Titre: Title:	Couplage des méthodes d'agrégation dynamique de contraintes et de stabilisation pour résoudre le problème d'horaires de véhicules avec dépôts multiples.
Auteur: Author:	Pascal Benchimol
Date:	2011
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Benchimol, P. (2011). Couplage des méthodes d'agrégation dynamique de contraintes et de stabilisation pour résoudre le problème d'horaires de véhicules avec dépôts multiples. [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/535/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/535/
Directeurs de recherche: Advisors:	Guy Desaulniers, & Jacques Desrosiers
Programme: Program:	Mathématiques appliquées

UNIVERSITÉ DE MONTRÉAL

COUPLAGE DES MÉTHODES D'AGRÉGATION DYNAMIQUE DE CONTRAINTES ET DE STABILISATION POUR RÉSOUDRE LE PROBLÈME D'HORAIRES DE VÉHICULES AVEC DÉPÔTS MULTIPLES.

PASCAL BENCHIMOL
DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION DU DIPLÔME DE
MAÎTRISE ÈS SCIENCES APPLIQUÉES
(MATHÉMATIQUES APPLIQUÉES)
MARS 2011

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

COUPLAGE DES MÉTHODES D'AGRÉGATION DYNAMIQUE DE CONTRAINTES ET DE STABILISATION POUR RÉSOUDRE LE PROBLÈME D'HORAIRES DE VÉHICULES AVEC DÉPÔTS MULTIPLES.

présenté par : M. BENCHIMOL Pascal, Ing.
en vue de l'obtention du diplôme de : Maîtrise ès Sciences Appliquées
a été dûment accepté par le jury constitué de :

M. SOUMIS François, Ph.D., président.
M. DESAULNIERS Guy, Ph.D., membre et directeur de recherche.
M. DESROSIERS Jacques, Ph.D., membre et co-directeur de recherche.
M. ORBAN Dominique, Ph.D., membre.

Remerciements

Je remercie tout d'abord mes directeurs de recherche M. Guy Desaulniers et M. Jacques Desrosiers pour leur encadrement pertinent ainsi que pour leur relecture attentive de ce mémoire.

Je remercie également M. Issmail El Hallaoui pour ses explications sur l'agrégation dynamique de contraintes et sur le fonctionnement du logiciel GENCOL.

Enfin je tiens à remercier Montréal et le Québec pour les bons moments que j'y ai passés avec mes amis.

Résumé

La méthode de génération de colonnes est très utilisée pour résoudre des problèmes de programmation linéaire. Cependant elle présente des faiblesses face aux effets de la dégénéréscence rencontrée sur certains problèmes de grande taille. Plusieurs approches ont été proposées pour contrer ses effets négatifs, en particulier l'agrégation dynamique de contraintes et la stabilisation des variables duales. Ce travail propose une méthode permettant de coupler ces deux approches pour les problèmes de partitionnement d'ensemble afin de tirer parti de leurs effets combinés.

La dégénéréscence apparaît lorsque le polytope des solutions réalisables contient des points extrêmes associées à plusieurs bases, ou de manière équivalente lorsque des solutions basiques contiennent des variables en base nulles. L'idée de l'agrégation dynamique de contraintes est de travailler avec une restriction du problème original qui comporte moins de contraintes. En conséquence, la taille des bases est réduite ainsi que le nombre de variables de base nulles. Cependant, la réduction du nombre de contraintes entraîne une perte d'information duale. Une des difficultés majeures de cette méthode est de désagréger l'information duale partielle disponible afin d'obtenir une solution duale au problème original. Elhallaoui *et al.* (2005) montrent que pour les problèmes de partitionnement d'ensemble, il est possible d'effectuer cette opération en résolvant un problème de plus court chemin. La méthode de stabilisation des variables duales propose quant à elle de pénaliser une grande partie de l'espace dual. Comme chaque base est associée à une solution duale, on évite ainsi l'exploration des bases les plus pénalisées. Cependant, le problème ainsi « stabilisé » est une relaxation du problème original. Pour résoudre ce dernier à l'optimalité, il faut résoudre une suite de problèmes stabilisés.

Coupler les méthodes d'agrégation dynamique de contraintes et de stabilisation des variables duales revient à réduire le nombre de contraintes des problèmes stabilisés. Malheureusement, l'information duale ainsi obtenue est partielle non seulement pour les contraintes du problème original, mais aussi pour la pénalité. Dans ce travail, nous montrons comment désagréger l'information duale par la résolution d'un problème de plus court chemin pour les problèmes de partitionnement d'ensemble stabilisés.

Cette nouvelle méthode est appliquée au problème d'horaires de véhicules avec dépôts multiples (MDVSP). C'est un problème de partitionnement d'ensemble auquel s'ajoutent des contraintes supplémentaires limitant le nombre de véhicules disponibles par dépôt. La taille des instances utilisées varie entre 500 et 3000 tâches. On obtient une réduction des temps de calcul d'un facteur d'environ 2 pour la résolution de la relaxation linéaire sur les instances de grande taille.

Abstract

Column generation is a widely used method for solving linear programming problems. It is very efficient but shows some weaknesses when faced with highly degenerate problems. Stabilization of dual variables and dynamic constraint aggregation are two examples of approaches used to damp the effects of degeneracy. This work shows how two combine the strength of both for set-partitioning problems.

Degeneracy occurs when the polyhedron of feasible solutions contains degenerate extreme points. An extreme point is degenerate when it can be described by multiple basis, which means that these basic solutions contain basic variables with zero value. The idea behind dynamic constraint aggregation is to use a restricted problem with fewer constraints. Thus the size of the basis is reduced along with the number of zero variables. However some dual information is lost when reducing the number of constraints. The challenge of this method is to desaggregate the partial dual information and compute a dual solution for the original problem. For set-partitioning problems, Elhallaoui *et al.* (2005) shows that this computation can be done by solving a shortest-path problem. Dual variables stabilization consists of penalizing most of the dual space. Since every basis is associated with a dual solution, this amounts to prevent the exploration of most penalized basis. Yet such a "stabilized" problem is a relaxation of the original one. In order to find an optimal solution, a sequence of stabilized problems needs to be solved.

