceaux portant sur les profils d'injection et de soutirage des figures 4.

Fonction objectif

$$\begin{array}{ll} \min & \sum_{z_1,z_2} \mathbf{CL}_{z_1,z_2} \cdot \gamma\mathbf{CL}_{z_1,z_2} \\ & + \sum_z \mathbf{CS}_z \cdot \gamma\mathbf{CS}_z \\ & + \sum_z \mathbf{CS}_z(\mathbf{nJin}_z + \mathbf{nJout}_z) \cdot \gamma\mathbf{CS}_z \\ & + \sum_{z,ca,t} \mathbf{QAP}_{z,ca,t} \cdot \gamma\mathbf{QA}_{z,ca,t} \\ & + \sum_{z,ca,t} \mathbf{SPin}_{z,t} \cdot \gamma\mathbf{SPin}_{z,t} \\ & - \sum_{z,ca,t} \mathbf{SPout}_{z,t} \cdot \gamma\mathbf{SPout}_{z,t} \end{array} \tag{4.12}$$

4.9 Modèle stochastique

La modélisation présentée ci-dessus prend en compte une seule succession de demande et de prix possible. Or, la demande des clients $\mathbf{D}_{z,t}$ à chaque période et à chaque zone d'équilibrage, ainsi que les prix au marché spot $\gamma\mathbf{SP}_{z,t}$ ne sont pas connus avec certitude. Ils sont des paramètres aléatoires qui sont considérés comme étant donnés à travers un arbre de scénarios, ressemblant à celui présenté au paragraphe 3.1.

On considère un arbre de scénarios dont la racine représente l'instant actuel. À chaque instant où un paramètre aléatoire se réalise, on considère avoir un branchement sur l'arbre. Remarquons qu'à chaque branchement, une décision doit être prise (sinon, on pourrait regrouper plusieurs étapes en une seule). On associe alors à chaque nœud une réalisation des paramètres aléatoires et un vecteur de variables de décision.

Le passage du modèle déterministe au modèle stochastique est immédiat. Il suffit de remplacer l'indice portant sur le temps t par l'indice portant sur le nœud n correspondant. On présente ci-après une partie seulement des équations déjà présentées au paragraphe 4.8, mais cette fois-ci, en stochastique. On numérote tous les nœuds de l'arbre de 0 à N–1. L'indice n porte sur le nœud et $\alpha(n)$ sur le nœud-père (prédécesseur) du nœud n.

Satisfaction de la demande

$$\begin{aligned}\textstyle\sum_{ca} \mathbf{QA}_{z,ca,n} + \mathbf{QL}_{z',z} - \mathbf{QL}_{z,z'} + \mathbf{QSin}_{z,n} - \mathbf{QSout}_{z,n} + \\ + \mathbf{SPin}_{z,n} - \mathbf{SPout}_{z,n} = \mathbf{D}_{z,n} \qquad \forall\ z,\ n\end{aligned} \tag{4.13}$$

Mise à jour du stock

$$\mathbf{QS}_{z,\alpha(n)} + \mathbf{QSout}_{z,\alpha(n)} - \mathbf{QSin}_{z,\alpha(n)} = \mathbf{QS}_{z,n} \qquad \forall\ z,\ n \neq 0 \tag{4.14}$$

Mise à jour du bilan de la quantité d'approvisionnement cumulée

$$\mathbf{QAC}_{z,ca,\alpha(n)} + \mathbf{QA}_{z,ca,\alpha(n)} = \mathbf{QAC}_{z,ca,n} \qquad \forall\ z,\ n \neq 0 \tag{4.15}$$

Respect de capacité journalière d'approvisionnement maximale

$$\mathbf{QA}_{z,ca,n} \leq \mathbf{CA}_{z,ca} \frac{\mathbf{fAj}^{max}_{z,ca}}{\mathbf{nJ}_{z,ca}} \qquad \forall\ z,\ ca,\ n \tag{4.16}$$

Fonction objectif

$$\begin{aligned}\min \quad & \textstyle\sum_{z_1,z_2} \mathbf{CL}_{z_1,z_2} \cdot \gamma\mathbf{CL}_{z_1,z_2} \\ & + \textstyle\sum_{z} \mathbf{CS}_{z} \cdot \gamma\mathbf{CS}_{z} \\ & + \textstyle\sum_{z} \mathbf{CS}_{z}(\mathbf{nJin}_{z} + \mathbf{nJout}_{z}) \cdot \gamma\mathbf{CS}_{z} \\ & + \textstyle\sum_{z,ca,n} \mathbf{QAP}_{z,ca,n} \cdot \gamma\mathbf{QA}_{z,ca,n} \\ & + \textstyle\sum_{z,ca,n} \mathbf{SPin}_{z,n} \cdot \gamma\mathbf{SPin}_{z,n} \\ & - \textstyle\sum_{z,ca,n} \mathbf{SPout}_{z,n} \cdot \gamma\mathbf{SPout}_{z,n}\end{aligned} \tag{4.17}$$

La structure du problème en stochastique sera étudiée dans le chapitre suivant. Une grande famille de problèmes ressemblant aux problèmes de dimensionnement de capacités et gestion de flux et de stocks s'indentifieront dans la structure du modèle présenté.

Remarque

Il serait utile, avant de conclure ce chapitre, de faire remarquer que la modélisation par scénarios parallèles avec des contraintes de non-anticipativité est facilement applicable sur la version déterministe

présentée au paragraphe 4.8. Il suffit d'ajouter l'indice s portant sur les différents scénarios à toute variable de décision du modèle. Bien évidemment, la demande et les prix (les paramètres aléatoires) seront également indicés. Il ne faut toutefois pas manquer de rajouter au modèle les contraintes de non-anticipativité, qui garantissent l'implémentabilité de la solution optimale[1].

Voici quelques contraintes écrites suivant la modélisation par scénarios parallèles avec contraintes de non-anticipativité :

Mise à jour du stock

$$\mathbf{QS}^s_{z,t-1} + \mathbf{QSout}^s_{z,t-1} - \mathbf{QSin}^s_{z,t-1} = \mathbf{QS}^s_{z,t} \qquad \forall\ z,\ t{>}1,\ s \tag{4.18}$$

Contraintes de non-anticipativité

$$\mathbf{QS}^s_t - \mathbf{QS}^{s'}_t \quad \forall\, t,\ s{\neq}s' \in \mathcal{B}_t \tag{4.19}$$

$$\mathbf{QSin}^s_t - \mathbf{QSin}^{s'}_t \quad \forall\, t,\ s{\neq}s' \in \mathcal{B}_t \tag{4.20}$$

$\mathcal{B}_t$ représente l'ensemble de scénarios qui partagent le même passé au moment t et dont les valeurs doivent, par conséquent, être égales.

4.10 Conclusions

On a introduit l'application qui nous a incités à étudier le domaine des modèles de recours multi-étapes. Le contexte dans lequel s'inscrit cette application est un marché gazier avec divers opérateurs (producteurs, transporteurs et opérateurs de stockage).

Tout expéditeur souhaitant entrer dans ce marché gazier doit passer des contrats – portant essentiellement sur les capacités – avec chaque acteur, son objectif étant de minimiser le coût de souscription et de gestion sur un horizon d'étude fini. Sous-dimensionner ses capacités oblige l'expéditeur à acheter des quantités supplémentaires à des prix pénalisants. D'autre part, le surdimensionnement des capacités fait augmenter le coût sans aucune utilité.

Les sources d'aléa portent sur la demande clientèle et le prix du gaz au marché spot. On dispose d'un ensemble de scénarios qui est censé représenter tout cas possible.

[1] Remarquons que pour les applications industrielles, où le modèle déterministe existe déjà, la prise en compte de l'incertitude suivant la modélisation par contraintes de non-anticipativité est particulièrement attirante. Il souffit de créer autant de modèles que de scénarios et de rajouter seulement les contraintes de non-anticipativité à la fin.

L'expéditeur cherche les capacités lui permettant de répondre à tout scénario de demande et de prix à un coût minimum *en moyenne* et sur un horizon d'étude qui s'étend à plusieurs périodes (typiquement d'un an).

La modélisation déterministe d'une version simplifiée du problème a été présentée. Sa version stochastique est basée sur la modélisation par arbre de scénarios (à l'opposé des scénarios parallèles avec contraintes de non-anticipativité, cf au paragraphe 3.1), et elle a été également présentée.

Dans le chapitre suivant, on essayera d'exploiter sa structure particulière. On analysera l'application de la méthode de décomposition imbriquée (présentée au paragraphe 3.3) sur ce problème. La taille de l'arbre de scénarios nécessaire, pour décrire aussi finement que possible la réalité, augmente exponentiellement avec le nombre de périodes, ceci limitant la possibilité de considérer de très grandes instances (de l'ordre de centaines de milliers de scénarios). On proposera dans les paragraphes du chapitre 5 de nouvelles techniques pour traiter des problèmes multi-étapes de très grande taille, qui présentent, en même temps, la même structure que le problème présenté dans ce chapitre. On identifiera, à travers la structure de sa matrice de contraintes, une grande famille de problèmes similaires.

Chapitre 5

Nouvelles mises en œuvre de la décomposition imbriquée sur des arbres de scénarios de très grande taille

5.1 Introduction

Dans le présent chapitre on développera de nouvelles mises en œuvre de l'algorithme de décomposition imbriquée sur une version simplifiée du problème de dimensionnement de capacités et de gestion de flux et de stocks stochastiques décrit au chapitre 4. La décomposition imbriquée constitue une des méthodes des plus utilisées dans les applications. La méthode ne fait intervenir aucun terme quadratique (contrairement aux méthodes lagrangiennes présentées au chapitre 3) et donc les sous-problèmes peuvent être efficacement résolus en utilisant tout solveur de programmation linéaire[1]. Les tailles des arbres que nous allons traiter atteignent des centaines de milliers de

[1]On fait surtout allusion aux logiciels libres comme COIN-OR, qui semblent être particulièrement efficaces lorsque les problèmes traités sont linéaires, ceci n'étant apparemment pas le cas pour les problèmes quadratiques ou mixtes.

nœuds, dépassant largement les dimensions pouvant être traitées par des logiciels standards tels que CPLEX. L'efficacité de la méthode que nous allons décrire apparaîtra d'autant plus importante que la taille des problèmes (et des arbres de scénarios) sera grande.

Le problème de dimensionnement présente une structure particulière que l'on essayera d'exploiter. Dans le paragraphe qui suit on identifiera la structure du problème qui nous intéresse ; des structures analogues apparaissent également dans de nombreux autres problèmes de dimensionnement, liés à la localisation des entrepôts et au transport ; dimensionnement des réseaux de télécomunication ; gestion de portefeuilles d'actions, etc. Dans le paragraphe 5.3 on formalise l'algorithme de la décomposition imbriqueé dans le cas de recours complet et on identifie les points sur lesquels l'algorithme nous a semblé améliorable. Les améliorations proposées portent sur deux niveaux : sur la réduction du temps de calcul par itération et sur la réduction du nombre d'itérations. Les méthodes respectives seront décrites dans les paragraphes 5.4 et 5.5. Les expériences qui ont été menées seront expliquées au paragraphe 5.6 et les résultats suivront au paragraphe 5.7.

5.2 Structure d'un problème de dimensionnement

On souhaite souscrire un certain nombre de contrats qui minimisent sur un horizon d'étude fini, le coût de souscription et le coût moyen de gestion de stocks et de pilotage de flux à suivre. Le choix des capacités joue un rôle essentiel dans cette procédure. Il conditionne toutes les décisions qui seront prises ultérieurement, à savoir : les décisions d'injection et de soutirage au niveau des stocks, les décisions sur les quantités approvisionnées, les décisions sur les quantités acheminées et les décisions sur les achats et les ventes sur le marché spot. Le choix de la valeur de chacune des capacités est donc une décision de nature dimensionante.

Remarquons que ce type de problématique peut apparaître dans divers domaines de prise de décisions. Dans nombreuses applications, on pourrait partager les variables de décision en deux ensembles :

1. les variables de décision dimensionantes et structurantes, comme par exemple :

localisation des entrepôts, location ou achat des camions, location des pétroliers, construction des usines, des avions, des barages hydrauliques, des pipelines, des réseaux de télécomunication, reservation des capacités de gazoducs ;

2. les variables de décision de type correctif (de recours), comme par exemple : allocation des véhicules, implémentation des programmes de transport, des programmes d'ordonnancement, de plannification, d'affectation.

Le premier ensemble consiste en des décisions irrévocables, des décisions qui une fois prises ne seront plus remises en question. Le second ensemble consiste en des décisions qui dépendent de la réalisation -au fur et à mesure- des paramètres aléatoires.

Bien-entendu, comme il s'agit d'un problème dynamique, il y aura des décisions prises au moment t qui conditionneront nos actions aux instants ultérieurs $t'>t$. Pour voir ceci, il faut considérer la vision d'arbre de scénarios que l'on a présetée au paragraphe 3.1. Sur cet arbre, les décisions qui ont cette propriété sont liées aux variables couplantes entre différents nœuds, à savoir les variables portant sur le niveau de stocks ainsi que les variables portant sur les quantités approvisionnées, c'est-à-dire les variables qui interviennent dans les contraintes qui décrivent la partie dynamique du problème (cf au paragraphe 4.8 pour la description contraintes dynamiques). C'est cette partie là qui donne au problème son caractère multi-étapes.

Le flux d'information peut être représenté à partir d'un arbre de scénarios. L'information communiquée concerne les réalisations de différentes paramètres aléatoires qui apparaissent dans le problème. Afin de voir l'explosion en dimension, on considère le cas d'un arbre qui s'étend à T périodes d'étude. À l'instant t=0 on a d réalisations possibles des paramètres incertains. Pour chacune de ces réalisations, on aura à l'instant t=1 également d réalisations possibles, et ainsi de suite. Le nombre total de nœuds de l'arbre sera donc de $\sum_{t=0}^{T-1} d^t$. Le nombre de scénarios (c'est-à-dire de feuilles de l'arbre sera de d^{T-1}. À chaque nœud de l'arbre on prend également des décisions qui doivent utiliser toute l'information disponible des instants antérieurs mais qui doivent également faire face à tout ce qui peut se produire aux instants ultérieurs. Le problème qui nous intéresse, exprimé en tant que programme linéaire, possède une structure par blocs qui correspond à la structure de l'arbre des scénarios comme indiqué sur la figure 1.

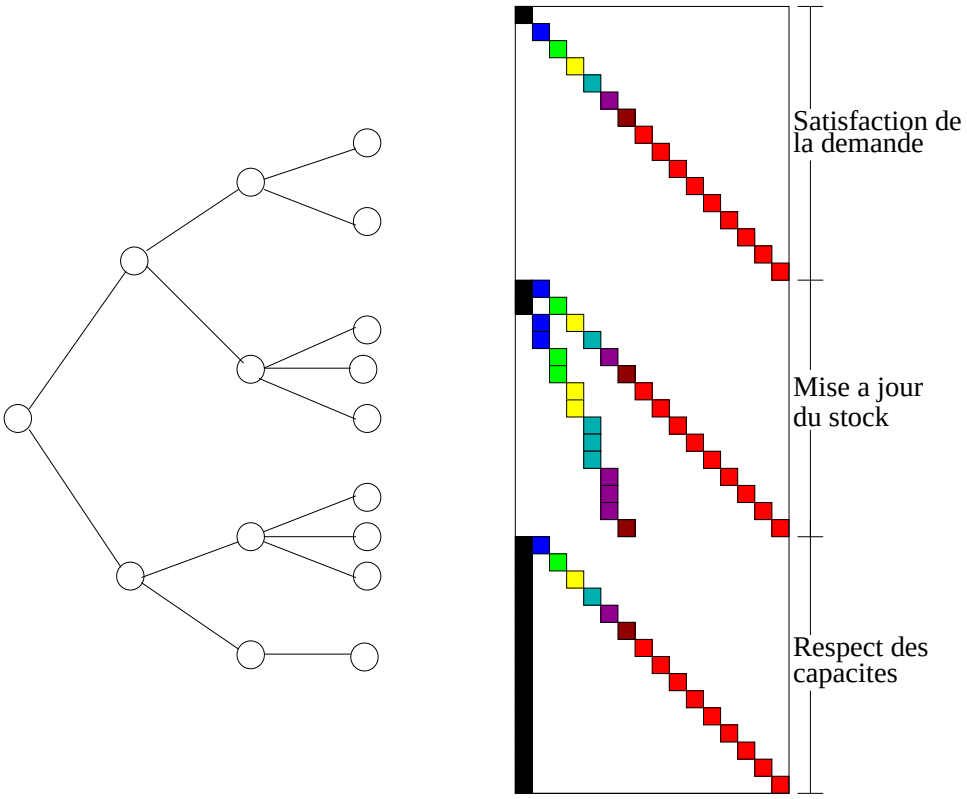


Fig. 1: Arbre de scénarios et structure des blocs du problème de dimensionnement.

La partie supérieure de la matrice de la figure 1 contient les contraintes de satisfaction de la demande et elle est complètement découplée par nœud. La partie du milieu représente la partie dynamique. Il est à noter que la dépendance entre les paires de nœuds ancêtre-successeur, évidente sur la représentation de l'arbre à gauche, apparaît également dans la matrice de contraintes à travers le couplage entre les blocs. Enfin, suit la partie consacrée au respect des capacités qui présente la structure d'un modèle en deux étapes. Cette partie est la partie qui occupe le plus de place en mémoire dans la matrice comme le montre le tableau 2.

Le problème appartient à la catégorie des modèles multi-étapes, or il n'y a que 27% de contraintes qui sont couplantes, tandis qu'il y a deux fois plus de contraintes dont la structure ressemble à celle d'un modèle en deux étapes. En dehors de cela,

Nature de la contrainte	**Couplage entre blocs**	**Nombre de contraintes** *(N :#nœuds de l'arbre de scénarios)*
Satisfaction de la demande	Découplé	$4.N$
Dynamique du système	Couplage multi-étapes	$6.N$
Respect des capacités	Couplage en 2 étapes	$12.N$

Tab. 2: Nombre de contraintes par niveau de couplage.

les variables des capacités interviennent à chaque nœud de l'arbre et donc leur valeur influence toutes les étapes de décision du système. Ceci suggère de s'intéresser en priorité au choix des variables des capacités plutôt qu'aux autres variables de décision du modèle.

5.3 Algorithme de la décomposition imbriquée : recours complet

L'hypothèse de recours complet signifie qu'il n'y aura dans l'algorithme aucune coupe de réalisabilité (le programme (3.4) - page 60 ne contient plus le quatrième groupe de contraintes). Ceci est dû au fait qu'aucun problème de l'arbre ne sera non-réalisable.

On précise que dans la suite on utilisera n pour désigner un nœud quelconque, $\alpha(n)$ sera l'indice de son nœud-père et $\mathcal{S}(n)$ sera l'ensemble des indices de ses nœuds successeurs.

Dans la figure 1 on a vu la structure du grand problème non-décomposé qui nous intéresse. Or, dans l'algorithme de la décomposition imbriquée, un nœud doit avoir un seul père. En regardant les deux parties inférieures de la matrice dans la figure 1, on constate qu'un nœud $n{\neq}0$ quelconque a deux pères : son prédécesseur immédiat et le nœud $n{=}0$.

Afin que la modélisation soit en accord avec le principe d'un seul nœud-père, on a reformulé le problème en rajoutant des contraintes d'égalité qui ne font que propager les variables du nœud-racine. Soient y_0 les variables couplantes, correspondant à une

décision structurante prise au nœud-racine, dont les conséquences affectent tous les autres nœuds. Dans l'application qui nous intéresse, ce sont les variables portant sur les capacités, qui interviennent dans la partie inférieure de la matrice dans la figure 1. Pour propager les valeurs des variables (capacités) du nœud-racine aux nœuds suivants, on crée des copies de ces variables à chaque nœud et on les propage en rajoutant la contrainte suivante :

$$y_n = y_{\alpha(n)} \quad \forall\, n{\neq}0 \tag{5.1}$$

On rappelle que y_n n'est qu'une partie du vecteur de variables x_n. On précise que l'on a également crée des copies de variables – que l'on fait propager de la même façon – des variables appartenant au nœud-père n qui influencent les nœuds-fils (pour l'application que l'on s'est proposée, il s'agit des variables de niveau de stocks et des variables portant sur le bilan de quantités cumulées d'approvisionnement).

On appellera les problèmes associés à chaque nœud de l'arbre des *sous-problèmes*. Tous les sous-problèmes issus du problème que l'on s'est proposé au chapitre 4 se formulent de la façon suivante :

$$\begin{aligned} &\min c_n^T x_n \\ &A x_n = D_n \\ &B x_n = 0 \\ &W x_n = T x_{\alpha(n)} \\ &x_n \geq 0 \end{aligned} \tag{5.2}$$

Le premier groupe de contraintes porte sur la satisfaction de la demande (D_n est la demande au nœud n) et elle représente donc la partie supérieure de la matrice de la figure 1. Le deuxième groupe de contraintes correspond à toutes les autres contraintes : les contraintes de mise à jour de stocks et les contraintes de respect de capacités. On rappelle que l'on a créé des copies de variables des capacités et de variables des niveaux de stocks pour chaque nœud séparément. Le troisième groupe de contraintes correspond aux contraintes (5.1) et c'est elle qui resteindra ces copies à prendre les valeurs prédéfinies aux nœuds précédents. Dans ce groupe sont également contenues les contraintes qui propageront les capacités choisies au nœud-racine, ainsi que les niveaux de stocks fixés au nœud précédent.

Pendant les itérations de la méthode, de nouvelles coupes d'optimalité sont rajoutées au fur des itérations. À l'itération 0, le programme linéaire correspondant au nœud n ressemblera au programme (5.3) :

$$
(\mathrm{PL}_n) \qquad \begin{aligned}
&\min c_n^T x_n + \rho_n \beta_n + (1-\rho_n) \sum_{i \in \mathcal{S}(n)} \theta_n^i p_n^i \\
&A x_n = D_n \\
&B x_n = 0 \\
&W x_n = T x_{\alpha(n)} \\
&\beta_n \geq -M \\
&\theta_n^i \geq -M \quad \forall i \in \mathcal{S}_n \\
&x_n \geq 0
\end{aligned} \tag{5.3}
$$

où p_n^i est la probabilité conditionnelle pour aller du nœud n au nœud i (appartenant à l'ensemble nœuds sucesseurs) et M un grand nombre positif. Les variables β_n et θ_n^i représentent le maximum point à point des fonctions linéaires correspondant aux coupes ajoutées au cours des itérations. Aucune coupe n'est mise à l'itération $k=0$. On reste dans un cadre générique, d'où l'introduction du coefficient $\rho_n \in \{0,1\}$ choisi par l'utilisateur. On pose $\rho_n=1$ si on applique au nœud n la version multi-coupes et $\rho_n=0$ si on applique au nœud n la version avec coupes agrégées. À notre connaissance la modélisation (5.3) regroupant l'application de deux protocoles de coupes est nouvelle. Au cas où on envisage d'appliquer au nœud n les deux protocoles à la fois (à différentes itérations), il faudrait rajouter la contrainte supplémentaire :

$$\beta_n = \sum_{i \in \mathcal{S}(n)} p_n^i \theta_n^i$$

On rappelle que les coupes d'optimalité que l'on rajoute à chaque itération de la méthode sont des hyperplans d'appui de la fonction *"cost-to-go"*. Les coupes (5.4) et (5.5) ci-dessous donnent les coupes à rajouter dans le cas des coupes agrégées et des multi-coupes respectivement. La solution $\bar{x}$ représente la solution de l'itération précédente ; on utilise la notation $\bar{x}$ pour éviter d'ajouter davantage d'indices désignant

l'itération courante.

$$\beta_n \geq \sum_{i \in \mathcal{S}(n)} p_n^i f_i - \lambda_i^T T \bar{x}_n + (T^T \lambda_i)^T x_n \tag{5.4}$$

$$\theta_n^i \geq f_i - \lambda_i^T T \bar{x}_n + (T^T \lambda_i)^T x_n \quad \forall i \in \mathcal{S}(n) \tag{5.5}$$

Nous pouvons maintenant expliciter une description formelle de l'algorithme de la décomposition imbriquée sous hypothèse de recours complet.

Remarque

Lors de l'implémentation de l'algorithme, on a essayé d'exploiter les similarités des sous-problèmes linéaires, le but étant de réduire au maximum la charge au niveau de la mémoire de la machine. On a partagé les nœuds en deux sous-ensembles : le sous-ensemble $\mathcal{F}$ de nœuds qui constituent les feuilles de l'arbre et le sous-ensemble $\mathcal{N}$ de nœuds intérmédiaires. S est le nombre de feuilles (c.à.d le nombre de scénarios) et N est le nombre total de nœuds de l'arbre (feuilles et nœuds intermédiaires ensemble). Si le nœud-racine est numéroté comme n=0, les deux sous-ensembles sont les suivants :

$$\mathcal{N} = \{0, 1, ..., (N–S–1)\} \qquad \mathcal{F} = \{N–S, ..., (N–1)\}$$

Remarquons que toutes les sous-matrices constituant la matrice de contraintes (5.2) ne dépendent pas du nœud n. Les seules parties qui diffèrent d'un nœud à l'autre sont les vecteurs c_n, D_n et $x_{\alpha(n)}$. On essayera d'exploiter cette similarité. On rappelle qu'au cours de l'algorithme, on ajoutera plusieurs coupes à chaque nœud. Ce n'est que les nœuds intermédiaires qui récevront les coupes ; aucune coupe ne sera ajoutée aux feuilles. Cette observation nous incite à créer un seul problème linéaire pour toutes les feuilles. Le problème linéaire générique associé à chaque feuille sera le problème (5.2). Pour passer d'une feuille à l'autre, on modifiera les vecteurs c_n, D_n et $x_{\alpha(n)}$. Les problèmes linéaires correspondant aux feuilles seront résolus séparement. De plus, la résolution de chaque problème donnera un bon point de départ (une base réalisable) pour les problèmes suivants.

En revanche, les coupes rajoutées aux nœuds intermédiaires feront que la similarité entre eux, apparente au début de l'algorithme, disparaîtra au cours des itérations. Pour les nœuds intermédiaires on associera un problème linéaire par nœud. Il serait plus couteux en termes de temps de calcul de créer un seul programme pour tous les nœuds et de modifier les parties différentes en passant d'un nœud à l'autre que de réserver un programme linéaire à chaque nœud. Pourtant la seconde solution augmente la taille de mémoire nécessaire. Le problème linéaire associé au nœud $n \in \mathcal{N}$ avant le début de l'algorithme sera le programme (5.3).

En concluant, lors de l'implémentation, on a crée $N–S$ programmes linéaires (5.3), chacun

correspondant à un nœud intermédiaire, et un seul programme générique (5.2) correspondant à chaque feuille. Ceci reste toutefois un aspect secondaire pour la présentation de la méthode. On souligne que dans la description formelle qui suit (et qui a précédé) on a considéré avoir N programmes linéaires (PL_n), un pour chaque nœud n (que n corresponde à un nœud intermédiaire ou une feuille de l'arbre).

ALGORITHME DE LA DÉCOMPOSITION IMBRIQUÉE EN RECOURS COMPLET

Étape 0. Initialisation

(a). Numéroter tous les nœuds de l'arbre de $0, ..., N–1$.

(b). Construire PL_n pour tout $n=0, ..., N–1$.

Étape 1. Passage vers l'avant

(a). Pour tout $n=0, ..., N–1$:

Soit $x_{\alpha(n)}$ la solution optimale du $PL_{\alpha(n)}$. Résoudre PL_n (5.3).

(b). Calculer la borne supérieure : $\overline{z}=\sum_{n=0}^{N-1} p_n c_n^T x_n$.

Étape 2. Passage vers l'arrière

(a). Pour tout $n=N–S, ..., 0$ (c'est-à-dire pour tout nœud intermédiaire) :

– Si on applique la version de coupes agrégées en nœud n, alors rajouter la coupe d'optimalité au problème (5.4) PL_n.

– Si on applique la version de multi-coupes en nœud n, alors rajouter les coupes d'optimalité au problème (5.5) PL_n.

(b). Soit f_0 la valeur de la fonction objectif du problème PL_0. Calculer la borne inférieure : $\underline{z}=f_0$

Étape 3. Critère d'arrêt

Calculer $\psi=\frac{\overline{z}-\underline{z}}{\overline{z}}$.

Si $\psi \leq \varepsilon$ alors arrêter. Les vecteurs x_n contiennent la politique optimale à appliquer à chaque nœud et $\overline{z}$ le coût respectif. Sinon, retourner à l'Étape 1.

En analysant globalement la méthode de décomposition imbriquée, il y a princi-

palement deux sources d'amélioration possibles, à savoir :

1. La réduction du temps de calcul passé en moyenne sur une itération.
2. La réduction du nombre d'itérations jusqu'à obtention de la condition d'arrêt de l'algorithme.

Les paragraphes 5.4 et 5.5 vont traiter successivement de ces deux aspects respectivement. Afin d'améliorer la performance de la méthode on fera appel à l'échantillonnage (cf au paragraphe 3.10).

5.4 Réduction du temps de calcul passé sur une itération

Dans ce paragraphe nous allons présenter des améliorations de la méthode de décomposition imbriquée qui portent sur la réduction du temps de calcul passé sur une itération. On pourrait agir à deux niveaux : d'une part diminuer le nombre de nœuds parcourus à chaque passage sur l'arbre et d'autre part accélérer le temps de résolution sur chacun de nœuds (paragraphe 5.4.1). Le premier aspect est traité en utilisant une technique d'échantillonnage. Les méthodes d'échantillonnage appliquées au sein de la décomposition imbriquée présupposent l'indépendance entre les variables aléatoires à travers les étapes et ceci simplifie considérablement leur analyse (on trouvera une revue des principales méthodes d'échantillonnage – de Pereira et Pinto [53] et de Donohue et Birge [32] – antérieurement proposées dans la littérature dans l'annexe A). Le problème qui nous intéresse ne remplit pas la condition d'indépendance des variables aléatoires à travers les étapes, et par conséquent, aucune des deux méthodes présentées dans l'annexe ne pourrait être appliquée. Le second aspect est traité par une nouvelle approche sur l'utilisation des coupes agrégées et des multi-coupes ; c'est le sujet qui nous intéressera au paragraphe 5.4.1.

5.4.1 Réduction du temps de calcul passé sur la résolution d'un nœud

On pourrait penser à des stratégies qui conduiraient à une accélération du temps de calcul au niveau microscopique, à savoir au niveau d'un nœud particulier. Le temps de calcul par nœud augmente d'une itération à l'autre à cause des nouvelles coupes que l'on rajoute au programme linéaire associé.

Un objectif envisageable serait de limiter le nombre de coupes par nœud. Ceci peut être réalisé en appliquant la version de coupes agrégées de la méthode de décomposition imbriquée (cf au paragraphe 2.2.1). Néanmois, en n'appliquant qu'une seule coupe par nœud et par itération on défavorise la rapidité de la convergence. Ceci est dû au fait que l'information propagée dans le cas de coupes agrégées n'est pas aussi riche que dans le cas de coupes multiples. D'autre part, ajouter plusieurs coupes par itération, bien que cela réduise le nombre d'itérations, alourdit la charge concernant la mémoire engagée et le temps de calcul peut augmenter considérablement d'une itération à l'autre. Une idée intéressante consiste donc à chercher un compromis entre les deux approches : propager le maximum d'informations sans pour autant augmenter la taille des sous-problèmes.

Le compromis que nous proposons ici consiste en une procédure hybride utilisant des coupes agrégées pour les nœuds éloignés de la racine et des multi-coupes pour les nœuds proches de la racine de l'arbre. À titre d'exemple sur un arbre de 8 étapes avec un facteur de branchement égale à 4 en chaque nœud, on a $4^0, 4^1, 4^2, ..., 4^7$ nœuds à chaque étape respectivement. L'approche par génération de multi-coupes à tous les nœuds conduit à rajouter à chaque itération 21844 coupes. Or, en générant des coupes agrégées, ce nombre est divisé par 4 et donc le nombre de coupes rajoutées n'est plus que de 5461 coupes. Le compromis proposé serait d'appliquer la génération de multi-coupes à un nombre réduit de nœuds, et les coupes agrégées aux nœuds restants. En appliquant par exemple des multi-coupes aux nœuds des 5 premiers niveaux et des coupes agrégées aux nœuds des niveaux ultérieurs, on à généré sur le même exemple) seulement 6484 coupes supplémentaires à chaque nouvelle itération.

Dans la suite on va calculer le nombre de coupes que l'on rajoute à chaque itération dans le cas : (i) des coupes agrégées (CA) et (ii) des multi-coupes (MC). On calculera ensuite le nombre de coupes rajoutées dans le cas hybride (H) que l'on propose. On note τ le point avant lequel on applique la version multi-coupes, alors qu'à partir duquel on applique la version de coupes agrégées. On précise que la numérotation des étapes (périodes) est de $t = 0, ..., T$–1. On se trouve dans le cas d'un arbre équilibré, c'est-à-dire un arbre dont chaque nœud a d>1 fils.

$$CA = \sum_{t=0}^{T-2} d^t \tag{5.6}$$

$$MC = \sum_{t=1}^{T-1} d^t \tag{5.7}$$

$$H = \sum_{t=1}^{\tau} d^t + \sum_{t=\tau+1}^{T-2} d^t \tag{5.8}$$

Proposition 5.1 *Soit un arbre équilibré où chaque nœud a d fils et τ la période avant laquelle on applique la version multi-coupes, et à partir de laquelle on applique la version de coupes agrégées. Soit $\alpha \in [0,1]$ tel que $H = (1-\alpha)CA + \alpha MC$. Alors $\tau = \frac{\ln(\alpha(d^{T-1}-1)+1)}{\ln d}$*

Démonstration *Par substitution :*

$$\begin{aligned}
\sum_{t=1}^{\tau} d^t + \sum_{t=\tau+1}^{T-2} d^t = (1-\alpha)\sum_{t=0}^{T-2} d^t + \alpha \sum_{t=1}^{T-1} d^t \Leftrightarrow \\
\sum_{t=1}^{T-2} d^t + d^\tau = \sum_{t=0}^{T-2} d^t + \alpha(d^{T-1}-1) \Leftrightarrow \\
d^\tau = \alpha(d^{T-1}-1) + 1 \Leftrightarrow \\
\tau = \frac{\ln(\alpha(d^{T-1}-1)+1)}{\ln d}
\end{aligned}$$

♢

Comme τ est entier, il n'est pas toujours possible d'atteindre toute valeur de α que l'on souhaite. Pratiquement, on a à décider entre $\lfloor \tau \rfloor$ et $\lceil \tau \rceil$.

Le paramètre α représente le compromis entre les deux versions (multi-coupes et coupes agrégées) au niveau du nombre de coupes rajoutées. À titre d'exemple, $\alpha = 0$ donnerait l'application de la version de coupes agrégées, alors que $\alpha = 0.5$ placerait τ à une période telle que le nombre de coupes rajoutées à chaque itération soit au milieu entre le nombre de coupes rajoutées dans le cas de coupes agrégées et dans le cas de multi-coupes.

Sommairement, le grand avantage de la procédure hybride de coupes en comparaison avec la version de multi-coupes est que l'on économise toutes les coupes que renvoient les nœuds de la dernière période à leurs nœuds-pères. Comme contrepartie, on rajoute d^τ coupes suivant le point τ où on choisit de passer des multi-coupes aux coupes agrégées. L'avantage que la réduction $H{-}MC$ apporte est donc d'autant plus important que le nombre de feuilles (nombre de scénarios) est grand.

La question qui se pose est comment choisir cette période τ où on alterne des multi-coupes aux coupes agrégées. Il n'y a pas une réponse générale à cette question. Dans les problèmes où les coupes agrégées se comportent mieux que les multi-coupes, il serait préférable de placer le point τ plus proche de la racine que dans les cas où les multi-coupes présentent un meilleur comportement par rapport aux coupes agrégées. En tout cas, un choix du paramètre α dans l'intervalle $[0.3, 0.5]$ semble empiriquement être un bon choix, comme le montrent les résultats expérimentaux présentés en fin de chapitre.

Au niveau algorithmique, on se réfère à la description de l'algorithme de décomposition imbriquée (cf au paragraphe 5.3). En ayant choisi le point τ où on passe des multi-coupes aux coupes agrégées, on applique à l'Étape 2.a l'instruction correspondante, suivant que l'on est en période $t{\geq}\tau$ ou $t{<}\tau$ respectivement.

5.5 Réduction du nombre d'itérations

La méthode de décomposition imbriquée peut être considérée comme une généralisation de la décomposition de Benders. La décomposition de Benders comprend principalement deux étapes :

- Pour la solution courante génerer le plan sécant qui approximerait le mieux l'impact de ce choix à la deuxième étape.
- Calculer une nouvelle solution candidate, en résolvant un programme linéaire.

Chaque itération de la méthode a pour résultat de diminuer l'écart entre un minorant et un majorant de l'optimum. Une itération résussie serait une itération où la différence entre ces deux bornes baisse le plus possible. Si une série d'itérations n'est constituée que d'itérations réussies, on obtiendrait plus rapidement la convergence de

ces deux bornes.