Stabilization and dynamic constraint aggregation are brought together by reducing the number of constraints of stabilized problems. Unfortunately, this yields to partial dual information not only for the removed constraints, but also for penalty values. In this work, we show how to disaggregate dual information by solving a shortest-path problem for stabilized set-partitioning problems. This new method is applied to the multiple depot vehicle scheduling problem (MDVSP). It is a set-partitioning problem with side constraints that limit the number of vehicles used from each depot. Experiments were conducted on randomly generated instances which sizes range between 500 and 3000 tasks. Results show a reduction of computational times by a factor 2 approximately for large instances.

Table des matières

Liste des tableaux

Liste des figures

Liste des sigles et abréviations

cplex	solveur CPLEX implémentant la méthode du simplexe
dca	méthode d'agrégation dynamique de contraintes
gencol	méthode de génération de colonnes
MDVSP	problème d'horaires de véhicules avec dépôts multiples
stab	méthode de stabilisation de variables duales
stab-dca	méthode couplant la stabilisation des variables duales et l'agrégation dynamique de contraintes

Chapitre 1

Introduction

Depuis plusieurs dizaines d'années maintenant, de nombreux problèmes pratiques ont été résolus grâce à des méthodes d'optimisation mathématique, en particulier avec la programmation linéaire. Celle-ci permettant de modéliser une grande variété de problèmes et de les résoudre de manière efficace, elle est très utilisée. Cependant, lorsque la taille des problèmes devient très grande, les méthodes classiques de programmation linéaire, en particulier la méthode du simplexe, se trouvent ralenties par les effets de la dégénéréscence. Plusieurs méthodes assez différentes ont été proposées pour contrer ces effets, en particulier la stabilisation et l'agrégation dynamique de contraintes. Le but de ce travail est de coupler ces deux méthodes pour tirer parti de leurs effets combinés. Pour vérifier l'utilité de cette nouvelle méthode, elle sera testée sur une classe particulière de problèmes : le problème d'horaires de véhicules avec dépôts multiples.

1.1 Concepts de base

La dégénéréscence est un phénomène qui ralentit en pratique un certain nombre de méthodes de programmation linéaire (Gal, 1993), en particulier la méthode du simplexe. Elle apparaît lorsque le polytope des solutions réalisables contient des points extrêmes associés à plusieurs bases, ou de manière équivalente lorsque des solutions basiques contiennent des variables en base nulles. Face à ce type de points extrêmes, la méthode du simplexe peut effectuer des pivots dégénérés, c'est-à-dire explorer un certain nombre de bases avant de pouvoir atteindre un nouveau point extrême. Lorsque le nombre de pivots dégénérés devient important, les performances de la méthode en sont grandement affectées. Plusieurs approches ont été proposées pour lutter contre cet effet comme des règles de pivotage spécifiques (voir Terlaky (1993) pour une vue d'ensemble), la perturbation des contraintes (Charnes, 1986), la relaxation lagrangienne (Geoffrion, 1974), l'agrégation de contraintes statique (Rogers *et al.*, 1991) ou dy-

namique (Shetty et Taylor, 1987; Pan, 1998; Elhallaoui *et al.*, 2005, 2008, 2010) ou encore la stabilisation des variables duales (du Merle *et al.*, 1999; Oukil *et al.*, 2007). Ce travail est construit à partir des méthodes d'agrégation dynamique de contraintes (Elhallaoui *et al.*, 2005, 2008, 2010) et de stabilisation des variables duales (du Merle *et al.*, 1999; Oukil *et al.*, 2007) qui vont être expliquées en détail plus loin.

L'idée de l'agrégation dynamique de contraintes est de travailler avec une restriction du problème original qui comporte moins de contraintes. En conséquence, la taille des bases est réduite ainsi que le nombre de variables de base nulles. Cette réduction du nombre de contraintes est effectuée via une projection de l'ensemble réalisable dans un sous-espace de dimension réduite. Pour les problèmes de partitionnement d'ensemble auxquels on s'intéresse, cette projection revient à partitionner les contraintes et conserver une seule contrainte par partie. Cependant, la réduction du nombre de contraintes entraîne une perte d'information duale. Une des difficultés majeures de cette méthode est de désagréger l'information duale partielle disponible afin d'obtenir une solution duale au problème original. Elhallaoui *et al.* (2005) montrent que pour les problèmes de partitionnement d'ensemble, il est possible d'effectuer cette opération en résolvant un problème de plus court chemin.

La méthode de stabilisation des variables duales propose quant à elle de pénaliser une grande partie de l'espace dual. Comme chaque base est associée à une solution duale, on évite ainsi l'exploration des bases les plus pénalisées. Dans un premier temps, on définit une zone de confiance dans laquelle les solutions duales sont non pénalisées. Un coût proportionnel à la distance à la zone de confiance est ajouté aux solutions situées à l'extérieur. Cependant, le problème ainsi « stabilisé » est une relaxation du problème original. Pour résoudre ce dernier à l'optimalité, une suite de problèmes stabilisés doit être résolue.

1.2 Objectifs de recherche

La finalité de ce travail est l'utilisation conjointe des méthodes d'agrégation dynamique de contraintes et de stabilisation des variables duales. En pratique, ce couplage revient à appliquer l'agrégation dynamique de contraintes sur les problèmes stabilisés résolus au cours de la méthode de stabilisation des variables duales. Malheureuse-

ment, l'information duale obtenue de ces problèmes stabilisés et agrégés est partielle non seulement pour les contraintes du problème original, mais aussi pour la pénalité. Le but de ce travail est d'être encore capable de désagréger l'information duale par la résolution d'un problème de plus court chemin pour les problèmes de partitionnement d'ensemble stabilisés.

Pour montrer l'utilité de cette nouvelle méthode, elle est appliquée au problème d'horaires de véhicules avec dépôts multiples (MDVSP). C'est un problème de partitionnement d'ensemble auquel s'ajoute des contraintes supplémentaires limitant le nombre de véhicules disponibles par dépôt (une définition plus précise du MDVSP sera donnée au chapitre 4). L'objectif est de réduire les temps de résolution pour la relaxation linéaire par rapport aux méthodes pré-citées.

1.3 Plan du mémoire

Après avoir décrit dans le détail les méthodes d'agrégation dynamique de contraintes et de stabilisation au chapitre 2, le chapitre 3 présentera l'algorithme permettant de les coupler. Ensuite le chapitre 4 introduira le problème d'horaires de véhicules avec dépôts multiples ainsi qu'une heuristique pour le résoudre. Les solutions primale et duale produites par cette heuristique sont utilisées pour démarrer les méthodes d'agrégation dynamique de contraintes et de stabilisation des variables duales. Enfin le chapitre 5 donnera les résultats produits par l'utilisation cette nouvelle méthode sur des instances de tailles diverses du problème d'horaires de véhicules avec dépôts multiples. Une brève conclusion est présentée au chapitre 6.

Chapitre 2

Revue de la littérature

Dans ce chapitre, nous présenterons deux méthodes permettant d'optimiser des programmes linéaires dégénérés de grande taille : l'agrégation dynamique de contraintes et la stabilisation des variables duales. Ces deux approches seront utilisées dans le cadre de la génération de colonnes, qui sera donc notre point de départ.

2.1 La génération de colonnes

La génération de colonnes est un cadre assez général pour la résolution de programmes linéaires présentant des groupes de contraintes correspondant à des problèmes bien étudiés. L'idée remonte à Dantzig et Wolfe (1960) et a été étudiée et exploitée avec succès depuis (Desrosiers *et al.*, 1995; Barnhart *et al.*, 1998; Desaulniers *et al.*, 1998, 2005; Lübbecke et Desrosiers, 2005). Cette section présente les grandes lignes de ce cadre.

2.1.1 Contexte d'application

La génération de colonnes est principalement utilisée (et utile) pour résoudre la relaxation linéaire de problèmes linéaires en nombres entiers ($PLNE$). Une formulation générique de cette classe de problèmes, obtenue par application du principe de décomposition de Dantzig-Wolfe (Dantzig et Wolfe, 1960) est la suivante :

$$
(PLNE) \qquad
\begin{aligned}
\min_{\lambda} \quad & \sum_{x \in \Delta} c^T x \lambda_x \\
& \sum_{x \in \Delta} A x \lambda_x = b \\
& \sum_{x \in \Delta} x \lambda_x \ \in \ \mathbb{Z}^n \\
& \lambda_x \geq 0 \qquad\qquad \forall x \in \Delta
\end{aligned}
$$

où

- Δ est l'ensemble des points extrêmes d'un polytope inclus dans $\mathbb{R}^n$
- $\lambda \in \mathbb{R}^{|\Delta|}$
- $A \in \mathbb{R}^m \times \mathbb{R}^n$
- $b \in \mathbb{R}^m$
- $c \in \mathbb{R}^n$

Dans un cadre plus général, cette formulation est modifiée pour prendre en compte les rayons extrêmes du polytope Δ. Par souci de simplicité, et dans toute la suite, on considérera que Δ est un polytope borné et donc qu'il ne contient pas de rayons extrêmes. Remarquez qu'un point extrême $x \in \Delta$ est associé à une variable λ_x et une colonne Ax. Dans la suite et en l'absence d'ambiguïté, on utilisera indifféremment les termes « point extrême », « variable » ou « colonne » pour se référer au même objet.

À cause de la contrainte d'intégrité, ce type de problème est en général NP-difficile. On résout donc des relaxations linéaires du problème qui serviront de bornes inférieures dans une méthode d'énumération implicite (branch & bound).

2.1.2 La méthode

La relaxation linéaire de (PLNE) reste un problème difficile, car le nombre de points extrêmes $|\Delta|$ d'un polytope peut être très grand. On travaille donc avec un problème maître restreint (PMR_Ω) à un sous-ensemble $\Omega \subset \Delta$. Mais il est possible que la solution optimale utilise des points extrêmes n'appartenant pas à Ω. On définit alors un sous-problème (SP) chargé de générer de nouveaux points extrêmes associés à des variables de coût réduit négatif qui sont ajoutés à Ω. Le problème maître restreint et le sous-problème se formulent ainsi :

$$(PMR_\Omega) \quad \begin{array}{ll} \min\limits_{\lambda} & \sum\limits_{x\in\Omega} c^T x \lambda_x \\ & \sum\limits_{x\in\Omega} Ax\lambda_x = b \\ & \lambda_x \geq 0 \ \ \forall x \in \Omega \end{array} \qquad (SP) \quad \bar{c} = \min_{x\in\Delta} \ c^T x - \alpha^T A x$$

où $\alpha \in \mathbb{R}^m$ est le vecteur de variables duales associé aux contraintes du problème maître restreint. Pour que (PMR_Ω) soit réalisable, il faut bien sûr choisir un ensemble

Ω qui contient une solution réalisable au problème.

L'algorithme de génération de colonnes est le suivant :

1. Choisir un sous-ensemble de colonnes $\Omega \subset \Delta$ contenant une solution réalisable.
2. Construire (PMR_Ω) et le résoudre par une méthode de programmation linéaire (par la méthode du simplexe par exemple).
3. Constuire et résoudre le sous-problème.
4. Si $\bar{c} \geq 0$, arrêter.
5. Sinon le sous-problème a généré des variables de coût réduit négatif. Elles sont ajoutées à Ω.
6. Retourner en 2.