Ce qui fait alors qu'une itération est réussie dépend pour une grande part, du choix de la *solution candidate*. Pour un problème en deux étapes, la *solution candidate* serait évidemment le choix de la première étape. Or, dans le cas des modèles multi-étapes, il y a une solution candidate à chaque paire de nœuds du type *ancêtre-successeur* qui est la solution propagée du nœud-ancêtre aux nœuds successeurs. Comme on l'a fait remarquer dans le paragraphe 5.2, la structure d'un problème de dimensionnement nous incite à accorder une plus grande importance aux variables de capacités qu'aux autres. C'est la raison pour laquelle, dorénavant lorsque l'on se référera au terme *solution candidate* on entendra tacitement la solution restreinte aux variables du premier nœud de l'arbre.

Le choix de la solution candidate semble jouer un rôle important à la convergence de la décomposition imbriquée. Si on consacrait plus de temps à trouver la solution qui produirait une forte réduction de l'écart entre borne supérieure et inférieure, le temps de calcul total de la méthode décroîtrait. Notre objectif devient la recherche des solutions candidates susceptibles de réduire la distance entre les deux bornes.

À notre connaissance ce type d'approche ne semble pas avoir été étudiée. L'idée que l'on proposera est d'avoir un ensemble de solutions candidates parmi lesquelles on choisira celle qui semble être la plus prometteuse. Il nous faudra alors une règle pour choisir cet ensemble et un critère d'évaluation de chacune de ces solutions. La réponse aux deux questions posées sera donnée à l'aide de l'échantillonnage.

5.5.1 Construction d'un ensemble de solutions candidates

De l'arbre initial, on extrait par tirage aléatoire un sous-arbre de telle manière à ce que le nombre de périodes reste invariable et le facteur de branchement diminue. On passe alors d'un arbre de N périodes et de d branches par nœud à un arbre de N périodes et de $d'<d$ branches par nœud. Cet échantillonnage est connu comme un *échantillonnage conditionnel* (cf au paragraphe 3.10 et référence [64]). Le problème correspondant à ce nouvel arbre est évidemment plus facile à résoudre et nécessite des temps de calcul réduits. On peut facilement extraire plusieurs sous-arbres de ce type

en créant ainsi plusieurs instances de taille réduite du problème.

La procédure de construction de l'ensemble de solutions candidates est déclenchée au début de chaque itération de la méthode que l'on est en train de proposer. À une itération quelconque de l'algorithme, on dispose de l'arbre complet dont les nœuds contiennent déjà un certain nombre de coupes. On génère plusieurs sous-arbres suivant la procédure d'échantillonnage décrite ci-dessus. Plus on souhaite avoir différentes solutions candidates, plus on génère des sous-arbres (dans nos expériences, 5 sous-arbres ont été générés). Les nœuds des sous-arbres échantillonnés préservront les coupes associées aux nœuds échantillonnés[2]. Pour chacun de ces sous-arbres, on fait un certain nombre d'itérations de la décomposition imbriquée (passage vers l'avant et vers l'arrière) jusqu'à ce que la convergence soit atteinte à une tolérance fixée près. De chaque sous-arbre on ne garde que deux solutions : (i) la solution ε-optimale, ainsi que (ii) la solution qui, au cours des itérations, a produit la plus forte réduction de l'écart entre la borne inférieure et la borne supérieure.

Suivant le nombre de sous-arbres générés, on aura construit un ensemble de solutions candidates. On injecte dans le même ensemble la solution que l'algorithme de décomposition imbriquée donnerait à ce phase. Ceci est fait car peut être cette solution est meilleure (au sens défini) que celles proposées par les sous-arbres. Les solutions calculées à la fin de cette procédure passeront le test de l'évaluation. Il ne reste qu'à les évaluer et à proposer finalement celle qui expérimentalement produit la plus forte réduction de l'écart entre les deux bornes.

5.5.2 Critère d'évaluation des solutions candidates

On passe à la deuxième phase de l'algorithme proposé : la phase de l'évaluation des solutions candidates. Il n'y en aura qu'une, parmi toutes celles qui font partie de l'ensemble précédemment créé, qui déterminera les valeurs des variables pour le nœud-racine pour l'itération suivante de la ND appliquée à l'arbre complet.

Un choix possible (qui ne serait évidemment pas le meilleur en termes de temps de calcul) serait de les tester toutes sur l'arbre complet, l'arbre qui est censé représenter

[2]Évidemment les coefficients p_n^i dans la fonction objectif du programme (5.3) ainsi que dans la contrainte (5.4) seront modifiées pour que $\sum_i p_i = 1$; ceci pour tous les sous-arbres.

tous les évènements de façon aussi fine que possible. Or, ceci demanderait un temps de calcul égal au temps de calcul d'une itération complète de l'algorithme ND, et ceci pour toute solution candidate ; le temps de calcul pour une seule itération dans les grands arbres peut déjà être très important (de l'ordre de quelques dizaines de minutes).

L'évalutation ne sera donc pas effectuée sur l'arbre complet, *mais sur un arbre échantillonné*. On extrait alors un autre sous-arbre avec le même nombre de périodes que l'arbre complet et avec un facteur de branchement $d'<d''<d$ et on effectue une seule itération de la ND sur cet arbre qui va jouer en quelque sorte le rôle de simulateur : on simule ce qui se passerait si on appliquait cette solution candidate comme choix de solutions au premier nœud. Les bornes supérieures et inférieures qui seront obtenues nous donneront une estimation des valeurs de ces bornes qui auraient été obtenues par une itération de la ND sur l'arbre complet. La solution retenue sera celle qui présentera l'amélioration la plus grande sur la distance entre les deux bornes.

Les deux étapes décrites aux paragraphes 5.5.1 et 5.5.2 recherchent, évaluent et enfin choisissent une seule solution candidate que l'on impose au nœud-racine. Elles s'effectuent avant chaque itération de la décomposition imbriquée. Nous donnons ci-dessous une description formelle de l'algorithme que l'on propose.

DÉCOMPOSITION IMBRIQUÉE AVEC SOLUTIONS DE DÉPART

Étape 0. Initialisation

- On initialise le compteur d'itérations $k = 0$.
- Soit $\mathcal{A}^k$ l'abre complet à l'itération k comportant N périodes et d branches par nœud.
- Soit $\mathcal{S} = \emptyset$ l'ensemble de solutions à proposer, vide au départ.
- Nous noterons $s^i_{\mathcal{A}'}$ les différentes solutions candidates issues des sous-arbres $\mathcal{A}'\subset\mathcal{A}^k$ qui feront partie de l'ensemble $\mathcal{S}$.

Étape 1. Recherche de solutions candidates

(a). Générer à partir de $\mathcal{A}^k$, des sous-arbres $\mathcal{A}_1, \mathcal{A}_2, ..., \mathcal{A}_M$ par tirages aléatoires indépendants avec un nombre d'étapes N et un facteur de branchement $d'<d$.

(b). Pour la solution $s_{\mathcal{A}^k}$ et pour tout $i = 1, ..., M$ effectuer autant d'itérations

de la méthode ND (cf au paragraphe 5.5.2) que nécessaire sur l'arbre $\mathcal{A}'_i$ jusqu'au moment où une solution ε-optimale soit atteinte (ε est une tolérance prédéfinie, choisie par l'utilisateur).
Au cours des itérations sur l'arbre $\mathcal{A}'_i$, stocker deux solutions :

- $s^1_{\mathcal{A}_i}$ $\rightarrow$ la solution qui a produit la plus grande réduction de l'écart entre borne inférieure et borne supérieure.
- $s^2_{\mathcal{A}_i}$ $\rightarrow$ la solution ε-optimale obtenue.

Rajouter ces deux solutions à l'ensemble $\mathcal{S}$: $\mathcal{S} \leftarrow \mathcal{S} + \{s^1_{\mathcal{A}_i}, s^2_{\mathcal{A}_i}\}$.

(c). Rajouter la solution $s_{\mathcal{A}^k}$ du premier nœud que l'algorithme ND propose à l'itération k : $\mathcal{S} \leftarrow \mathcal{S} + \{s_{\mathcal{A}^k}\}$.

Étape 2. Évaluation des solutions candidates

(a). Génerer (par tirage aléatoire) un nouveau sous-arbre $\mathcal{B} \subset \mathcal{A}^k$ avec un nombre d'étapes N et un facteur de branchement $d'' < d$.

(b). Pour tout i et j effectuer une seule itération de la ND (cf au paragraphe 5.5.2)) en utilisant la solution $s^j_{\mathcal{A}_i}$ sur le sous-arbre $\mathcal{B}$ simulant une itération sur l'arbre $\mathcal{A}^k$.

- En passant d'un sous-arbre à l'autre, supprimer toute coupe rajoutée à tous les nœuds.
- Garder la solution s^* qui a produit le meilleur écart entre borne inférieure et borne supérieure.

Étape 3. Itération sur l'arbre complet **avec** solution de départ

Imposer comme solution de départ la solution s^* retenue à l'Étape 2b. Effectuer une itération de la décomposition imbriquée sur l'arbre complet. Si le critère d'arrêt de la ND n'est pas satisfait, alors retourner à l'Étape 1.

5.6 Expériences numériques

Afin de tester les améliorations proposées précédemment on a généré une série de diverses instances, concernant toujours le problème de dimensionnement des capacités présenté dans le chapitre 4. Les instances créés diffèrent en termes de nombre d'étapes

et de facteur de branchement. On a ainsi créé des instances qui représentent des arbres de struture variées : longs, courts, larges et étroits.

On a codé la version de référence de la décomposition imbriquée et on l'a comparée par rapport aux mises en œuvre des diverses améliorations suggérées précédemment. Les tests ont été effectués sur des PC qui tournent sur LINUX ; la cadence des processeurs utilisés a été autour de 2GHz et la mémoire varie de 1Go à 2Go.

Les arbres générés sont tous équilibrés, c'est-à-dire que le nombre de branches par nœud est constant ; chaque nœud contient la demande clientèle et les prix du marché spot pour chacune des deux zones d'équilibrage. La génération d'arbres de scénarios n'est en général pas un travail trivial[3] : on ne s'intéresse pas à un seul scénario qui sera représentatif en moyenne, mais plutôt un ensemble de scénarios représentatifs ([49]).

La demande et les prix ont été simulés à partir d'un modèle du retour à la moyenne de type processus Ornstein-Uhlenbeck ([31], p.76). Le modèle initial est un modèle du type :

$$dx = h(\bar{x} - x)dt + \sigma dz \tag{5.9}$$

Ceci représente le processus stochastique d'une variable aléatoire x avec la présence d'une *force* qui la pousse vers la valeur $\bar{x}$. La vitesse du processus de retour vers $\bar{x}$ est notée par h, alors que σ désigne la volatilité (écart type). L'équation (5.9) est une équation différentielle stochastique qui peut se résoudre explicitement [44]. Une fois résolue, on discrétise pour obtenir une formule utilisable en simulation :

$$\alpha e^{-hdt} + X(1 - e^{-hdt}) + s\sqrt{\frac{1 - e^{-2hdt}}{2h}}N(0,1)$$

où X est un terme saisonnier et α est la demande/prix du nœud ancêtre, alors que $N(0,1)$ désigne un bruit blanc. Les processus de ce type (mouvement brownien géométrique, retour à la moyenne et processus de diffusion avec sauts) sont parmi les processus aléatoires le plus souvent employés pour simuler des prix dans le domaine de la finance.

[3] Dans ce travail, la génération d'un ensemble représentatif de scénarios n'a pas été notre première préoccupation.

5.7 Résultats

5.7.1 Résultats concernant la réduction du temps de calcul par itération

Dans le tableau 3 qui suit sont contenus les résultats de la comparaison des trois protocoles de traitement de nœuds lors d'une itération, à savoir l'application des multi-coupes, des coupes agrégées et enfin le protocole hybride, en fonction du choix du niveau de l'arbre où on passe des multi-coupes aux coupes agrégées. Sur les expériences montrées dans le tableau 3 ainsi qu'à celles du tableau 5 le point où on arrête l'application des multi-coupes et on passe aux coupes agrégées à été posé de telle manière à ce que $\alpha = 0.3$ (cf à la Proposition 5.1). Ceci signifie que l'on souhaite avoir un nombre de coupes qui se rajouteront à chaque itération qui soit 30% plus proche du nombre de coupes en version de coupes agrégées qu'en version multi-coupes. On arrête l'algorithme dès que l'écart entre borne inférieure et supérieure tombe en dessous de 1%.

	Pb3.150	**Pb4.30**	**Pb5.12**	**Pb6.7**	**Pb7.5**	**Pb8.3**	**Pb9.3**
	22650 nœuds	27930 nœuds	22620 nœuds	19607 nœuds	19530 nœuds	3279 nœuds	9840 nœuds
Coupes agrégées	158 secs	229 secs	299 secs	344 secs	422 secs	76 secs	501 secs
	24 itér	30 itér	38 itér	43 itér	45 itér	42 itér	73 itér
	6.58 s/itér	7.63 s/itér	7.87 s/itér	8.00 s/itér	9.38 s/itér	1.81 s/itér	6.86 s/itér
Multi-coupes	233 secs	322 secs	257 secs	259 secs	460 secs	62 secs	264 secs
	14 itér	20 itér	19 itér	21 itér	32 itér	27 itér	32 itér
	16.64 s/itér	16.10 s/itér	13.53 s/itér	12.33 s/itér	14.38 s/itér	2.30 s/itér	8.25 s/itér
Hybride	101 secs	124 secs	194 secs	167 secs	245 secs	57 secs	192 secs
	15 itér	16 itér	22 itér	21 itér	29 itér	31 itér	33 itér
	6.73 s/itér	7.75 s/itér	8.82 s/itér	7.95 s/itér	8.45 s/itér	1.84 s/itér	5.82 s/itér
Amélioration	1.56	1.85	1.32	1.55	1.72	1.09	1.38

Tab. 3: Comparaison entre les trois protocoles de génération de coupes pour les différentes instances.

On remarque que la version hybride est toujours considérablement plus efficace que les deux autres. La ligne *Amélioration* est justement le rapport entre le temps de la version la plus rapide (multi-coupes ou coupes agrégées) et le temps de la version hybride, et ce rapport dans les expériences a toujours été significativement supérieur à 1. La version hybride semble alors cumuler les avantages des deux autres méthodes, à savoir la légère augmentation du temps de calcul d'une itération à l'autre des coupes agrégées et le nombre faible d'itérations suffisant pour atteindre la convergence des multi-coupes. Le ratio sec/itér représente simplement le ratio entre le temps de calcul et le nombre d'itérations et il constitue ainsi une vitesse moyenne de chaque itération. Bien-sûr la vitesse diminue d'une itération à l'autre puisque de nouvelles coupes se rajoutent.

La figure 4 montre comment la vitesse d'une itération change d'une itération à l'autre suivant le protocole choisi. L'augmentation de la mémoire engagée de la machine est liée à l'augmentation du temps de calcul par itération. Il est clair que la version multi-coupes augmente plus vite ses exigences en mémoire tandis que les deux autres restent relativement modestes.

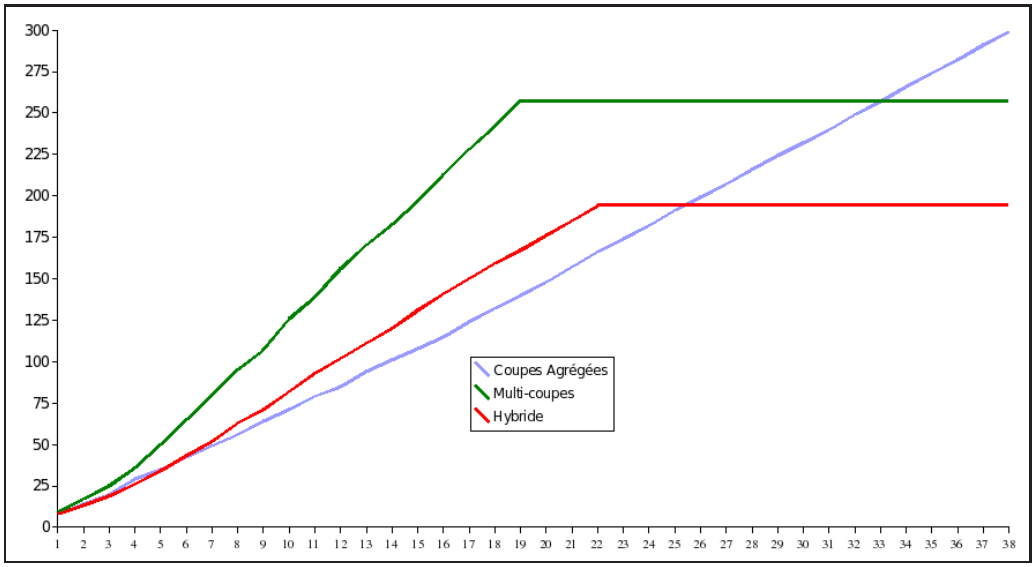


Fig. 4: Temps de calcul par itération en fonction des différentes itérations pour les trois protocoles.

Il mérite d'être noté qu'en général dans les applications, on ne peut pas savoir à l'avance quelle version entre multi-coupes et coupes agrégées est la plus convenable en termes de vitesse de convergence. On est alors obligé soit d'en choisir une au hasard, soit de tester les deux. Si pourtant on utilisait toujours la version hybride, on pourrait être assuré que le temps de calcul ne dépasse pas le temps du meilleur choix entre les deux versions, voire même elle fonctionnera beaucoup mieux.

Avant de conclure ce paragraphe, on joint une seconde série d'expériences sur de plus grandes instances et on compare dans le tableau 5 les résultats obtenus qui confirment les résultats précédents. Les facteurs d'amélioration semblent augmenter par rapport aux instances du tableau 3. Une autre différence entre les deux tableaux est le fait qu'au tableau 5 la version de coupes agrégées semble devancer toujours la version multi-coupes, ce qui montre qu'aux grandes instances où un nombre important de coupes se rajoute à chaque itération, la version multi-coupes (que ce soit sur des arbres longs ou larges) alourdit considérablement la mémoire de la machine (vérification de la remarque de Ruszczyński [61]). Ceci a pour conséquence une diminution importante de la vitesse par rapport à la vitesse de coupes agrégées. Quant à la version hybride, la vitesse moyenne diminue en comparaison avec le tableau 3, dû au fait que les instances sont plus grandes, mais elle reste aussi basse que la vitesse moyenne de coupes agrégées. Il y a même plusieurs instances où la vesion hybride devance en termes de vitesse moyenne la version de coupes agrégées. Ceci est justifié par la diminution du nombre d'itérations jusqu'à convergence.

	Pb3.200	**Pb4.35**	**Pb5.15**	**Pb6.8**	**Pb7.6**	**Pb8.4**	**Pb9.4**
	40200 nœuds	44153 nœuds	54240 nœuds	37448 nœuds	55986 nœuds	21844 nœuds	87380 nœuds
Coupes agrégées	341 secs	421 secs	642 secs	559 secs	1182 secs	621 secs	6220 secs
	27 itér	31 itér	37 itér	38 itér	41 itér	48 itér	59 itér
	12.63 s/itér	13.58 s/itér	17.35 s/itér	14.71 s/itér	28.83 s/itér	12.94 s/itér	105.42 s/itér
Multi-coupes	397 secs	433 secs	733 secs	661 secs	3067 secs	799 secs	34606 secs
	13 itér	17 itér	19 itér	22 itér	31 itér	32 itér	35 itér
	30.54 s/itér	25.47 s/itér	38.58 s/itér	30.05 s/itér	98.94 s/itér	24.97 s/itér	988.74 s/itér
Hybride	141 secs	207 secs	423 secs	324 secs	861 secs	339 secs	3499 secs
	12 itér	16 itér	23 itér	21 itér	33 itér	30 itér	36 itér
	11.75 s/itér	12.94 s/itér	18.39 s/itér	15.43 s/itér	26.09 s/itér	11.30 s/itér	97.19 s/itér
Amélioration	$\times$2.42 fois	$\times$2.03 fois	$\times$1.52 fois	$\times$1.73 fois	$\times$1.37 fois	$\times$1.83 fois	$\times$1.78 fois

Tab. 5: Comparaison entre les trois protocoles de résolution de nœuds pour une seconde série d'instances.

5.7.2 Résultats concernant la réduction du nombre d'itérations

Nous allons procéder à des expériences constituées de deux différentes séries d'instances. La première série contient des problèmes pouvant être résolus par CPLEX, ce sont donc des problèmes dont la matrice de contraintes pourrait être stockée entièrement dans la mémoire de la machine. La seconde série d'expériences comprend des instances où CPLEX ou un autre logiciel d'optimisation ne pourraient pas actuellement être utilisés à cause de leur taille. Dans les deux séries on compare la méthode de la décomposition imbriquée standard avec la version accélérée avec des solutions de départ que l'on a précédemment présentée ; dans la première série on a également inclus les résultats par rapport au temps de calcul issus de différents algorithmes proposés par CPLEX (simplex et barrier).

Que l'on utilise la décomposition imbriquée normale ou la version accélérée que l'on a proposé, on applique le protocole hybride de coupes comme celui-ci a été décrit et a été montré efficace au paragraphe précédent.

L'échantillonnage que l'on effectue sur les arbres de scénarios est un échantillonnage conditionnel et équilibré, c'est-à-dire qu'à chaque étape (période) on génère de façon aléatoire le même nombre de réalisations pour chaque nœud en suivant sa distribution conditionnelle. On aurait pu choisir d'autres manières d'échantillonner qui réduiraient éventuellement la taille du sous-arbre. Quelques remarques et expériences sur ce sujet sont présentés à l'annexe B.

Pour la résolution de grands problèmes déterministes on a utilisé CPLEX9.1. En revanche, les sous-problèmes dans les méthodes de décomposition ont été résolus avec COIN-OR. COIN-OR nous a permis de mieux gérer la mémoire au niveau de modélisation de l'arbre, tandis que le temps de résolution pour les sous-problèmes (problèmes de petite taille ne comportant que de variables continues) n'est en général pas considérablement plus lent que CPLEX.

Première série d'expériences

Dans cette série d'expériences on a arrêté les algorithmes de décomposition imbriquée (ND) et de sa version accélérée (ND++) dès que l'écart entre borne inférieure

et supérieure est tombé en dessous de 0.01% en atteignant ainsi une précision d'ordre 10^{-4}. On applique la stratégie proposée avant que l'écart tombe en dessous de 3%, et ensuite on alterne à la version standard. Au bout d'un certain niveau d'écart, par exemple de 2.5%, il n'est plus facile de distinguer si une solution avec un écart estimé à 2% a réellement une valeur autour de 2% ou si ceci est une erreur statistique. On considère donc qu'en dessous de 3% les solutions sont proposées comme l'algorithme de la décomposition imbriquée le prévoit.

On compare les résultats de la stratégie portant sur le choix des solutions candidates (solution du premier nœud) à la décomposition imbriquée.

Instance nb de nœuds	**Pb3.200** 40200 nœuds	**Pb4.35** 44135 nœuds	**Pb5.15** 54240 nœuds	**Pb6.8** 37448 nœuds	**Pb7.6** 55986 nœuds	**Pb8.4** 21844 nœuds	**Pb9.3** 9840 nœuds
ND	18 itér 234 secs	24 itér 321 secs	31 itér 569 secs	35 itér 537 secs	54 itér 1645 secs	49 itér 566 secs	53 itér 304 secs
ND++	12 itér 180 secs	17 itér 278 secs	24 itér 548 secs	30 itér 496 secs	39 itér 1021 secs	32 itér 401 secs	33 itér 337 secs
Speedup	23.08%	13.40%	3.69%	7.64%	37.93%	29.15%	-10.86%
Simplex	28713 secs	$> 3 \cdot 10^5$ secs	$> 3 \cdot 10^5$ secs	$> 3 \cdot 10^5$ secs	$> 3 \cdot 10^5$ secs	16417 secs	4427 secs
Barrier	449 secs	1625 secs	1276 secs	643 secs	1213 secs	258 secs	64 secs

Tab. 6: Comparaison entre décomposition imbriquée accélérée (stratégie sur le choix des solutions candidates – ND++) et décomposition imbriquée simple (ND) sur différentes instances pouvant être traitées par le logiciel CPLEX .

Le tableau 6 montre que la plupart de temps les algorithmes de décomposition[4]. devancent CPLEX. Barrier est bien plus efficace que Simplex ; Simplex semble d'ailleurs être dramatiquement lent surtout quand le nombre de nœuds et le nombre de périodes augmentent. Simplex n'est jamais plus rapide qu'aucune autre méthode parmi celles affichées au tableau 6, tandis que Barrier pourrait très souvent devancer les méthodes de décomposition et à fortiori la méthode de Simplex.

La capacité des méthodes de points intérieurs sur les traitements des matrices creuses est mise en évidence en observant le comportement de Barrier sur des instances où le nombre de périodes augmente. Voici dans le tableau 7 sur une série d'expériences – où le nombre de périodes est grand – comment Barrier se comporte beaucoup mieux que les méthodes de décomposition.

Instance	**ND**	**ND++**	Barrier
Pb6.5	83 secs	79 secs	30 secs
3905 nœuds			
Pb7.4	122 secs	173 secs	35 secs
5460 nœuds			
Pb7.5	424 secs	373 secs	136 secs
19530 nœuds			
Pb8.3	85 secs	114 secs	24 secs
3279 nœuds			
Pb8.4	566 secs	401 secs	258 secs
21844 nœuds			
Pb9.3	304 secs	337 secs	64 secs
19530 nœuds			

Tab. 7: Instances où la méthode de Barrier devance clairement les méthodes de décomposition.

La conclusion que l'on retient de cette série d'expérience est la vérification d'une

[4]On précise que lors de la résolution des sous-problèmes avec COIN-OR (dans le cadre de la ND et de la ND++) le mode "presolve" était désactivé

observation faite à plusieurs reprises dans la littérature, à savoir que sur des instances qui peuvent être contenues dans la mémoire de l'ordinateur, le meilleur choix à faire serait d'utiliser un logiciel d'optimisation avec une méthode de points intérieurs qui traite bien les parties creuses de la matice des contraintes. En revanche, là où se posent les limites de cette stratégie est le moment où les dimensions du problème deviennent aussi grandes que la mémoire de la machine n'est plus suffisante pour contenir le modèle. Dans la seconde série d'expériences on s'attaquera à des problèmes dont la taille dépasse les capacités que CPLEX (ou un autre logiciel) peut aborder et on est donc obligé de faire appel à des méthodes de décomposition.

Seconde série d'expériences

Une instance de plus de 80.000 nœuds est déjà une instance dont CPLEX n'arrivera jamais à stocker la matrice de contraintes sur une machine d'une mémoire de 2Go avec une mémoire swap de 2Go également. Dans le grand problème déterministe que CPLEX traite, le nombre total de nœuds est caractéristique de la taille du problème. Le nombre de contraintes par nœud se multiplie avec le nombre total de nœuds et aussitôt que ce nombre dépasse les limites de la mémoire, le programme s'arrête brusquement. Or, dans les deux versions de la décomposition imbriquée ce n'est plus le nombre total de nœuds qui fait augmenter la difficulté du problème en termes de mémoire, mais plutôt le nombre de nœuds intermédiaires, qui réçoivent des coupes contrairement aux feuilles qui n'en reçoivent pas.

Dans les expériences qui suivent, l'algorithme a été arrêté dès que l'écart est tombé en dessous de 1%. Il est pourtant vrai que la vitesse de réduction de l'écart (entre borne supérieure et inférieure) diminue beaucoup plus vite quand on est au dessus de 1% que l'inverse. Cela est noté pour faire bien remarquer que le temps que met l'algorithme pour arriver à un écart de 1% n'est pas du tout proche du temps de calcul pour arriver à l'optimum. Par exemple, le tableau 8 montre sur l'instance de dix périodes où on branche trois fois par nœud, comment il faut 17 itérations pour arriver à un écart 1.01638% et encore 17 longues itérations pour arriver à un écart de 0.00946591%. Un écart de 1% pourrait pourtant être acceptable dans certains problèmes. Cet écart peut être obtenu assez rapidement avec la décomposition imbriquée en sa version accélérée

comme ce sera montré par la suite.

Dans toutes les instances du tableau 9 la ND++ – qui représente la stratégie de proposition des solutions candidates – devance la décomposition imbriquée standard. Le nombre d'itérations baisse comme attendu et par conséquent le temps de calcul diminue.

On remarque que sur les arbres courts et larges (peu de périodes – beaucoup de branchements) la décomposition accéléré (ND++) marche beaucoup mieux que la décomposition imbriquée normale (ND). Ceci est caractéristique de la façon dont marche la ND++. On rappelle que la procédure suivie dans la version que l'on propose comprend la phase de la recherche de solutions et la phase d'évaluation des solutions et la selection d'une entre elles ; ensuite suit la phase d'une itération normale de la décomposition imbriquée avec comme solution initiale la solution choisie à l'étape d'avant. Pour que la ND++ soit plus performante que la ND, il faut que les conditions suivantes soient remplies :

1. Le temps d'une itération complète de la ND soit long.
2. Le temps de deux autres phases (recherche et évaluation) soit petit.
3. Le sous-arbre soit assez représentatif de l'arbre réel.

Si alors une itération vaut cher, tandis qu'une simulation peut aller très vite, on souhaiterait passer plus de temps à chercher de bonnes solutions candidates, passer également plus de temps à mieux évaluer la meilleure de ces solutions pour qu'à l'issue d'une longue itération normale on obtienne une amélioration considérable sur l'écart. En revanche, pour avoir une évaluation correcte des solutions candidates, il faudrait que le sous-arbre échantillonné permette d'en déduire des estimations précises sur le calcul des deux bornes (inférieure et supérieure).

Par ailleurs, il faut noter que la procédure de la recherche et de l'évaluation des solutions candidates peut aussi occuper une partie considérable du temps de calcul, dépendant des tailles des sous-arbres échantillonnés. Dans le tableau 10 on affiche les temps mis par itération pour la recherche des solutions candidates, leur évaluation et enfin une itération normale avec la solution proposée comme choix de capacités de la décomposition imbriquée. On inclut à la dernière colonne l'écart atteint à l'issue de chaque itération.

Itération	Écart	Temps de calcul cumulé
1	68.0616%	91 secs
2	34.7524%	142 secs
3	12.7564%	198 secs
4	11.2111%	247 secs
5	8.52785%	288 secs
6	8.01891%	321 secs
7	5.0247%	353 secs
8	3.45571%	386 secs
9	3.45571%	419 secs
10	3.28641%	454 secs
11	2.72922%	486 secs
12	2.58371%	504 secs
13	2.30703%	520 secs
14	2.04377%	534 secs
15	2.03908%	548 secs
16	1.35135%	564 secs
17	1.01638%	580 secs
18	0.772738%	598 secs
19	0.498958%	616 secs
20	0.462486%	634 secs
21	0.189544%	651 secs
22	0.172132%	667 secs
23	0.144736%	684 secs
24	0.142543%	700 secs
25	0.131688%	715 secs
26	0.0742733%	730 secs
27	0.0632109%	747 secs
28	0.0574532%	766 secs
29	0.0568549%	782 secs
30	0.0147081%	798 secs
31	0.0117892%	813 secs
32	0.0114488%	829 secs
33	0.0102172%	847 secs
34	0.00946591%	867 secs

Tab. 8: Évolution de l'écart entre borne supérieure et inférieure dans le temps au cours des itérations pour la versions ND++ sur une instance de 10 périodes et de 3 fils par nœud

Pb3.500	
ND	**ND++**
1320 secs	564 secs
11 itér	5 itér
Speedup : 57.27%	

Pb4.60	
ND	**ND++**
727 secs	321 secs
12 itér	5 itér
Speedup : 55.85%	

Pb5.20	
ND	**ND++**
810 secs	260 secs
15 itér	5 itér
Speedup : 67.9%	

Pb6.12	
ND	**ND++**
1050 secs	594 secs
9 itér	5 itér
Speedup : 43.43%	

Pb7.7	
ND	**ND++**
742 secs	632 secs
10 itér	7 itér
Speedup : 14.82%	

Pb7.8	
ND	**ND++**
12295 secs	1951 secs
18 itér	8 itér
Speedup : 84.13%	

Pb8.5	
ND	**ND++**
834 secs	568 secs
17 itér	8 itér
Speedup : 31.89%	

Pb10.4	
ND	**ND++**
1662 secs	748 secs
12 itér	5 itér
Speedup : 54.99%	

Pb11.4	
ND	**ND++**
3232 secs	1804 secs
12 itér	6 itér
Speedup : 44.18%	

Tab. 9: Comparaison des deux versions de décomposition sur une série d'instances où on ne pourrait pas appliquer CPLEX.

Itér	Recherche	Évaluation	IACSD	Écart
1	32 secs	7 secs	61 secs	40.47%
2	20 secs	4 secs	42 secs	24.70%
3	5 secs	7 secs	43 secs	6.97%
4	5 secs	7 secs	70 secs	5.38%
5	5 secs	6 secs	56 secs	3.51%
6	-	-	64 secs	1.06%
7	-	-	59 secs	1.01%
8	-	-	46 secs	0.53%

Tab. 10: Comparaison des temps de calcul demandés par itération pour ND++ pour chacune des trois étapes de la stratégie de choix de solutions candidates sur l'instance Pb8.5. IACSD : Itération sur l'arbre complet avec une solution de départ.

Dans la première colonne du tableau 10 on observe que les temps de calcul diminuent au cours des itérations. Ceci est normal quand on se rappelle que les coupes propres aux nœuds ne sont pas retirées lors de la recherche des solutions, ce qui a comme résultat le fait que quelques itérations plus tard l'algorithme commencera déjà avec des informations (coupes) supplémentaires et donc il convergera plus rapidement qu'avant. Le tableau 10 se réfère à une instance de 8 périodes où on branche 5 fois par nœud. Les sous-arbres destinés à la recherche ont un facteur de branchement de 2, tandis que le sous-arbre sur lequel s'effectue l'évaluation des solutions a un facteur de branchement de 3. Les remarques suivantes s'imposent :

- Dans la colonne *Recherche* on effectue une série d'itérations de la méthode de décomposition imbriquée alors que dans les colonnes de *Évaluation* et de *IACSD* on en effectue juste une.
- Dans la troisième colonne on effectue une seule itération sur l'arbre de degré 3, alors que dans la quatrième colonne on effectue une itération sur l'arbre de degré 5. Il est évident dans le tableau comment le temps de calcul augmente en rajoutant deux branches de plus par nœud.
- Le facteur de branchement sur les sous-arbres de recherche des solutions peut être différent, de préférence même inférieur, à celui des sous-arbres d'évaluation. En effet, on ne souhaite pas passer beaucoup de temps sur la recherche des solutions, puisque de toute façon un tel choix est à priori heuristique. C'est la raison pour laquelle on pourrait même se permettre, sur les grands sous-arbres, d'arrêter le calcul non pas quand l'écart tombe en dessous de 1% mais même en dessous de 5%. De toute manière, l'optimalité de ces solutions n'a pas d'importance majeure.
- Le facteur de branchement sur l'arbre d'évaluation des solutions est lié à la précision de l'évaluation. Plus ce facteur de branchement se rapproche du facteur de branchement de l'arbre réel, plus l'évaluation de la solution candidate est précise.

Si on arrive à maintenir à de bas niveaux le temps passé à la recherche et à l'évaluation des solutions, alors la réduction du nombre d'itérations jusqu'à la convergence aura un impact direct sur la réduction du temps de calul. Dans la figure 11 on

juxtapose la convergence entre borne supérieure et inférieure pour chacune de deux méthodes : la ND et la ND++.

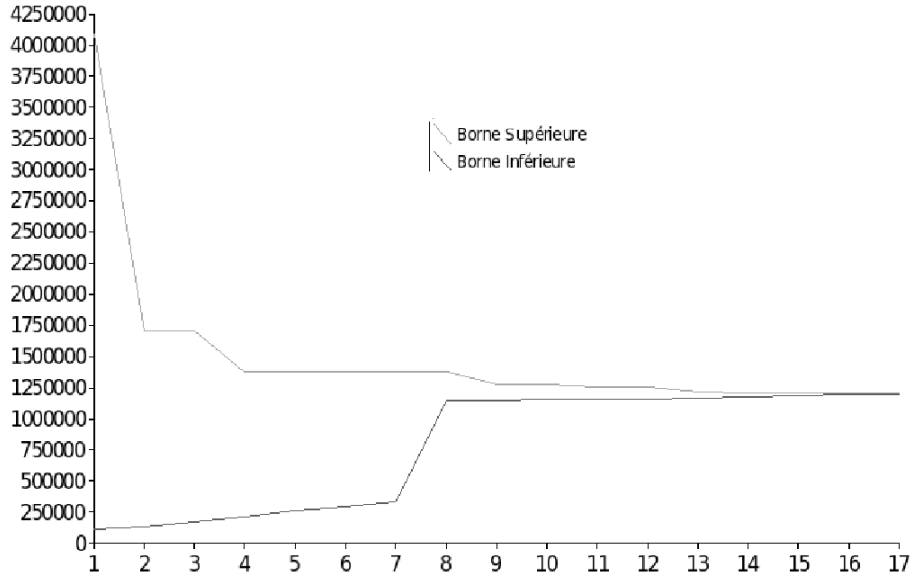
On conclut ce paragraphe en rajoutant les figures 12-17 qui mettent en évidence la supériorité de la ND++ contre la ND en terme de convergence au cours des itérations sur plusieurs instances.