On appellera *itération mineure* une itération de cet algorithme, c'est à dire une résolution du problème maître restreint (PMR_Ω) suivie d'une résolution du sous-problème.

2.1.3 Preuve d'optimalité

Quand l'algorithme s'arrête, il n'a pas énuméré tous les points extrêmes de Δ. Il faut donc être sûr que ceux qui ont été laissé de côté ne peuvent pas améliorer la fonction objectif. Une telle preuve nous est donnée par application du théorème des écarts complémentaires.

Théorème 1. *Soit λ une solution réalisable de (PMR_Δ). λ est une solution optimale de (PMR_Δ) si et seulement si il existe $\alpha \in \mathbb{R}^m$ tel que :*

$$\forall x \in \Delta \qquad \lambda_x(c^T x - \alpha^T A x) = 0 \tag{2.1}$$

$$\forall x \in \Delta \qquad c^T x - \alpha^T A x \geq 0 \tag{2.2}$$

*Un vecteur α vérifiant (2.1) sera appelée **solution duale complémentaire** à la solution λ.*
*Un vecteur α vérifiant (2.2) sera appelée **solution duale réalisable**.*

Notons Ω_f l'ensemble de points extrêmes générés lorsque notre algorithme s'arrête. Comme on a résolu (PMR_{Ω_f}) à l'optimalité, la solution obtenue (λ, α) vérifie la condition (2.1) pour tout $x \in \Omega_f$. En posant $\lambda_x = 0$ pour $x \in \Delta \setminus \Omega_f$, la condition (2.1) est vérifiée pour tout $x \in \Delta$. De plus, le sous problème n'a généré aucune variable de coût réduit négatif, ce qui signifie que la condition (2.2) est vérifiée pour tout $x \in \Delta$. On a donc obtenu une solution optimale à (PMR_Δ).

Cette méthode très efficace montre ses limites face à des problèmes très dégénérés. Dans les sections suivantes, nous présenterons deux améliorations de la génération de colonnes qui permettent de réduire les effets indésirables de la dégénérescence.

2.2 L'agrégation dynamique de contraintes

L'agrégation dynamique de contraintes (Elhallaoui *et al.*, 2005) et ses extensions (Elhallaoui *et al.*, 2008, 2010) s'attaquent aux problèmes très dégénérés présentant principalement des contraintes de partitionnement d'ensemble (i.e. $\forall x \in \Delta,\ Ax \in \{0,1\}^m$ et $b_i = 1$ pour $i = 1, \ldots, m$), par exemple les problèmes de tournées de véhicules ou de confection d'horaires. Elle repose sur une idée simple : un problème est dégénéré lorsque la méthode du simplexe effectue beaucoup de pivots dégénérés. Ces pivots apparaissent sur les solutions basiques comportant un grand nombre de variables en base avec une valeur nulle. En réduisant le nombre de contraintes, la taille de la base et donc le nombre de variables nulles en base diminuera.

D'un point de vue algébrique, cela revient à considérer un problème agrégé dont les contraintes sont définies dans un sous-espace vectoriel V de $\mathbb{R}^m$ de dimension inférieure à m. Tout en travaillant dans V, on souhaite obtenir une solution satisfaisant toutes les contraintes dans $\mathbb{R}^m$.

2.2.1 Approche de principe

Prenons un sous-espace vectoriel V de $\mathbb{R}^m$ contenant le vecteur b. Soit $Q = \{C_1, ..., C_l\}$ une base orthogonale de V et $R = \{C_{l+1}, ..., C_m\}$ une base orthogonale de $V^\perp$. Soit H la matrice de passage de la base $\{Q, R\}$ à la base canonique de $\mathbb{R}^m$.

Comme on effectue un changement de base orthogonal, on peut décomposer H en

$$H = \begin{pmatrix} H^V \\ H^{V^\perp} \end{pmatrix}$$

où H^V est la matrice de la projection orthogonale sur V, et $H^{V^\perp}$ celle de la projection orthogonale sur $V^\perp$. Réécrivons le système de contraintes dans cette nouvelle base :

$$\sum_{x\in\Omega} Ax\lambda_x = b \Leftrightarrow H\sum_{x\in\Omega} Ax\lambda_x = Hb$$
$$\Leftrightarrow \begin{cases} H^V \sum_{x\in\Omega} Ax\lambda_x = H^V b \\ H^{V^\perp} \sum_{x\in\Omega} Ax\lambda_x = H^{V^\perp} b. \end{cases}$$

Comme $b \in V$, la projection de b sur $V^\perp$ est nulle, donc $H^{V^\perp} b = 0$. On se retrouve avec le système suivant :

$$H^V \sum_{x\in\Omega} Ax\lambda_x = H^V b \tag{2.3}$$
$$H^{V^\perp} \sum_{x\in\Omega} Ax\lambda_x = 0. \tag{2.4}$$

L'équation (2.4) demande que la combinaison linéaire des colonnes appartienne à V. Une condition suffisante pour vérifier cette contrainte est d'utiliser uniquement des colonnes appartenant à V. C'est ce qu'on va faire par la suite.

Soit Ω un ensemble de colonnes et V un sous-espace vectoriel contenant b. Le problème agrégé s'écrit finalement :

$$(PMRA_{\Omega,V}) \qquad \begin{aligned} \min_{\lambda} \quad & \sum_{x\in\Omega_V} cx\lambda_x \\ & \sum_{x\in\Omega_V} H^V Ax\lambda_x = H^V b \\ & \lambda_x \geq 0 \qquad\qquad \forall x \in \Omega_V, \end{aligned}$$

où Ω_V est le sous-ensemble de colonnes de Ω qui appartiennent à V.

Remarquez que toute solution réalisable du problème agrégé ($PMRA_{\Omega,V}$) vérifie les contraintes (2.3) et (2.4). Elle est donc réalisable pour le problème original (PMR_{Ω}). Pour que ($PMRA_{\Omega,V}$) soit réalisable, il faut bien choisir Ω et V. Des conditions suffisantes sont :

- Ω contient une solution réalisable du problème désagrégé.
- V contient les colonnes de Ω formant une solution réalisable.