5.8 Bilan global des améliorations proposées

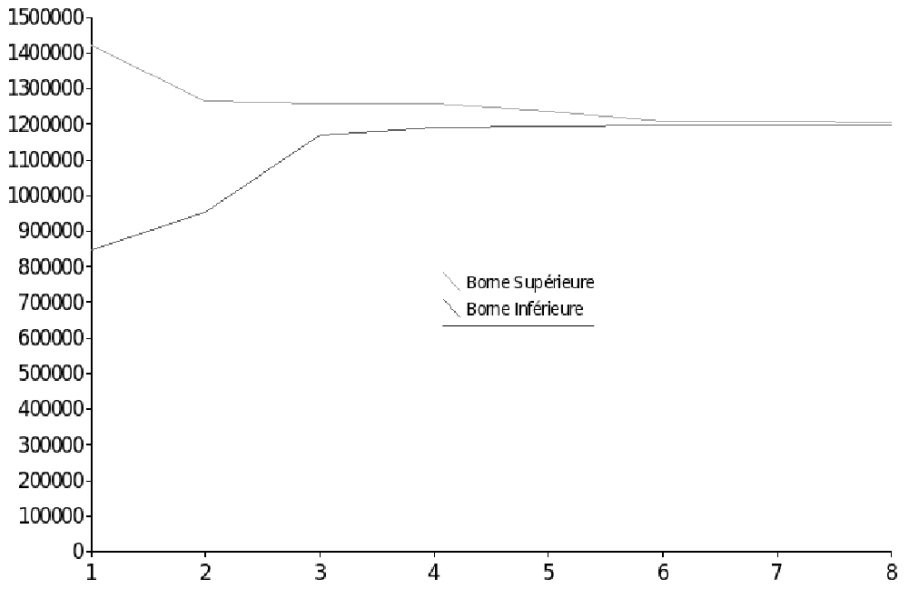
Lors des expériences présentées au paragraphe 5.7.2 concernant la réduction du nombre d'itérations, on a inclus dans la décomposition imbriquée standard la mise en œuvre du protocole hybride de coupes, dont les résultats ont été présentés au paragraphe 5.7.1. Par ailleurs, lors des expériences concernant la mise en œuvre du protocole hybride de coupes, le choix de solutions candidates n'a pas été considérée. Avant de conclure ce chapitre, il serait utile de mesurer, de façon globale, l'impact des améliorations que l'on a proposées sur le temps de calcul. Les expériences se sont produites sur la seconde série d'instances (cf au paragraphe 5.7.2) où aucun logiciel d'optimisation ne pourrait être appliqué.

Le tableau 18 met en évidence la différence entre la mise en œuvre des améliorations proposées aux derniers paragraphes et la version de la décomposition imbriquée normale. La colonne "ND" du tableau représente alors le temps de calcul de la décomposition imbriquée standard, pour les deux protocoles de coupes : coupes agrégées et multi-coupes. La colonne "ND++" représente le temps de calcul de la décomposition imbriquée fonctionnant avec le protocole hybride de coupes et le choix de solutions de départ La figure 19 fait visualiser le gain accumulé par les améliorations que l'on a proposées par rapport à la méthode de décomposition imbriquée standard fonctionnant avec le protocole de coupes agrégées (puisque c'est le protocole qui semble être le plus performant sur cette série de grandes instances). L'amélioration effective est constamment supérieure à un facteur de 2 (deux fois plus rapide) avec une moyenne supérieure à un facteur de 3.

Le tableau 20 montre les dimensions des plus grands problèmes que l'on a résolus, le temps de calcul, ainsi que l'amélioration globale apportée par notre implémentation.



(a) Convergence ND



(b) Convergence ND++

Fig. 11: Comparaison de la convergence entre les deux méthodes pour l'instance Pb8.5.

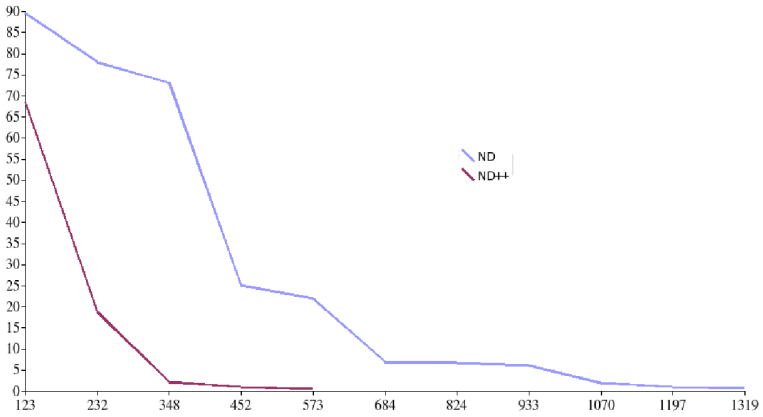


Fig. 12: Pb3.500 – Évolution de l'écart des bornes en fonction du temps.
Rouge : ND – Bleu : ND++

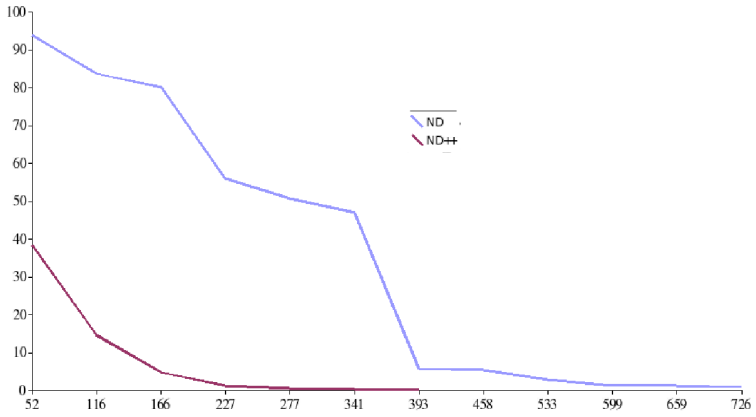


Fig. 13: Pb4.60 – Évolution de l'écart des bornes en fonction du temps.
Rouge : ND – Bleu : ND++

5.9 Conclusions

Le problème de dimensionnement qui nous a intéressé possède une structure particulière ; une structure qui met en évidence des couplages entre les variables fixées à la première étape et les variables affectées aux étapes ultérieures. Sur l'application qui nous intéresse, il s'agit des variables de capacités, présentes aux contraintes de tous les nœuds.

L'idée de la méthode que l'on a proposé est de mettre en cause le fonctionnement

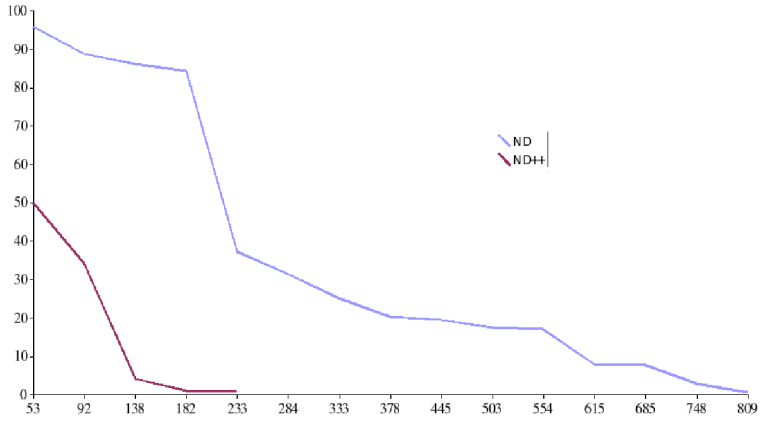


Fig. 14: Pb5.20 – Évolution de l'écart des bornes en fonction du temps.
Rouge : ND – Bleu : ND++

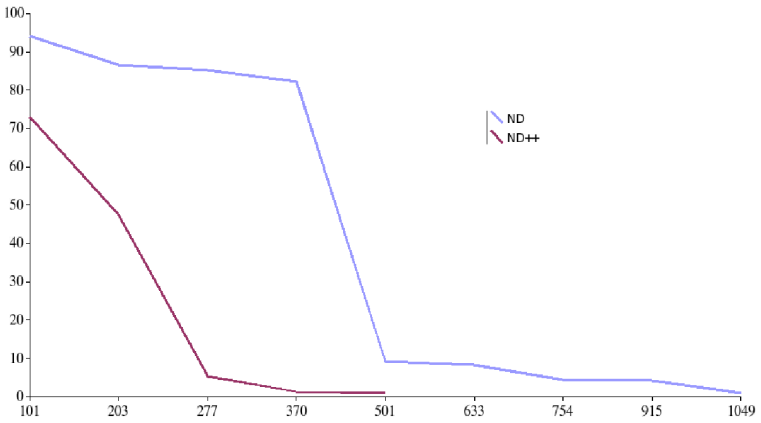


Fig. 15: Pb6.12 – Évolution de l'écart des bornes en fonction du temps.
Rouge : ND – Bleu : ND++

de l'algorithme sur le choix des solutions de départ (solutions sur les capacités), et de consacrer un peu plus de temps en essayant de créer un ensemble de solutions potentiellement performantes, ensuite choisir la meilleure solution de cet ensemble pour l'utiliser enfin comme solution fixée de variables du nœud-racine. Afin de rechercher ainsi que d'évaluer la qualité de ces solutions, on a recouru à l'échantillonnage conditionnel de plusieurs sous-arbres, qui maintiennent la complexité des scénarios de départ (scénarios couplés, non-parallèles). Ces sous-arbres jouent le rôle du simulateur qui propose de bonnes solutions et ensuite il les estime. Les résultats sur plusieurs instances

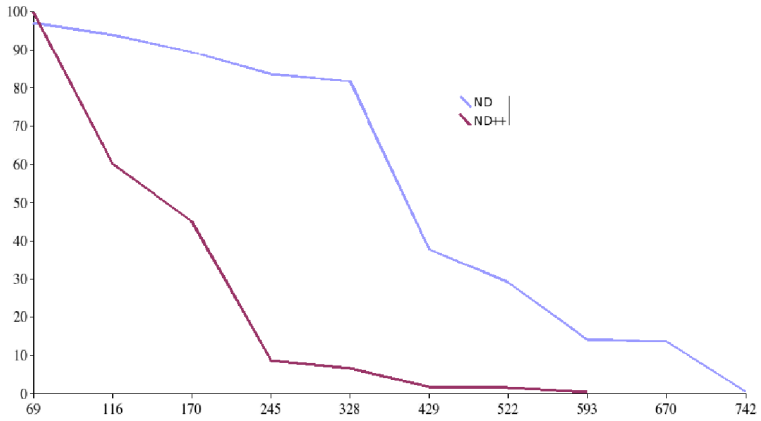


Fig. 16: Pb7.7 – Évolution de l'écart des bornes en fonction du temps.
Rouge : ND – Bleu : ND++

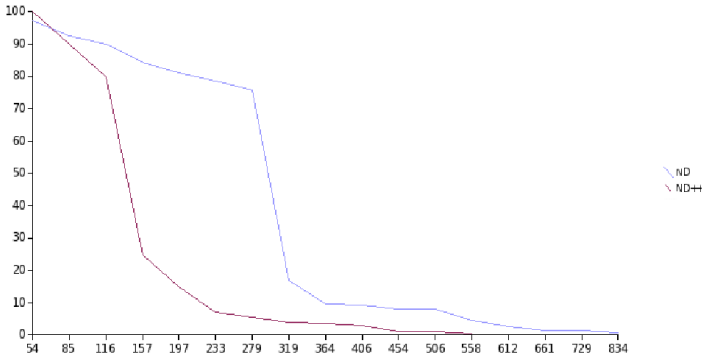


Fig. 17: Pb8.5 – Évolution de l'écart des bornes en fonction du temps.
Rouge : ND – Bleu : ND++

ont montré la supériorité de cette nouvelle mise en œuvre en comparaison avec la décomposition imbriquée normale sur des instances, ne pouvant pas être traitées par les logiciels d'optimisation de type CPLEX, à cause de leur grandes dimensions.

Une autre amélioration proposée porte sur le couplage entre les deux protocoles de construction de coupes : coupes agrégeés et multi-coupes. Appliquer les multi-coupes aux nœuds proches du nœud-racine augmente l'information nécessaire pour la convergence de l'algorithme. Appliquer des coupes agrégées aux étapes ultérieures accélère la résolution par itération. Les résultats ont montré que le protocole hybride hérite des avantages des deux protocoles et évite en grande partie leurs inconvénients.

Instance	**ND**		**ND++**
	Coupes agrégées	Multi-coupes	
Pb3.500	1714 secs	3700 secs	564 secs
Pb4.60	1055 secs	1755 secs	321 secs
Pb4.80	2952 secs	Mém.Insuf.	748 secs
Pb4.100	5257 secs	Mém.Insuf.	1804 secs
Pb5.20	868 secs	2035 secs	260 secs
Pb6.12	2448 secs	Mém.Insuf.	594 secs
Pb7.7	1556 secs	7150 secs	632 secs
Pb7.8	9813 secs	Mém.Insuf.	1951 secs
Pb8.5	1310 secs	4383 secs	568 secs

Tab. 18: Comparaison de la ND avec la ND++ sur la série d'instances que CPLEX 9.1 ne peut pas traiter. La ND est implémenté avec les deux protocoles de coupes. La ND++ est implémentée avec le protocole hybride de coupes et la stratégie sur le choix de solutions de départ.

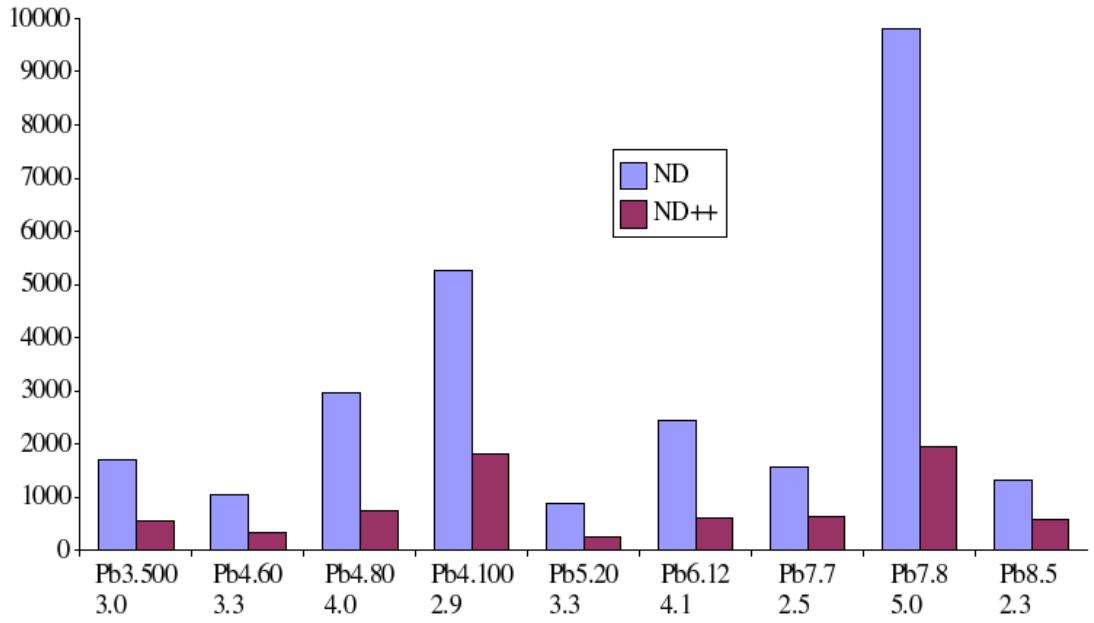


Fig. 19: Réduction globale du temps de calul en appliquant l'ensemble d'améliorations proposées.

Instances	**# nœuds**	**# contraintes**	**# variables**	**Temps ND++**	**Ratio**
Pb11.3	88,772	3,011,448	2,302,872	5382 secs	4.21
Pb8.5	97,655	3,320,270	2,539,030	568 secs	2.31
Pb5.20	168,420	5,726,280	4,378,920	260 secs	3.34
Pb4.60	219,660	7,468,440	5,711,160	321 secs	3.29
Pb3.500	250,500	8,517,000	6,513,000	564 secs	3.04
Pb6.12	271,452	9,229,368	7,057,752	594 secs	4.12
Pb7.8	299,592	10,186,128	7,789,392	1951 secs	5.03
Pb4.80	518,480	17,628,320	13,480,480	748 secs	3.95
Pb4.100	1,010,100	34,343,400	26,262,600	1804 secs	2.91

Tab. 20: Série des plus grandes instances résolues avec les nouvelles mises en œuvre de la ND proposées. Les dimensions en termes de nombre de nœuds, nombre de contraintes et de variables sont affichées. **Ratio** représente le ratio entre le temps de calcul de la ND (ND implémentée au meilleur – en termes de temps de calcul – protocole entre coupes agrégées et multi-coupes) et le temps de calcul de la méthode que l'on propose (ND++ avec le protocole hybride).

Conclusions – Perspectives

Le problème qui a motivé l'étude des modèles de recours multi-étapes a été une application réelle. L'objectif dans cette application, a été de souscrire un certain nombre de contrats avec plusieurs acteurs d'un marché gazier, de manière à ce que le coût des capacités souscites, ainsi que le coût de la gestion à suivre, soient minimum. Le problème, en sa version déterministe, se modélise en tant que programme linéaire et l'aléa est modélisé à travers un arbre de scénarios. Cette représentation de l'aléa maintient le caractère linéaire du problème, mais elle fait augmenter de façon exponentielle sa taille.

Dans le domaine des applications, il est couramment admis qu'une fois que l'on arrive à faire passer un modèle sur un logiciel d'optimisation du type CPLEX, (c'est-à-dire si la mémoire requise pour stocker la matrice technologique se trouve dans les capacités de la machine) le meilleur choix à faire serait de résoudre le problème avec ce logiciel en utilisant un algorithme de points intérieurs (comme par exemple Barrier). Que ce passe-il lorsque l'on n'arrive pas à faire passer la matrice de contraintes dans la mémoire de la machine ? Une solution serait de procéder à des méthodes de décomposition.

Une telle méthode est la décomposition imbriquée. Elle s'applique après avoir discrétisé l'espace de réalisations possibles, le résultat étant un arbre de scénarios qui représente tout évènement possible qui pourrait se produire dans l'avenir. L'arbre de scénarios est constitué de nœuds auquels on associe des problèmes linéaires. La méthode de décomposition imbriquée a été également appelée dans la littérature (par exemple [53]) *méthode de programmation dynamique stochastique duale*, parce que l'idée est d'approximer la fonction de recours (*fonction "cost-to-go"*) par un ensemble de coupes

construites à partir des informations duales (solutions duales optimales) issues des problèmes des étapes ultérieures.

Plusieurs améliorations ont été déjà proposées et des résultats prometteurs sur des applications réeles ont été présentées. Deux méthodes qui traitent la réduction de la dimension du grand arbre de scénarios ont été rapportées à l'annexe A (Donohue & Birge [20] et Pereira & Pinto [53]), mais elles n'ont pas été utilisées à cause des hypothèses non-vérifiables qu'elles imposent (hypothèse d'indépendance des réalisations de variables aléatoires à travers les étapes). On a élaboré des nouvelles mises en œuvre de la décomposition imbriquée qui abordent le problème de la grande taille de l'arbre en couplant l'optimisation avec la simulation.

On a insisté sur la façon dont l'algorithme de la décomposition imbriquée choisit les solutions du nœud-racine. Les variables associées au nœud-racine apparaissent dans tout programme linéaire associé à n'importe quel nœud de l'arbre, mettant en évidence l'importance du choix des solutions associées à ces variables. En échantillonnant plusieurs sous-arbres, on a crée un ensemble de solutions potentiellement bonnes qu'ensuite on s'est efforcé d'évaluer.

La construction d'un ensemble de bonnes solutions de départ (solutions à imposer au nœud-racine) est une tâche heuristique. On s'est basé sur deux critères pour qualifier ces *"bonnes solutions"* : la solution optimale d'un sous-arbre et la solution qui a produit au cours de l'algorithme de la décomposition imbriquée (sur le même sous-arbre) la plus grande réduction entre borne inférieure et borne supérieure. Ceux-ci sont des critères heuristiques. Une bonne solution de départ est la solution qui produit à l'issue du passage vers l'avant et vers l'arrière sur l'arbre complet la plus forte réduction sur l'écart des deux bornes. Cette solution n'est peut-être pas impérativement une solution optimale (comme on a suggéré) d'un arbre échantillonné. Ce pourrait également être une très mauvaise solution en termes de coût qui est pourtant associée à un point de la fonction de recours, dont l'évaluation permettrait le rapprochement des deux bornes. Le sujet de la recherche de solutions sur lequels on approxime bien la fonction de recours est un sujet ouvert à discussion.

Des aspects portant sur un meilleur échantillonnage de l'arbre complet restent également à explorer. Des sous-arbres équilibrés ne sont pas forcément le meilleur

échantillon que l'on puisse générer. Un arbre qui branche vers des directions de plus grande incertitude plutôt que vers d'autres directions est une possibilité envisageable. Dempster et coauteurs ([30] et [29]) donnent plusieurs nouvelles idées intéressantes sur la manière d'utiliser l'information issue de l'EVPI (valeur espérée d'information parfaite, cf au paragraphe 1.8) à chaque nœud, comme critère de stochasticité. Pour mettre en œuvre ces idées dans le contexte de notre méthode, on pourrait considérer que lors des passages vers l'avant et vers l'arrière sur les arbres échantillonné ou même sur l'arbre complet, un nœud avec une valeur faible d'EVPI ne présente aucun intérêt d'être finement exploré ; au contraire, une relaxation de la non-anticipativité ou un faible échantillonnage à partir de ce nœud aurait été suffisant pour approximer la fonction de recours.

Dans le protocole hybride de coupes agrégées et de multi-coupes que l'on a proposé, il y a un choix à faire sur les nœuds où on applique le premier ou le second protocole. Les caractéristiques qu'un nœud devrait avoir afin de lui appliquer tel ou tel protocole de coupes mériterait d'être étudiées. L'EVPI pourrait éventuellement être utilisée comme critère. Un nœud avec une valeur faible d'EVPI, donc un nœud peu stochastique ne nécessiterait pas forcément une description aussi fine que celle apportée par la version multi-coupes ; il serait plus économique de se contenter à des coupes agrégées concernant ce nœud.

La méthode de décomposition imbriquée est déjà parallélisable. Le premier code qui implémetait la décomposition imbriquée a été écrit par Gassmann dans [33]. Depuis, il existe bien des tentatives de parallélisation dans la littérature dont quelques travaux sont [21], [46] ainsi que [30] ou [29].

La stratégie de choix de solutions candidates (ce que l'on a appelé ND++), elle aussi, est parallélisable. Le travail de deux premières phases, celle de la recherche des solutions et celle de l'évaluation de ces dernières pourrait être affecté à plusieurs processeurs. La figure 21 met en évidence la décomposabilité de la méthode. La partie "Itération ND" désigne la phase du passage vers l'avant et vers l'arrière sur l'arbre complet, en ayant comme solution de départ la solution issue de l'étape de "l'Évaluation". On se place du point de vue de notre méthode, et c'est pour cela que l'on a affecté un seul processeur, or cette phase pourrait s'effectuer également en parallèle, puisque la

méthode ND est elle même parallélisable.

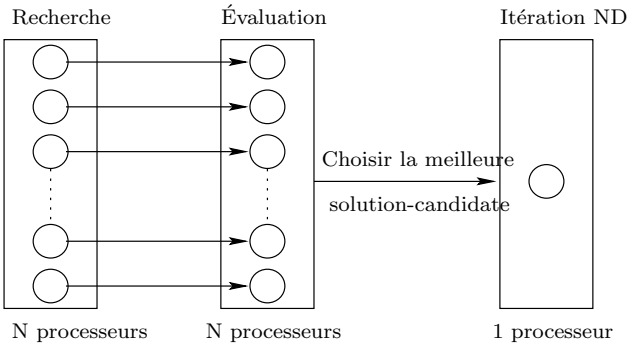


Fig. 21: Parallélisation de la stratégie proposée.

D'autres perspectives portant sur l'extension du modèle mathématique de l'application qui nous a intéressé sont également envisageables. On pourrait imaginer des contrats qui se négocient non pas une seule fois, mais régulièrement, à titre d'exemple toutes les t périodes. Ceci ne consiste pas en une extension propre à notre application, mais elle pourrait être suggérée pour n'importe quel autre problème de dimensionnement. Les variables du nœud-racine continueraient à jouer un rôle important dans la façon dont fonctionne la méthode que l'on a proposée, mais d'autres variables deviennent également importantes. L'ensemble de solutions candidates contiendrait désormais plusieurs variables de plusieurs nœuds appartenant à différentes périodes, et la simulation (le choix et l'évaluation des solutions sur les sous-arbres) devrait nous fournir non pas une seule solution pour le nœud-racine, mais autant de solutions que les nœuds correspondant aux décisions structurantes. Ceci consisterait en une généralisation de la méthode proposée. Nénanmois, on estime que l'impact d'une stratégie de choix et d'évaluation de solutions aurait été d'autant moins important que le nombre de nœuds où des décisions structurantes sont prises augmente.

Bibliographie

[1] H. Agarwal, J.E. Renaud, E.L. Preston, and D. Padmanabhan. Uncertainty quantification using evidence theory in multidisciplinary design optimization. *Reliability Engineering and Systems Safety*, 2004.

[2] E. Balas, S. Ceria, and G. Cornuéjols. A lift-and-project algorithme for mixed 0-1 programs. *Mathematical Programming*, 58 :295–324, 1993.

[3] J. Balasubramanian and I.E. Grossmann. Approximation to multistage stochastic optimization in multiperiod batch plant scheduling. Department of Chemical Engineering, Carnegie Mellon Universi, 2003.

[4] E.M.L. Beale. On minimizing a convex function subject to linear inequalities. *J. Roy. Statist. Soc.*, 17, 1955.

[5] A. Ben-Tal and A. Nemirovski. Robust convex optimization. *Mathematics of Operations Research*, 23 :769–805, 1998.

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

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

[8] P. Beraldi, L. Grandinetti, R. Musmanno, and C. Triki. Parallel algorithms to solve two-stages stochastic linear programs with robustness constraints. *Parallel Computing*, 26 :1889–1908, 2000.

[9] B. Bereanu. On the distribution of the optimum in stochastic linear programming. *An. Univ. Bucuresti, Ser. Sti. Natur., Mat.-Mec. 14, No.2, 41-47*, 1965.

[10] B. Bereanu. On stochastic linear programming distribution problems, stochastic technology matrix. *Z. Wahrsch. theorie u. verw. Geb.*, 8 :148–152, 1967.

[11] B. Bereanu. The continuity of the optimum in parametric programming and applications to stochastic programming. *J. Optimization Theory Appl.*, 18(3) :319–333, 1976.

[12] Bernard Bereanu. On some distribution-free, optimal solutions/bases, in stochastic linear programming. *Rev. Roumaine Math. Pures Appl.*, 21(6) :643–657, 1976.

[13] Arjan Berkelaar, Cees Dert, Bart Oldenkamp, and Shuzhong Zhang. A primal-dual decomposition-based interior point approach to two-stage stochastic linear programming. *Oper. Res.*, 50(5) :904–915, 2002.

[14] Arjan Berkelaar, Roy Kouwenberg, and Zhang Shuzhong. A primal-dual decomposition algorithm for multistage stochastic convex programming. 2000.

[15] John R. Birge. Decomposition and partitioning methods for multistage stochastic linear programs. *Operations Research*, 33(5) :989–1007, 1985.

[16] John R. Birge. Stochastic programming computation and applications. *INFORMS J. Comput.*, 9(2) :111–133, 1997.

[17] John. R. Birge. Using stochastic programming problem structure to gain computational efficiency. Technical report, Northwestern University, 2003.

[18] John R. Birge and François Louveaux. *Introduction to stochastic programming.* Springer-Verlag, New York, 1997.

[19] John R. Birge and François V. Louveaux. A multicut algorithm for two-stage stochastic linear programs. *European J. Oper. Res.*, 34(3) :384–392, 1988.

[20] J.R. Birge and C.J. Donohue. An upper bound on the expected value of a non-increasing convex function with convex marginal return functions. *Operations Research Letters 18 :213-221*, 1996.

[21] J.R. Birge, C.J. Donohue, D.F. Holmes, and O.G. Svintsitski. A parallel implementation of the nested decomposition algorithm for multistage stochastic linear programs. *Mathematical Programming*, 75(2) :327–352, 1996.

[22] Jörgen Blomvall. A multistage stochastic programming algorithm suitable for parallel computing. *Parallel Computing*, 29 :431–445, 2003.

[23] Jörgen Blomvall and Per Olov Lindberg. A Riccati-based primal interior point solver for multistage stochastic programming—extensions. *Optim. Methods Softw.*, 17(3) :383–407, 2002. Stochastic programming.

[24] Claus C. Carøe and Jørgen Tind. L-shaped decomposition of two-stage stochastic programs with integer recourse. *Math. Programming*, 83(3, Ser. A) :451–464, 1998.

[25] A. Charnes, W.W. Cooper, and G.H. Symonds. Cost horizons and certainty equivalents : An approach to stochastic programming of heating oil. *Management Science*, 4 :183–195, 1958.

[26] Zhi-Long Chen and Warren B. Powell. A convergent cutting-plane and partial-sampling algorithm for multistage stochastic linear programs with recourse. Technical Report SOR-97-11, Department of Civil Engineering and Operations Research, Priceton University, Priceton, NJ 08544, 1998.

[27] G.B. Dantzig. Linear programming under uncertainty. *Management Science*, 1 :197–206, 1955.

[28] M.A.H. Dempster and A. Papagaki-Papoulias. Computational experience with an approximate method for the distribution problem. In *Stochastic programming, Proc. int. Conf., Oxford 1974, 223-243*, 1980.

[29] M.A.H. Dempster and R.T. Thompson. Parallelization and aggregation of nested benders decomposition. *Annals of Operations Research*, 81 :163–187, 1998.

[30] M.A.H. Dempster and R.T. Thompson. Evpi-based importance sampling solution procedures for multistage stochastic linear programmes on parallel mimd architectures. *Annals of Operations Research*, 90 :161–184, 1999.

[31] K. Dixit, Avinash and Robert S. Pindyck. *Investment under Uncertainty*. Princeton University Press, 1994.

[32] C.J. Donohue and J.R. Birge. The abridged nested decomposition method for multistage stochastic linear programs with relatively complete recourse. *Algorithmic Operations Research*, 1 :20–30, 2006.

[33] H. Gassmann. MSLIP, a computer code for the multistage stochastic linear programming problem. *Mathematical Programming*, 47 :407–423, 1990.

[34] Jacek Gondzio and Roy Kouwenberg. High-performance computing for asset-liability management. *Operations Research*, 49(9) :879–891, 2001.

[35] J.L. Higle and S. Sen. Stochastic decomposition : A statistical method for large scale stochastic linear programming. *Kluwer Academic Publishers, Dordrecht*, 1996.

[36] Julia L. Higle and Suvrajeet Sen. Stochastic decomposition : an algorithm for two-stage linear programs with recourse. *Mathematics of Operations Research*, 16(3) :650–669, 1991.

[37] P. Kall and S.W. Wallace. *Stochastic Programming*. Wiley, Chichester etc., 1994.

[38] A.J. King. *Asymptotic Behaviour of Solutions in Stochastic Optimization : Nonsmooth analysis and the Derivation of Non-Normal Limit Distributions*. PhD thesis, University of Seatle, New York, 1986.

[39] Willem K. Klein Haneveld and Maarten H. van der Vlerk. Stochastic integer programming : general models and algorithms. *Ann. Oper. Res.*, 85 :39–57, 1999. Stochastic programming. State of the art, 1998 (Vancouver, BC).

[40] Willem K. Klein Haneveld and Maarten H. van der Vlerk. Stochastic integer programming : general models and algorithms. *Ann. Oper. Res.*, 85 :39–57, 1999. Stochastic programming. State of the art, 1998 (Vancouver, BC).

[41] W.K. Klein Haneveld, L. Stougie, and M.H. van der Vlerk. Stochastic integer programming with simple recourse. Research Memorandum 455, Institute of Economic Research, University of Groningen, 1991.

[42] Anton J. Kleywegt, Alexander Shapiro, and Tito Homem-de Mello. The sample average approximation method for stochastic discrete optimization. *SIAM J. Optim.*, 12(2) :479–502 (electronic), 2001/02.

[43] G.J. Klir and R.M. Smith. On measuring uncertainty and uncertainty-based information : recent developments. *Annals of Mathematics and artificial Intelligence*, 2001.

[44] P.E. Kloeden and E. Platen. *Numerical solution of stochastic differential equations*. Springer-Verlag, Berlin, 1992.

[45] G. Laporte and F.V Louveaux. The integer l-shaped method for stochastic integer programs. *Operations Research Letters*, 13 :48–62, 1993.

[46] Jeff Linderoth and Stephen Wright. Decomposition algorithms for stochastic programming on a computational grid. *Computational Optimization and Applications*, 24, 2003.

[47] Irvin J. Lustig, John M. Mulvey, and Tamra J. Carpenter. Formulating two-stage stochastic programs for interior point methods. *Oper. Res.*, 39(5) :757–770, 1991.

[48] H. Markowitz. Portofolio secection. *Journal of Finance*, 7 :77–91, 1952.

[49] J.M. Mulvey and B. Shetty. Financial planning via multi-stage stochastic optimization. *Computers and Operations Research*, 31 :1–20, 2004.

[50] John M. Mulvey and Andrzej Ruszczyński. A diagonal quadratic approximation method for linear multistage stochastic programming problems. *Operations Reseasrch Letters*, 12 :205–215, 1992.

[51] John M. Mulvey, Robert J. Vanderbei, and Stavros Zenios. Robust optimization of large-scale systems. *Operations Research*, 43(2), 1995.

[52] W.L. Oberkampf, J.C. Helton, and K. Sentz. Mathematical representation of uncertainty. In *Non-deterministic Approaches Forum, AIAA*.

[53] M.V.F. Pereira and L.M.V.G. Pinto. Multi-stage stochastic optimization applied to energy planning. *Math. Programming*, 52(2, Ser. B) :359–375, 1991.

[54] Andràs Prékopa. Logarithmic concave measures with application to stochastic programming. *Acta Sci. Math. (Szeged)*, 32 :301–316, 1971.

[55] Andràs Prékopa. *Stochastic Programming*. Kluwer Academic Publishers, Dordrecht, 1995.

[56] A.H.G. Rinnooy Kan. Stochastic integer programming : the distribution problem. In *Stochastic programming (Gargnano, 1983)*, volume 76 of *Lecture Notes in Control and Inform. Sci.*, pages 140–150, Berlin, 1986. Springer.

[57] R.T. Rockafellar and Roger J.-B. Wets. A dual solution procedure for quadratic stochastic programs with simple recourse. In *Numerical methods (Caracas, 1982)*, volume 1005 of *Lecture Notes in Math.*, pages 252–265. Springer, Berlin, 1983.

[58] R.T. Rockafellar and Roger J.-B. Wets. Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of Operations Research*, 16(1) :119–147, 1991.

[59] A. Ruszczyński. A regularized decomposition method for minimizing a sum of polyhedral functions. *Mathematical Programming*, 35 :309–333, 1986.

[60] A. Ruszczyński and A. Shapiro. Optimization of convex risk functions. *Mathematics of Operations Research*, 31 :433–452, 2006.

[61] A. Ruszczyński and A. Świ˛tanowski. Accelerating the regularized decomposition method for two stage stochastic linear problems. *European Journal of Operational Research*, 101 :328–34, 1997.

[62] Andrzej Ruszczyński. Augmented lagrangian decomposition for sparse convex optimization. Working Paper WP-92-75, IIASA - International Institute for Applied Systems Analysis, Laxenburg, Austria, 1992.

[63] Andrzej Ruszczyński. On augmented lagrangian decomposition methods for multistage stochastic programs. Working paper WP-94-05, IIASA - International Institute for Applied Systems Analysis, Laxenburg, Austria, 1994.

[64] Andrzej Ruszczyński and Alexander Shapiro, editors. *Stochastic Programming*, volume 10 of *Handbooks in Operations Research and Management Science*. Elsevier Science, Amsterdam, 2003.

[65] Kataoka S. A stochastic programming model. *Econometrica*, 31 :181–196, 1963.

[66] N. Sahinidis. Optimization under uncertainty : state-of-the-art and opportunities. Technical report, Computers and Chemical Engineering, 2003.

[67] Rüdiger Schultz. Continuity properties of expectation functions in stochastic integer programming. *Math. Oper. Res.*, 18(3) :578–589, 1993.

[68] Alexander Shapiro. Inference of statistical bounds for multistage stochastic programming problems. *Mathematical Methods of Operations Research*, 58 :57–68, 2003.

[69] S Stephanopoulos and W. Westerberg. "the use of hestenes" method of multipliers to resolve dual gaps in engineering system optimization. *J. Optim Theory Appl.*, 15 :285–309, 1975.

[70] Sen Suvrajeet. *Encyclopedia of OR/MS*, chapter Stochastic Programming : Computational Issues and Challenges.

[71] K. Tammer. On the solution of the distribution problem of stochastic programming. In *Progress in operations research, Vols. I, II (Proc. Sixth Hungarian Conf., Eger, 1974)*, pages 907–920. Colloq. Math. Soc. János Bolyai, Vol. 12, Amsterdam, 1976. North-Holland.