Ce problème agrégé est plus facile à résoudre que le problème original. D'abord il a moins de variables, et ensuite il a moins de contraintes. Il est donc moins sujet à la dégénérescence. Cependant c'est une restriction du problème original : on a remplacé la contrainte (2.4) « la solution appartient à V », par la contrainte beaucoup plus forte $\lambda_x = 0 \ \ \forall x \notin \Omega_V$, i.e, « la solution est formée de colonnes appartenant à V ».

Il est donc fort probable que les solutions optimales du problème original ne soient pas réalisables pour le problème agrégé. Le problème agrégé sera optimisé. Si la solution obtenue est optimale pour le problème original, la méthode s'arrête. Si ce n'est pas le cas, le sous-espace V est modifié afin d'explorer de nouvelles solutions. Ces opérations seront répétées jusqu'à ce que V contienne toutes les colonnes formant une solution optimale au problème original.

Reste à savoir de quelle façon modifier le sous-espace V. Les conditions d'optimalité du problème original conduisent au raisonnement suivant : la solution optimale du problème agrégé vérifie la contrainte (2.2) pour toutes les colonnes appartenant à V. Par contre, il est possible que des colonnes n'appartenant pas à V ne vérifient pas (2.2), c'est-à-dire qu'elles ont un coût réduit négatif. Comme l'utilisation de ces colonnes peut améliorer l'objectif, V sera étendu pour les contenir.

La modification de V se base sur les coût réduits, et le calcul des coûts réduits demande la connaissance d'une solution duale au problème original. Or la résolution du problème agrégé ne donne qu'une solution duale agrégée. Il faut donc trouver une méthode pour calculer une solution duale complète (étape dite de *désagrégation des variables duales*).

Dernier obstacle : définir les contraintes du problème agrégé et identifier les colonnes appartenant à V demande de calculer les matrices des projecteurs sur V et $V^{\perp}$, c'est-

à-dire d'inverser une matrice carrée de taille m. Une inversion de matrice peut être coûteuse en terme de temps de calcul ($O(m^3)$ en utilisant l'élimination de Gauss-Jordan). D'autant plus coûteuse que l'on va devoir résoudre une suite de problèmes agrégés, et donc inverser souvent des matrices. Les sections suivantes montreront comment, en se restreignant aux contraintes de partitionnement et en choisissant intelligement V et sa base Q, définir le problème agrégé et identifier les colonnes de V sans aucune inversion de matrice.

L'algorithme de l'agrégation dynamique de contraintes est le suivant :

1. Choisir un sous-ensemble de variables Ω formant une solution réalisable.
2. Construire un sous-espace V contenant les colonnes de Ω.
3. Construire et résoudre $(PMRA_{\Omega,V})$.
4. Désagréger les variables duales.
5. Résoudre le sous-problème (SP).
6. Si $\bar{c} \geq 0$, arrêter.
7. Si le sous-problème a généré des colonnes appartenant à V et de coût réduit négatif, les ajouter à Ω et retourner en 3.
8. Sinon, le sous-problème n'a généré que des colonnes de coût réduit négatif n'appartenant pas à V, les ajouter à Ω, puis retourner en 2.

Cette approche sera appliquée aux problèmes de partitionnement d'ensemble. Dans toute la suite de ce chapitre, les colonnes Ax seront donc des colonnes binaires, et $b_i = 1$ pour $i = 1, \ldots, m$. Dans un premier temps, la construction du sous-espace V à partir d'une solution réalisable sera présentée. Ensuite, on montrera comment cette construction permet, sans aucune inversion de matrice, d'identifier les colonnes appartenant à V et de définir le problème agrégé. Ensuite, la modification de V sera abordée. Enfin on décrira comment désagréger les variables duales.

2.2.2 Construction d'un sous-espace vectoriel à partir d'une solution réalisable

Considérons un problème de partitionnement d'ensemble ainsi qu'une solution réalisable λ^0 au problème (PMR_Δ). Notons V^0 le sous-espace vectoriel de $\mathbb{R}^m$ en-

gendré par les colonnes associées aux variables non nulles de λ^0. Notons T l'ensemble des contraintes et $(e_t)_{t\in T}$ la base canonique de $\mathbb{R}^m$. Une famille génératrice de V^0, orthogonale et formée de colonnes binaires sera fabriquée. Cette famille génératrice sera une base du sous-espace de travail V. Commençons par une propriété importante de ce type de famille.

Proposition 1. *Soit $Q = \{C_l \mid l \in L\}$ une famille de colonnes de $\{0,1\}^m$. Q est orthogonale si et seulement si $\forall l \in L$ et $\forall t \in T$:*

$$(C_l)^T e_t = 1 \qquad \Rightarrow \qquad \forall l' \in L,\ l' \neq l \qquad (C_{l'})^T e_t = 0 \tag{2.5}$$

Démonstration. Soit Q une famille orthogonale et $t_0 \in T$. Supposons qu'il existe l et l' tels que $(C_l)^T e_{t_0} = (C_{l'})^T e_{t_0} = 1$. On a :

$$\begin{aligned}(C_l)^T C_{l'} &= \sum_{t\in T} (C_l)^T e_t (C_{l'})^T e_t \\ &\geq (C_l)^T e_{t_0} (C_{l'})^T e_{t_0}\end{aligned}$$

car les colonnes sont binaires (à coefficients positifs aurait suffit). On en déduit $(C_l)^T C_{l'} \geq 1$, ce qui contredit l'orthogonalité de Q.

Soit Q une famille vérifiant l'implication (2.5). Prenons deux colonnes distinctes de Q, C_l et $C_{l'}$. Pour tout $t \in T$ si $(C_l)^T e_t = 1$, alors $C_{l'}^T e_t = 0$. On en déduit que

$$(C_l)^T C_{l'} = \sum_{t\in T} (C_l)^T e_t (C_{l'})^T e_t = 0.$$

Donc Q est orthogonale. □

Prenons la famille $Q = \{Ax \in \mathbb{R}^m \mid \lambda_x^0 > 0\}$, qui est naturellement une famille génératrice de V^0. On souhaite obtenir une famille orthogonale et formée de colonnes binaires qui soit génératrice de V^0. En pratique, on va partir d'une solution réalisable entière. Dans ce cas, Q est déjà une famille orthogonale et formée de colonnes binaires. Si on ne part pas d'une solution entière, on peut appliquer l'algorithme suivant.
Algorithme d'orthogonalisation binaire :

1. Soit $Q = \{Ax \in \mathbb{R}^m \mid \lambda_x^0 > 0\}$. Tout au long de l'algorithme, Q restera une famille génératrice de V^0 formée de colonnes binaires.
2. Si Q forme une famille orthogonale, arrêter.
3. Sinon, il existe C_i et C_j dans Q telles que $(C_i)^T C_j \neq 0$. La proposition 1 indique qu'il existe un sous-ensemble non vide de contraintes $T_{ij} \subset T$ tel que :

$$\forall t \in T_{ij} \qquad (C_i)^T e_t = (C_j)^T e_t = 1$$

Prendre un ensemble T_{ij} de cardinalité maximale.
Notons

$$C'_k = \sum_{t \in T_{ij}} e_t$$
$$C'_i = C_i - C'_k$$
$$C'_j = C_j - C'_k$$

On pose $Q = Q \cup \{C'_k, C'_i, C'_j\} \setminus \{C_i, C_j\}$ et on retourne en 2.
Les colonnes C'_i, C'_j et C'_k sont des colonnes binaires. Le sous-espace vectoriel engendré par$\{C_i, C_j\}$ est inclus dans le sous-espace vectoriel engendré par $\{C'_k, C'_i, C'_j\}$, donc Q reste bien un famille génératrice de V^0.

On choisit un ensemble T_{ij} de cardinalité maximale pour obtenir la famille Q orthogonale de cardinalité minimale, le but étant de travailler dans un sous-espace V de dimension la plus petite possible. Cet algorithme converge car le nombre de paire de colonnes non orthogonales diminue strictement à chaque itération.

Remarquez que les colonnes de Q induisent naturellement une partition de l'ensemble des contraintes.

Proposition 2. *Pour chaque vecteur $C_l \in Q$, définissons la partie $W_l \subset T$:*

$$W_l = \{t \in T \mid C_l^T e_t = 1\}$$

$(W_l)_{l \in L}$ forme une partition de l'ensemble des contraintes T.

Démonstration. La proposition 1 nous dit que les parties $(W_l)_{l\in L}$ sont deux à deux disjointes. V^0 a été engendré par des colonnes formant une solution réalisable, donc $b \in V^0$. Q étant une famille génératrice de V^0, il est possible d'exprimer b comme combinaison linéaire des colonnes de Q. Comme $b_i = 1$, pour $i = 1, \ldots, m$ dans la base canonique et que les parties W_l sont deux à deux disjointes, la seule combinaison linéaire possible est :

$$b = \sum_{l\in L} C_l$$

On conclut que $\cup_{l\in L} W_l = T$.

□

Par abus de language, Q désignera aussi bien la famille de colonnes orthogonales que la partition associée.

On peut maintenant définir l'espace de travail V : c'est le sous-espace vectoriel engendré par la famille Q. Comme Q est une famille orthogonale, c'est une famille libre. C'est donc une base de V.

2.2.3 Identification des variables compatibles

Maintenant que l'espace de travail V est défini, regardons comment identifier les colonnes appartenant à V, c'est-à-dire les colonnes compatibles.

Définition 1. *La colonne Ax associée au point extrême $x \in \Delta$ est dite **compatible** avec le sous-espace vectoriel V si et seulement si $Ax \in V$.*

Dans un cas très général, on pourrait définir le projecteur sur $V^\perp$. Une colonne est alors compatible si et seulement si elle appartient au noyau du projecteur. Ici, on peut tirer parti de la structure binaire des colonnes et du caractère orthogonal de Q.

Proposition 3. *La colonne Ax associée au point extrême $x \in \Delta$ est **compatible** avec V si et seulement si elle est une combinaison linéaire binaire des colonnes de Q, i.e. il existe $\mu \in \{0,1\}^{|Q|}$ tel que*

$$Ax = \sum_{l\in L} \mu_l C_l$$

Démonstration. Soit une colonne Ax vérifiant les conditions de la proposition. Ax est une combinaison linéaire des colonnes de Q qui forment une base de V, donc $Ax \in V$.

Soit Ax une colonne compatible. Comme Q est une base de V, il existe $\mu \in \mathbb{R}^{|Q|}$ tel que

$$Ax = \sum_{l \in L} \mu_l C_l$$

Pour chaque colonne $C_l \in Q$, il existe au moins un vecteur e_{t_l} de la base canonique de $\mathbb{R}^m$ tel que $C_l^T e_{t_l} = 1$. La proposition 1 nous dit que pour tout $C_{l'} \in Q$, $l' \neq l$, on a $C_{l'}^T e_{t_l} = 0$. On en déduit que :

$$(Ax)^T e_{t_l} = \mu_l (C_l)^T e_{t_l} = \mu_l$$

Comme Ax est une colonne à coefficients binaires dans la base canonique, ceci montre que $\mu_l \in \{0, 1\}$ et ce pour tout $l \in L$. □

Cette proposition permet d'expliciter un algorithme très simple permettant de détecter si une colonne binaire Ax est compatible et, le cas échéant, d'exprimer la colonne dans la base Q.