[72] Maarten H. van der Vlerk. Stochastic programming bibliography. World Wide Web, http ://mally.eco.rug.nl/spbib.html, 1996-2003.

[73] R. Van Slyke and R.J.B Wets. L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal on Applied Mathematic*, 17 :638–663, 1969.

[74] Bram Verweij, Shabbir Ahmed, Anton Kleywegt, George Nemhauser, and Alexander Shapiro. The sample average approximation method applied to stochastic routing problems : A computational study. Optimization Online, urlhttp ://www.optimization-online.org, 2001.

[75] Hercules Vladimirou and Stavros A. Zenios. Stochastic linear programs with restricted recourse. *European Journal of Operations Research*, 101 :177–192, 1997.

[76] S.R. Watson and M. Buede D. *Decision Synthesis. The Principles and Practice of Decision Analysis*. 1987.

[77] R.J.B Wets. Programming under uncertainty : the equivalent convex program. *SIAM Journal on Applied Mathematic*, 14 :89–105, 1966.

[78] Roger J.-B. Wets. Solving stochastic programs with simple recourse. *Stochastics*, 10(3-4) :219–242, 1983.

[79] X. Xu, P.F Hung, and Y. Ye. A simplified homogeneous self-dual linear programing algorithm and its implementations. *Annals of Operations Research*, 62 :151–171, 1996.

[80] W.T. Ziemba. Stochastic programs with simple recourse. In P.L. Hammer and G. Zoutendijk, editors, *Mathematical Programming in Theory and Practice*, pages 213–273. North-Holland Publishing Company, Amsterdam, 1974.

Annexe A

Échantillonnage au sein de la décomposition imbriquée

Il y a principalement deux travaux intéressants qui abordent l'échantillonnage de nœuds dans le contexte de la méthode de décomposition imbriquée : [53] et [32].

Après avoir observé les étapes de la décomposition imbriquée (cf au paragraphe 5.3) : il y a un passage vers l'avant à l'issue duquel une borne supérieure est calculée et un passage vers l'arrière à l'issue duquel une borne inférieure est établie. Pereira & Pinto [53] et Donoue & Birge [32] proposent d'échantillonner des nœuds pendant le passage vers l'avant, la conséquence étant que le calcul de la borne supérieure est statistique. Dans la suite on tentera d'esquisser chacune de ces deux méthodes. Les deux méthodes font deux hypothèses : (i) l'hypothèse de recours complet et (ii) l'hypothèse d'indépendance des variables aléatoires à travers les périodes. On expliquera brièvement ce que cela signifie. La présentation de deux méthodes suivra ultérieurement.

Variables indépendantes par période

La figure 1 montre le cas où la distribution de la variable aléatoire (ou des variables aléatoires) à la période t n'est pas conditionnée par la distribution de la période précédente t–1. La méthode de Pereira & Pinto est de Donoue & Birge sont basées sur cette hypothèse.

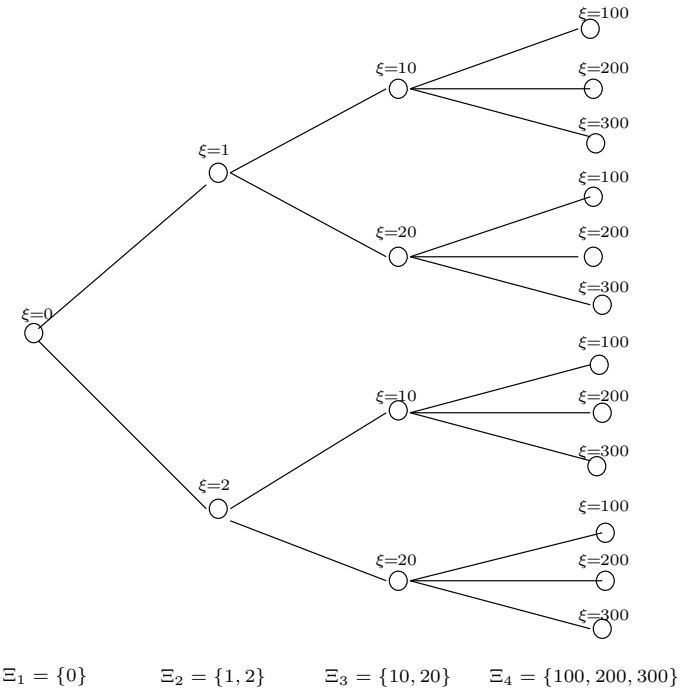


Fig. 1: Un arbre dont les réalisations des variables aléatoires sont indépendantes à travers les périodes.

Remarquons que l'hypothèse sur l'indépendance des variables aléatoires à travers les périodes est une très forte hypothèse qui simplifie drastiquement l'algorithme de décomposition. Plus précisément ceci signifierait que toute coupe généréé à la période t pour une solution x_t est valable pour tous les nœuds de cette période. Pour voir ceci il suffit de considérer deux nœuds n_1 et n_2 appartenant à la même période. Ils auront par définition les mêmes succeseurs ; si $\mathcal{S}_n$ désigne l'ensemble des nœuds successeurs au nœud n, on note $\mathcal{S}_{n_1} = \mathcal{S}_{n_2}$. Les coupes générées (que se soit des coupes agrégées ou des multi-coupes) pour n_1 ne contiendront que des coefficients issus des $n \in \mathcal{S}_{n_1}$. Comme $\mathcal{S}_{n_1}$=$\mathcal{S}_{n_2}$ ces coupes seront identiques à celles rajoutées au nœud n_2. Par conséquent, malgré le nombre de nœuds à la période t, les coupes décrivant la fonction de recours pour les intstances t'>t sont toutes identiques. Comme les coupes sont partagées par tous les nœuds, il n'y a plus besoin d'associer un programme linéaire à chaque nœud, mais plutôt un programme linéaire générique à chaque période. Ceci libère une grande

portion de mémoire engagée. À titre d'exemple sur un arbre de 6 périodes ou on branche 7 fois par période, au lieu de résoudre à la 5ème période 2401 sous-problèmes, on n'en résoud que 6.

La méthode de Pereira & Pinto

Soit un arbre $\mathcal{A}_T$ de T périodes avec un certain nombre de scénarios, comme par exemple celui affiché dans la figure 1. Dans la méthode de Pereira & Pinto [53] on effectue un passage vers l'avant sur un sous-arbre $\mathcal{A}'_T \subset \mathcal{A}_T$. Ce sous-arbre est généré comme suit : on décide combien de scénarios on souhaite générer, soit S (s désigne le compteur). À la première période on échantillonne S réalisations de la variable ξ_2 par tirage aléatoire depuis le support Ξ_2 (cf à la figure 1). Ensuite pour chaque réalisation ξ_t^s de la période t (parmi les S réalisations) on échantillonne une réalisation ξ_{t+1}^s depuis l'espace Ξ_{t+1}. La figure 2 montre un tel échantillonnage. Remarquons dans cette figure comment la réalisation ξ=1 à la période t=1 est échantillonnée dans les scénarios s=1, s=2 et s=3, mais les variables associées ne sont pas couplées. À l'issue de la phase d'échantillonnage on aura alors crée un sous-problème (puisque on a échantillonné le problème réel) auquel on a relaxé la non-anticipativité au delà de la première période. On effectue un passage vers l'avant sur ce sous-arbre.

À l'issue du passage vers l'avant sur l'arbre échantillonné on aura à notre disposition : (i) une borne supérieure statistique et (ii) un ensemble de "points de control" sur lesquels la fonction de recours sera approximée à chaque période lors du passage vers l'arrière.

L'estimation statistique sera la suivante :

$$\overline{z} = \frac{1}{S} \sum_s c_t^s x_t^s \tag{A.1}$$

où S est le nombre de scénarios générés dans l'arbre $\mathcal{A}'_T$. Ce coût correspond au sous-arbre $\mathcal{A}'_T$ qui représente un problème en deux étapes. *L'incertitude* dans cette estimation est mesurée par l'écart type :

$$\sigma_z = \sqrt{\frac{1}{S^2} \sum_s (\overline{z} - z_i)^2} \tag{A.2}$$

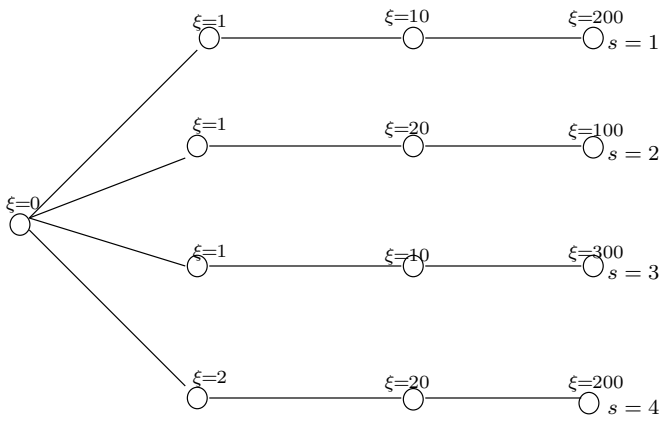


Fig. 2: Cet arbre est échantillonné à partir de l'arbre de la figure 1. On obtient un problème en deux étapes : la première étape est le nœud ξ=0 et la deuxième étape sont les scénarios parallèles.

où z_i est le coût de chaque scénario séparé. L'intervalle de confiance sur la borne supérieure est $[\overline{z} - 2\sigma, \overline{z} + 2\sigma]$.

Les solutions x_t^s que l'on aura calculées à l'issue du passage vers l'avant constitueront l'ensemble de décisions pour chaque période t. On a une décision x_0 à la première période qui est commune pour tous les scénarios et S décisions à chaque période t>0, $x_1^s, ..., x_T^s$, $\forall s$. La figure 3 montre la situation à l'issue du passage vers l'avant sur l'arbre échantillonné de la figure 2.

On est prêt à effectuer le passage vers l'arrière. L'indépendance des variables aléatoires simplifie grandement cette étape de l'algorithme : au lieu de traiter tous les nœuds de la période t (cf à la figure 1), on ne traite que les différentes réalisations ξ_t^s. On décrit formellement le passage vers l'arrière à la page 171. On rappelle que t=0 est la première période et donc la dernière est T–1 (le nombre de périodes étant T).

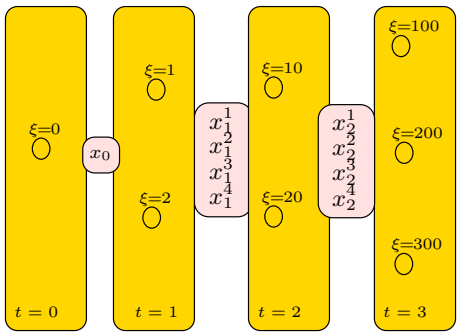


Fig. 3: Méthode de Pereira & Pinto : à l'issue de chaque passage vers l'avant, on dispose d'un nombre de solutions pour chaque période. Lors du passage vers l'arrière, ces solutions seront proposées aux nœuds de cette période et les coupes asscociées seront récupérées par les nœuds-pères. Comme les variables aléatoires sont indépendantes, les coupes ne sont plus associées aux nœuds mais aux différentes périodes.

Passage vers l'arrière

Après le passage vers l'avant, on dispose d'un ensemble de solutions $\{x_t^s\}$ pour chaque instant t et différents scénarios s :

– Pour toute période $t = T–1, T–2, ..., 1$:
  – Pour toute solution $x_t^s, \quad s = 1, 2, ..., S$:
    – Pour toute réalisation de la variable aléatoire à cette période $\xi_t^i, \forall i = 1, ..., I(t)$:
      Résoudre le problème linéaire :

$$\begin{aligned} & \min c_t x_t + \theta_t \\ \text{s.c} \quad & A_t x_t = \xi_t^i - B_{t-1} x_{t-1}^s \\ & \theta_t \in \Theta_t \end{aligned}$$

      Récupérer les variables duales optimales $\lambda_t^{s,i}$ et les valeurs de la fonction objectif $f_t^{s,i}$.
  – Calculer les coefficients pour la coupe à rajouter à la période $t–1$:

$$\pi_{t-1}^i = \sum_{s=1}^{S} p_t^i \lambda_t^{s,i} \quad , \quad \phi_{t-1}^i = \sum_{s=1}^{S} p_t^s f_t^{s,i}$$

Les précisions suivantes s'imposent :

- $I(t)$ est le nombre de réalisations de la variable aléatoire à l'instant t, c'est-à-dire la cardinalité de l'ensemble Ξ_t.
- Θ_t est le maximum point à point de la famille de coupes qui décrivent la fonction *"cost-to-go"*. Il est mis à jour à chaque itération.
- p_t^i est la probabilité conditionnelle de la réalisation i à la période t.
- Les variables duales optimales $\lambda_t^{s,i}$ se réfèrent à l'instant t, à la réalisation i, et qui ont été obtenues en utilisant la solution x_{t-1}^s de la période précédente. Les coefficients π sont obtenus en agrégeant les différents λ pour toute réalisation $i \in I(t)$. Il en va de même pour les coefficients ϕ.

À chaque période on effectue $I \cdot S$ résolutions des programmes linéaires. L'hypothèse sur l'indépendance des variables aléatoires empêche l'explosion exponentielle en fonction du nombre des périodes. Les coupes rajoutées à chaque itération à chaque période sont des coupes agrégées, mais la version multi-coupes pourait également être envisageable.

Le critère de terminaison est établi en comparant la borne inférieure $\underline{z} = f_0$ (f_0 est la valeur de la fonction objectif à l'instant 0) et la borne supérieure statistique $\overline{z}$ (A.1). Pratiquement, si $\underline{z}$ se trouve dans l'intervalle $\overline{z} - 2\sigma, \overline{z} + 2\sigma$, alors l'algorithme s'arrête.

La méthode a été appliquée sur un problème réel d'ordonnancement d'un générateur hydrothermique. Le modèle a été représenté par un arbre de scénarios en dix périodes où on branche deux fois par période. Le nombre de scénarios a été donc de 2^{10}=512. Lors du passage vers l'avant seulement 50 scénarios ont été générés et par conséquent le temps de calcul est très petit. En revanche, le passage vers l'arrière devient plus lourd, puisque à chaque réalisation de la période t on résout 50 fois le problème associé pour chaque solution x_t^s, s=1, ...S. Il faut par ailleurs noter qu'à la fin de chaque itération une seule coupe se rajoutera au nœud-racine, puisque il n'y a qu'une solution proposée aux nœuds de la deuxième période (cf à la figure 3).

Décomposition imbriquée "économique" (Abridged ND)

La validité de la borne (A.1) est basée sur le théorème centrale limite qui nécessite au minimum 30 scénarios échantillonnés. Ceci signifie qu'il a au moins 30 solutions à proposer à tous les nœuds de toutes les périodes (sauf la première), ce qui rend le passage vers l'arrière assez lourd surtout pour des problèmes où le nombre de réalisations de variables aléatoires à chaque période, ainsi que le nombre d'périodes sont importants.

Donohue et Birge [32] ont proposé une autre méthode d'échantillonnage au sein de la décomposition imbriquée. Ils ont fait, eux aussi, les hypothèses d'indépendance entre les variables aléatoires et de recours complet. La figure 4 illustre la méthode.

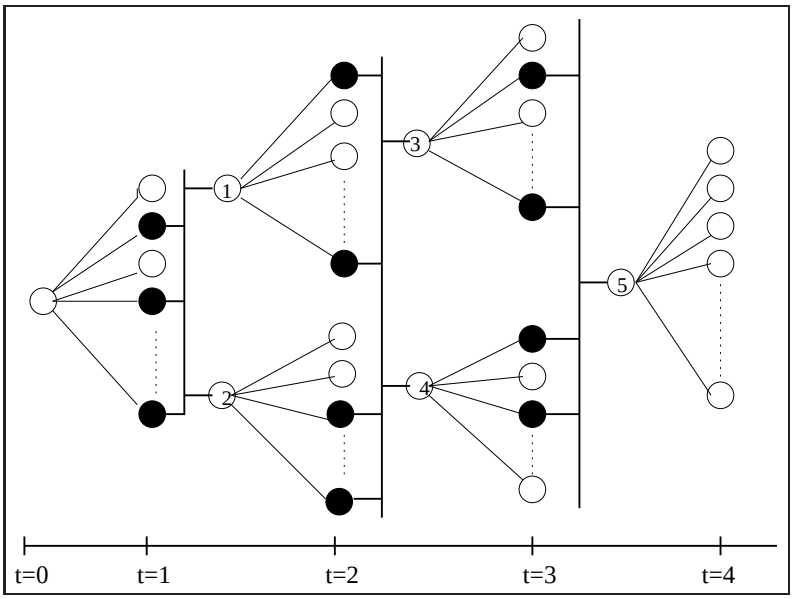


Fig. 4: Abridged ND : on échantillonne des réalisations de la variable aléatoire ainsi que des solutions de branchement que l'on fait propager aux instants suivants.

La nouvelle méthode porte le nom *"décomposition imbriquée économique"* et son nom vient du fait que l'arbre durant le passage vers l'avant est d'une certaine manière appauvri. L'idée de la méthode est la même que celle de la méthode de Pereira & Pinto précédemment montrée : lors du passage vers l'avant on cherche à obtenir une borne supérieure statistique et sourtout un ensemble de points sur lesquels on estimera la

fonction de recours à chaque période.

La différence par rapport à la méthode de Pereira & Pinto porte sur le passage vers l'avant. On échantillonne non seulement les réalisations de la variable aléatoire, mais aussi les solutions x_t^i pour différents nœuds que l'on propage à l'instant $t+1$. Les solutions que l'on fait propager aux périodes suivantes s'appellent *"valeurs de branchement"*.

On rappelle que l'on commence par $t=0$ et que l'horizon d'étude est atteint à $t=T-1$ (on a ainsi T périodes d'étude). D'après la figure 4, on commence en résolvant le nœud-racine. Ensuite, on échantillonne F_1 réalisations de la variable alétoire ξ_1 dont l'espace de réalisations est Ξ_1 (cf à la figure 1 pour illustration). Pour chaque realisation, on résout le programme linéaire qui ressort en transmettant la solution x_0 issue de la résolution du problème du nœud-racine. Parmi les F_1 solutions que l'on obtient à la période $t=1$ on en choisit $B_2 \leq F_1$ (dans l'exemple de la figure 4, $B_2=2$) comme valeurs de branchement. Il n'est pas obligatoire que les solutions faisant partie de B_2 soient issues des nœuds particuliers. Celles-ci peuvent constituer n'importe quelle combinaison de solutions issues des résolutions de différentes réalisations ξ_t^i de cette période. On rappelle l'hypothèse de recours complet qui garantit que pour toute solution de la période $t-1$ le problème de la période t soit réalisable.

Le passage vers l'arière s'effectue comme dans la méthode de Pereira & Pinto. Par conséquent, pour toute période t et toute solution x_t^i, $i=1, ..., B_t$ de la période t et pour toute réalisation on résout le programme linéaire associé, on récupère les solutions duales optimales que l'on utilise dans les coupes agrégées ou multi-coupes décrivant la fonction de recours. Le critère d'arrêt est le même que celui de la méthode de Pereira & Pinto.