Algorithme de détection des colonnes compatibles :

1. Choisir $C_l \in Q$, et définir

$$W_l = \{t \in T \mid C_l^T e_t = 1\}$$

2. Choisir $t_0 \in W_l$ et poser $\mu_l = (Ax)^T e_{t_0}$.
3. Pour tout $t \in W_l$, si $(Ax)^T e_t \neq \mu_l$, arrêter. La colonne est incompatible.
4. Poser $Q = Q \setminus \{C_l\}$.
5. Si $Q = \varnothing$, arrêter. La colonne est compatible et ses coordonnées dans la base Q sont données par le vecteur μ construit.
6. Retourner en 1.

Quelle est la complexité de cet algorithme ?

En pratique, on marque l'ensemble $\{t|(Ax)^T e_t = 1\}$des contraintes couvertes par la colonne Ax. Cette opération se fait en $O(m)$. Dans un second temps, on parcoure l'ensemble des parties W_l de la partition Q. Si une partie contient à la fois une

contrainte marquée et une contrainte non marquée, la colonne Ax est incompatible. Comme Q est une partition de l'ensemble des contraintes, cette seconde opération s'effectue aussi en $O(m)$.

2.2.4 Construction du problème agrégé

Le sous-espace de travail V est défini, une de ses bases connue et la détection des variables compatibles est efficace. Maintenant regardons comment définir les contraintes du problème agrégé ($PMRA_{\Omega,V}$) sans avoir à calculer la matrice de projection H^V.

Pour cela, il suffit de remarquer que le vecteur $H^V Ax$ n'est autre que le vecteur des coordonnées de la colonne Ax dans la base Q. De même, le vecteur $H^V b$ est le vecteur des coordonnées de b dans la base Q. Comme expliqué plus haut, $b = \sum_{l \in L} C_l$, donc le vecteur $H^V b$ a tous ses coefficients égaux à 1. Pour décider quelles colonnes seront utilisées dans ($PMRA_{\Omega,V}$), il faut appliquer l'algorithme de détection des colonnes compatibles, ce qui donne les coordonnées des colonnes compatibles dans la base Q. ($PMRA_{\Omega,V}$) est donc défini sans travail supplémentaire.

Le problème agrégé définit une contrainte T_l par vecteur C_l de la base Q. Il est important de remarquer que la contrainte T_l est identique à toutes les contraintes $t \in W_l$ de (PMR_{Ω_V}). Soit $l_0 \in L$ et $t \in W_{l_0}$, la contrainte t s'écrit :

$$\sum_{x \in \Omega} (Ax)^T e_t \lambda_x = 1.$$

Notons $\mu_l^x \in \{0, 1\}$ les coordonnées de la colonne Ax dans la base Q :

$$Ax = \sum_{l \in L} \mu_l^x C_l.$$

La proposition 1 nous dit que $(Ax)^T e_t = \mu_{l_0}^x$. On en déduit que toutes les contraintes $t \in W_{l_0}$ sont identiques et s'écrivent :

$$\sum_{x \in \Omega} \mu_{l_0}^x \lambda_x = 1,$$

ce qui est encore identique à la contrainte T_{l_0} du problème agrégé.

Le problème agrégé peut donc être défini comme le problème (PMR_{Ω_V}) auquel on a retiré toutes les contraintes, sauf une par partie W_l. L'ensemble des contraintes du problème agrégé $\{T_l \mid l \in L\}$ est donc un sous-ensemble des contraintes T du problème initial. Autrement dit, la matrice $H^V A$ n'est autre que la matrice A à laquelle on a retiré des lignes.

2.2.5 Modification dynamique de la partition

Un problème agrégé a été défini et sa solution optimale est réalisable pour le problème désagrégé. Malheureusement, résoudre à l'optimalité le problème agrégé n'assure pas l'obtention d'une solution optimale au problème désagrégé. L'optimum est atteint lorsque le sous-problème ne génère plus de variables de coût réduit négatif. Or le problème agrégé se restreint aux variables compatibles, et il est possible, et même probable, que de nombreuses variables incompatibles aient un coût réduit négatif. Tant que ces variables ne sont pas prises en compte dans le problème maître, le sous-problème continuera de les générer, et donc l'optimum ne sera pas atteint (ou du moins on n'aura pas prouvé que la solution courante est optimale). Pour prendre en compte ces variables incompatibles, il faut modifier V afin de les rendre compatibles.

Pour rendre une variable λ_x compatible, il suffit d'ajouter la colonne Ax à la famille Q, puis d'appliquer l'algorithme d'orthogonalisation binaire. Remarquez que l'algorithme de détection des variables compatibles nous donne, pour une colonne incompatible $C_i = Ax$, l'ensemble des colonnes C_j de Q telles que $C_i^T C_j \neq 0$ et des ensembles $T_{ij} = \{t \in T \mid C_i^T e_t = C_j^T e_t = 1\}$ de cardinalité maximale. Rendre une variable compatible se résume donc à créer les colonnes C'_i, C'_j, C'_k, les ajouter à Q et retirer les colonnes C_j de Q. Des structures de données adaptées permettent d'effectuer ces opérations en $O(m)$.

A moins d'avoir une très bonne solution initiale, on sera obligé de modifier V à un moment donné. Choisir judicieusement à quel moment modifier V peut avoir une influence sur la performance de la méthode. On peut attendre de n'avoir aucune colonne compatible de coût réduit négatif, ou on peut définir des critères heuristiques. Par exemple, Elhallaoui *et al.* (2005) proposent de comparer les coût réduits des colonnes

incompatibles à ceux des colonnes compatibles. S'il existe une colonne incompatible avec un coût réduit bien plus petit que ceux des colonnes compatibles, V est modifié.