La performance de la méthode a été vérifiée sur un ensemble de problèmes d'allocation dynamique de vehicules avec des demandes incertaines. Les résultats indiquent une amélioration nette par rapport à la méthode de Pereira & Pinto d'ordre de 3 à 20 concernant le temps de calcul.

Annexe B

Différentes manières d'échantillonner

Il est clair de la présentation de la méthode que l'on a propose au chapitre 5, que le rôle de l'échantillonnage n'est pas primordial dans la procédure, dans le sens où on ne se base pas sur un résultat issu de l'arbre échantillonné en prétendant que ceci constitue une solution statistiquement bonne pour l'arbre complet de scénarios. L'échantillonage ne se fait que pour nous fournir des solutions candidates, ainsi que pour nous permettre de évaluer rapidement chacune entre elles afin de choisir celle qui sera retenue pour participer comme solution de départ à une itération normale de la ND.

Même si l'échantillonnage ne joue pas un rôle crucial dans la méthode, un échantillonnage plus sophistiqué "*se tromperait*" moins sur les estimations portant sur la qualité de la solution de départ choisie et réduirait la marge d'erreur ; il réduirait typiquement les cas où parmi les dix solutions candidates, la plus performante est exclue.

Dans les expériences présentées au paragraphe 5.7.2, la recherche et l'évaluation des solutions candidates s'étaient faites sur des arbres échantillonnés de façon conditionnelle et équilibrée, ou autrement dit, sur des sous-arbres où on branche à chaque nœud d fois parmi les D branches. Dans une instance comme la Pb4.100 par exemple où d=100, on pourrait choisir un sous-arbre de d=10 ou de d=20 sans que le temps de

calcul mis sur la recherche et l'évaluation soit trop grand (moins de 10 secondes). Si le nombre de branches à chaque nœud est grand (typiquement plus de 5) on peut toujours choisir des sous-arbres où il n'est pas cher en termes de temps de calcul d'effectuer les deux premiers pas de l'algorithme. Si en revanche le nombre de périodes s'étend à 9 ou plus et le degré reste à 2, alors la situation change : le sous-arbre échantillonné de manière conditionnelle et équilibrée sera trop grand pour effectuer une recherche et une évaluation suffisamment vite.

Par exemple dans l'instance Pb12.2 un sous-arbre échantillonné ne peut qu'être un arbre de 12 périodes et d'un facteur de branchement de 1, c'est-à-dire un seul scénario. Or, travaillant sur un seul scénario comment pourrait-on chercher une solution sur les capacités et prétendre qu'une telle solution soit statistiquement bonne pour tous les scénarios, ou d'autre part, comment simuler un passage dans tout l'arbre sur la base d'un seul scénario ? On aurait également le choix d'échantillonner avec un facteur de branchement de 2. Si le sous-arbre généré était de facteur de branchement de 2, il s'agirait de l'arbre réel, la recherche et l'évaluation de solutions coûterait trop cher et ceci se traduirait à des itérations de la ND++ beaucoup plus longues que celles de la ND.

Une autre startégie d'échantillonnage serait sur l'espace de scénarios. On ne tire pas de façon aléatoire et progressive des nœuds, mais des feuilles de l'abre, en suivant leur chemin jusqu'à la racine. Or, échantillonner dans l'espace de scénarios ne donne aucune garantie que les scénarios générés formeront un arbre qui branchera à chaque période. Surtout quand le nombre de scénarios sur l'arbre est grand et les scénarios générés peu, l'ensemble de scénarios choisi sera probablement un ensemble des scénarios parallèles, ayant comme point commun la racine. Ceci ne capte pas la difficulté d'un problème multi-étapes et le résultat serait d'avoir une solution de départ éventuellement mal choisie et surtout mal estimée.

Une troisième manière d'échantillonner est de façon conditionnelle mais pas équilibrée. Il s'agirait donc de brancher avec un facteur de branchement de 2 à la première période, avec un facteur de branchement de 1 à la deuxième période, revenir à 2 branchement à la troisième période, puis à nouveau à 1 et ainsi de suite. Les figures 5 et 6 montrent ces trois façons d'échantillonner des sous-arbres à partir d'un arbre de

scénarios complet.

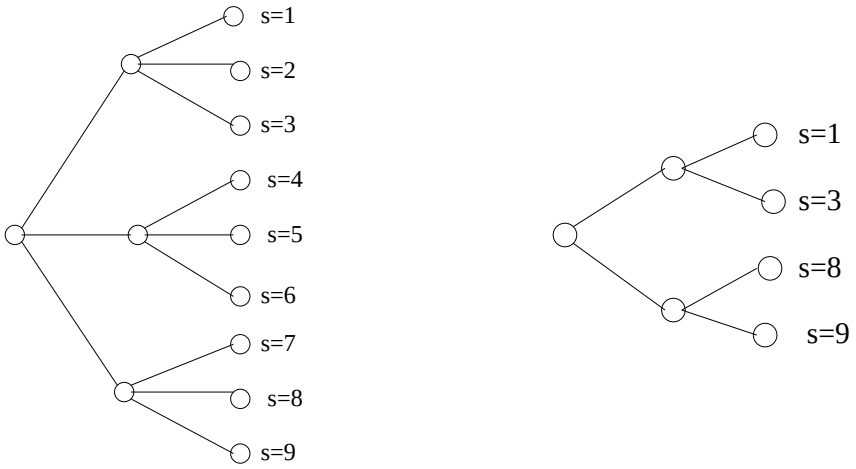


Fig. 5: Échantillonnage conditionnel et équilibrée : un facteur de branchement de 2 sur chaque période.

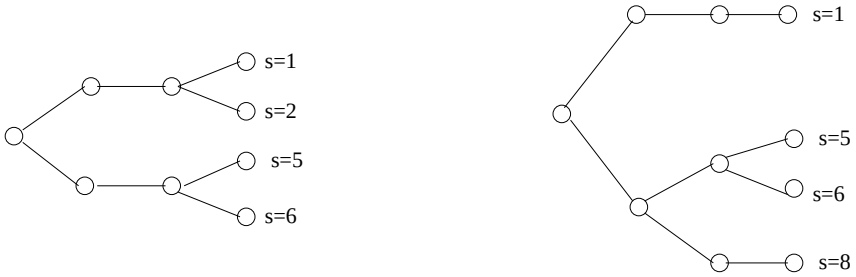


Fig. 6: Autre façons d'échantillonner : (en droite) échantillonnage conditionnel mais pas équilibré un facteur de branchement 2 sur la première période, un facteur de branchement 1 sur la deuxième période et un facteur de branchement 2 sur la troisième période – (en gauche) échantillonage sur l'espace de scénarios.

On a remarqué au cours des expériences que l'échantillonnage conditionnel équillibré s'est avéré plus efficace que les deux autres en termes de temps de calcul. Par ailleurs, les deux autres sont plus flexibles lorsque l'arbre complet est long avec un facteur de branchement de 2 (un arbre équilibré) où l'arbre de scénarios sur lequel on effectue la recherche et l'évaluation n'est pas de façon triviale qu'un seul scénario mais autant que l'on souhaite. Voici, dans le tableau 7 un comparatif entre un échantillonnage

d'un seul scénario et d'un écantillonnage où on branche 2 fois, puis une seule fois, puis 2 fois et ainsi de suite sur un arbre complet de 12 périodes avec un facteur de branchement de 2.

Instances	Sous-arbre 1	Sous-arbre 2
1	87 secs	62 secs
2	97 secs	67 secs
3	83 secs	60 secs

Tab. 7: Temps de calcul sur trois instances de type Pb12.2. Dans la première colonne le sous-arbre est constitué d'un seul scénario. Dans la seconde colonne le sous-arbre comporte plusieurs scénarios.

En général, l'échantillonnage conditionnel capte mieux la complexité d'un arbre qui branche à chaque nœud, contrairement à l'échantillonnage sur l'espace de scénarios. Par ailleurs, il n'est pas difficile à voir que dans le second type d'échantillonnage, pour le même nombre de scénarios, le nombre de nœuds obtenus dans le sous-arbre est plus important que dans le cas de l'échantillonnage conditionnel. On n'aurait par conséquent aucune raison d'utiliser un échantillonnage sur l'espace de scénarios, si le facteur de branchement sur l'arbre complet était plus grand que 2. On pourrait, d'autre part, montrer que l'échantillonnage conditionnel équilibré est l'échantillonnage qui produit le plus petit nombre de nœuds pour un nombre de scénarios donné par rapport aux autres types d'échantillonnage.

Annexe C

Résumé des dispositions légales applicables dans le secteur du gaz

Nous présentons dans cette annexe un aperçu de dispositions légales telles que définies dans le décret relatif aux obligations de service public dans le secteur du gaz.

C.1 Fournisseurs de gaz

Article 1 : Tous les fournisseurs du gaz naturel sont tenus d'avoir accès :

- à plusieurs sources d'approvisionnement diversifiées géographiquement ;
- au moins deux points d'entreé sur le réseau de transport national lorsque il s'approvisionnent plus de 10% ;
- au moins trois points d'entrée lorsque il s'approvisionnent plus de 20% du marché national.

Un point d'entrée s'entend comme un point d'interconnexion transfrontalier ou le lieu de raccordement à un site de production nationale ou une installation de gaz naturel liquéfié.

Article 2 : Les fournisseurs communiquent les règles et les modalités actuelles et prévisionnelles d'affectation de leurs ressources globales d'approvisionnement en gaz.

Article 3 : Les fournisseurs sont tenus d'assurer la fourniture continue de gaz à ses clients dans la limite des quantités et des débits stipulés par le contrat entre eux (s'il n'y a pas de contrat, comme avec les domestiques par exemple, la fourniture s'entend sans arrêt). Deux sont les cas où la fourniture de gaz peut être interompue :

- en cas de force majeure, où le client affecté est averti dès que possible
- en cas de travaux programmés de la part du gestionnaire du réseau (raccordement, entretien etc) où le client est averti 24 heures suivant l'heure de réception de cette information par le gestionnaire.

Article 4 : Pour les clients qui ne souhaitent pas une fourniture susceptible aux interruptions, la continuité de fourniture doit être assurée même dans les trois situations suivantes :

- la source principale d'approvisionnement est disparrue pendant 6 mois au maximum dans des conditions météorologiques moyennes ;
- pendant un hiver très froid tel qu'il s'en produit un tous les 50 ans ;
- une température très basse pendant trois jours au maximum telle qu'il s'en produit une tous les 50 ans.

Article 5 : Pour leur permettre de remplir les obligations de continuité de fourniture imposées, en cas de rupture de tout ou partie des approvisionnements, les fournisseurs disposent de trois recours :

- interompre ou moduler la fourniture à certains clients lorsque ceci est prévu dans leurs contrats ;
- acheter des quantités supplémentaires de gaz sous forme de contrats à court terme de gaz naturel liquéfié ;
- stocker du gaz pendant des périodes précédentes.

Article 6 : En cas d'impossibilité de la part du fournisseur d'honorer ses engagements contractuels, une fourniture de dernier recours est assurée aux clients non doméstiques qui assurent une mission d'intérêt général pendant les 5 premiers jours par le gestionnaire de réseau de transport. À l'issue de ce délai, et si les clients n'ont pas pu trouver un autre fournisseur, ils peuvent faire appel au fournisseur de dernier recours pour effectuer la préstation prévue jusqu'à la fin du contrat initial. Le ministre lance une procédure d'appel à candidatures afin de désigner le fournisseur de dernier recours.

Article 7 : Les fournisseurs de gaz alimentent des clients domestiques et maintiennent la fourniture aux personnes en situation de précarité.

Article 8 : Les fournisseurs doivent établir quotidiennement les programmes de mouvements de gaz qu'ils prévoient d'injecter ou de soutirer aux points du réseau de transport (ou de distribution) identifiés par les parties dans le contrat ou le protocole d'accès au réseau. Ils sont tenus de communiquer au minimum tous les mois leurs prévisions de réservation de capacités aux gestionnaires de réseaux de transport (et de distribution).

C.2 Opérateurs de réseaux de transport de gaz

Article 9 : L'acheminement du gaz peut être interrompu sans préjudice des stipulations contractuelles pour autant que cette interruption soit nécessaire ou inévitable, à voir dans deux cas :

- en cas de force majeure, où l'opérateur avertit sans délai le fournisseur et le client final affecté, et
- en cas de travaux programmés (ou raccordement ou entretien) où l'opérateur s'efforce de réduire ses interruptions au minimum et de les situer aux dates et heures susceptibles de provoquer le moins de gêne possible aux clients, et il est obligé de communiquer (aux fournisseurs et aux clients directement raccordés au réseau de transport) ces dates aux moins deux mois à l'avance.

L'opérateur doit pouvoir assurer la continuité de l'acheminement du gaz même dans les situations de l'Article 4.

Article 10 : En cas de manquement grave de l'opérateur à ses obligations, des mesures sont pris de la part du ministère.

Article 11 : Les opérateurs assurent la pression, le débit et les caractérisiques physico-chimiques du gaz livré tels qu'ils soient conformes aux engagements vis à vis des fournisseurs.

Article 12 : Le ministre peut retirer ou suspendre l'autorisation de transport en cas de non-respect des obligations fixées.

C.3 Opérateurs de réseaux de distribution de gaz

Article 13 – Articles 15 cf Opérateurs de réseaux de transport de gaz.

Article 16 : Un client final dont la consommation annuelle de gaz est inférieure à $5{\cdot}10^6 KWh$ ne peut se raccorder qu'à un réseau de distribution.

C.4 Opérateurs de stockage

Article 17 : Les opérateurs de stockage sont tenus d'informer quotidiennement les opérateurs des réseaux de transport des capacités disponibles afin de leur permettre de passer des contrats en vue de l'équilibrage instantané de leurs réseaux.

Article 18 : Les opérateurs de stockage sont tenus d'informer au moins 2 mois à l'avance les fournisseurs et les transporteurs des travaux ou opérations de maintenance sur leurs installations susceptibles de limiter ou d'interrompre les injections et soutirages du gaz. En cas de force majeure ils sont tenus d'informer les opérateurs de réseaux de transport auxquels sont raccordés leurs stockages dans les plus brefs délais.

C.5 Exploitants d'installations de gaz naturel liquéfié

Article 19 : Les exploitants d'installations de gaz naturel informent les opérateurs des réseaux de transport de leurs disponibilités. cf aux Articles 17, 18.

C.6 Dispositions communes et diverses

Article 20 : Les contrats conclus entre les différents acteurs cités précédemment doivent comporter au moins les éléments suivants :

- la durée des contrats ;
- les modalités de fourniture et de livraison ;
- les prix et les modalités relatives à la facturation / abonnements / paiements ;
- les éventuelles conditions de raccordement ;

- les obligations concernant les intallations inférieures pour les clients domestiques ;
- les spécifications du gaz aux points de livraison et la description des droits et obligations des parties en cas de non-respect de ces spécifications ;
- les quantités de gaz à livrer, les débits et les modalités de comptage du gaz consommé ;
- le régime de responsabilité applicable à chacune des parties ;
- le mode de résolution des différends.

Article 21 : Concerne les compétence du personnel et l'organisation adaptée pour répondre aux obligations.

Article 22 : Concerne le contrôle du respect des dispositions du présent.

Résume de thèse : Résolution de grands problèmes stochastiques multi-étapes : Application à un problème de dimensionnement de capacités et de gestion de flux et de stocks.

Dans un monde déterministe, toute donnée d'un problème d'optimisation est censée être connue avec certitude. Dans le monde réel, on est souvent confronté à des cas où certains paramètres sont incertains. La demarche consistant à considérer un seul jeu de paramètres, supposant que ceci représente suffisamment bien la réalité, est vite mise en cause.

On considère travailler sur plusieurs périodes temporelles et sur un espace d'incertitude discrétisé, en introduisant ainsi les notions des abres de scénarios et des modèles multi-étapes. Les dimensions de ces problèmes augmentent de façon exponentielle avec le nombre de périodes d'étude, rendant les méthodes directes de résolution impossibles à appliquer.

Le problème qui a motivé ce travail est issu d'une application industrielle réelle et concerne la souscription de contrats dans un marché gazier. Les prix du marché spot, ainsi que la demande clientèle sont considérés incertains, et représentés par un arbre de scénarios. Le modèle qui ressort possède une structure ressemblant à une grande famille de problèmes dynamiques de dimensionnement. À l'issue d'un travail bibliographique, mené particulièrement sur les méthodes de résolution des modèles multi-étapes, la décomposition imbriquée est la méthode qui est retenue.

Sur les très grandes instances, même les méthodes de décomposition peuvent s'avérer longues à converger. Cette thèse est consacrée à de nouvelles mises en œuvre de la décomposition imbriquée, le but étant de pouvoir traiter plus de scénarios en moins de temps. Certains aspects de la méthode sont remises en cause, nous permettant de réduire le nombre d'itérations jusqu'à ce que la convergence soit atteinte. D'autres aspects sont également étudiés dans l'objectif de réduire le temps de calcul passé sur chaque itération séparément. Les demarches proposées sont validées à travers plusieurs séries d'expériences qui mettent en valeur la supériorité de l'approche proposée par rapport à l'approche classique.

Mots clès : Programmation stochastique, modèles multi-étapes, arbre de scénarios, décomposition imbriquée, problème de dimensionnement.

Abstract : Solving large Solving large multistage stochastic problems : A real-world application on capacity planning in a gas market.

In a deterministic setting, data input are considered to be known. However, in real-world applications we're facing problems whose parameters are partially or totally incertain. The approach where we consider a single scenario, which is supposed to represent a mean case, shows quickly its limits.

We consider working on a discretized incertainty space spreading over several time periods ; we therefore consider scenario trees and introduce the *multistage* models associated. Problems dimensions rise exponentially with the number of stages which renders direct solution methods inappropriate.

What has motivated our work is an industrial application arising in a gaz market, concerning more precisely capacity reservation in the context of a contractual agreement that has to hold over a certain time horizon. Spot prices and clients' demands are considered to be incertain and are modeled using a scenario tree. The problem structure presents strong similarities with a wide family of problems, where variables are coupling with each other in a very caracteristic manner. After a litterature survey focusing on (but not limited to) solution methods for multistage models, the Nested Decomposition method has been chosen.

Over very large cases, even decomposition methods show their limits ; this concerns in principle convergence times. This work is mostly devoted to the developpement of new procedures inside the Nested Decomposition method in order to work with larger scenario trees in less time. Other aspects, concerning time reduction over a single iteration are also studied. Comparisons between the classic and the newly presented aproaches revealed the superiority of the latters over the former.

Keywords : Stochastic programming, multistage models, scenario trees, nested decomposition.