Mentionnons que Elhallaoui *et al.* (2010) présentent une méthode pour générer des colonnes « peu » incompatibles. Une colonne aura un nombre d'incompatibilité k s'il faut augmenter la partition de k parties pour la rendre compatible (autrement dit s'il faut ajouter k dimensions au sous-espace V pour la colonne compatible). Pour k donné, une modification des sous-problèmes permet de générer uniquement des colonnes dont le nombre d'incompatibilités est inférieur à k. Cette stratégie est intéressante car en générant des colonnes « peu » incompatibles, l'augmentation de la taille de la partition est ralentie.

La procédure de modification de la partition utilisée est la suivante. En début de résolution, on définit un nombre k d'incompatibilités maximal initialisé à 0. Ensuite, à chaque itération mineure, les étapes suivantes sont effectuées :

1. Générer des colonnes ayant un nombre d'incompatibilités d'au plus k.
2. Si moins de 5 colonnes ont été générées, augmenter k et retourner en 1.
3. S'il existe une colonne incompatible ayant un coût réduit inférieur au plus petit coût réduit des colonnes compatibles, modifier la partition ainsi :
   - Classer les colonnes incompatibles par coût réduit croissant.
   - Ajouter les colonnes incompatibles dans l'odre croissant des coûts réduits, seulement si elles sont orthogonales à toutes les colonnes incompatibles précédemment ajoutées.

Le nombre d'incompatibilités k prend successivement les valeurs $\{0, 1, 2, +\infty\}$, et sa valeur est conservée d'une itération mineure à la suivante.

2.2.6 Désagrégation des variables duales

La résolution du problème agrégé ($PMRA_{\Omega,V}$) donne un couple optimal de solutions primale-duale $(\lambda, \hat{\alpha})$. On a vu que la solution primale était solution réalisable du problème initial. Essayons de déduire de la solution duale agrégée $\hat{\alpha}$ une solution duale α au problème initial qui soit complémentaire à λ et duale réalisable.

La solution duale agrégée $\hat{\alpha}$ est définie seulement sur les contraintes $(T_l)_{l\in L}$ du problème agrégé. On peut la compléter par des zéros, c'est-à-dire poser $\alpha_t = 0$ pour $t \notin \{T_l \mid l \in L\}$. Par optimalité de $(PMRA_{\Omega,Q})$, la solution ainsi obtenue vérifie la condition de complémentarité (2.1) pour toutes les variables compatibles. Comme $\lambda_x = 0$ pour les variables x incompatibles, la condition (2.1) est vérifiée pour tout $x \in \Omega$. La solution duale ainsi obtenue est bien complémentaire à λ. Cependant elle n'est que partiellement réalisable. En effet la condition de faisabilité duale (2.2) est vérifiée pour toutes les variables λ_x qui sont compatibles par optimalité de $(PMRA_{\Omega,V})$, mais rien ne prouve la faisabilité duale pour les variables incompatibles. Cherchons donc une solution duale α, complémentaire à λ et qui soit le plus réalisable possible, c'est-à-dire qui vérifie la condition (2.2) pour toutes les variables compatibles mais aussi pour un certain nombre de variables incompatibles.

On veut une solution complémentaire à λ et réalisable pour toutes les variables compatibles. Par optimalité de $(PMRA_{\Omega,V})$, une condition suffisante pour vérifier cette propriété est :

$$\sum_{t\in W_l} \alpha_t = \hat{\alpha}_{T_l} \qquad \forall l \in L. \tag{2.6}$$

Transformation en un problème de plus court chemin

Dans chaque partie W_l, indexons les contraintes par les indices $\{1, ..., |W_l|\}$. Comme les parties $(W_l)_{l\in L}$ forment une partition de l'ensemble des contraintes, chaque variable $(\alpha_t)_{t\in T}$ peut être indexée par un couple (l, h) avec $l \in L$ et $h \in \{1, ..., |W_l|\}$. Effectuons le changement de variables suivant :

$$\left|\begin{array}{ll} \pi^l_h = \displaystyle\sum_{j=1}^{h} \alpha^l_j & \forall l \in L, \forall h \in \{1, ..., |W_l|\} \\ \pi^l_0 = 0 & \forall l \in L. \end{array}\right. \tag{2.7}$$

Pour chaque contrainte $t \in T$, on peut écrire α_t sous la forme :

$$\alpha_t = \pi^l_i - \pi^l_{i-1},$$

avec $t \in W_l$, $i \in \{1, ..., |W_l|\}$.
La contrainte (2.6) devient alors

$$\pi^l_{|W_l|} = \hat{\alpha}_{T_l} \qquad \forall l \in L.$$

Et pour une certaine classe de colonnes incompatibles, Elhallaoui *et al.* (2005) montrent que la condition de faisabilité duale (2.2) se met sous la forme suivante :

$$\pi^r_j - \pi^l_i \leq \tilde{c},$$

où $l, r \in L$, $j \in \{1, ..., |W_r|\}$, $i \in \{1, ..., |W_l|\}$ et $\tilde{c}$ est un coût réduit partiel.

Définissons le graphe G suivant.

- G comporte $|T| + 1$ noeuds : un noeud pour chaque contrainte (l, i) avec $l \in L$ et $i \in \{1, ..., |W_l|\}$ plus un noeud source correspondant à toutes les variables π^l_0.
- Chaque contrainte $\pi^l_i - \pi^r_j \leq \tilde{c}$ induit un arc partant du noeud (l, i) vers le noeud (r, j).
- La contrainte (2.6) peut se réécrire sous la forme :

$$\begin{aligned} \pi^l_{|W_l|} - \pi^l_0 &\leq \hat{\alpha}_{T_l} && \forall l \in L \\ \pi^l_0 - \pi^l_{|W_l|} &\leq -\hat{\alpha}_{T_l} && \forall l \in L, \end{aligned}$$

  ce qui correspond, pour chaque partie l, à un arc allant du noeud $(l, |W_l|)$ à la source et un arc allant de la source au noeud $(l, |W_l|)$.

Si π^l_i est la longueur du plus court chemin dans le graphe G entre le noeud source et le noeud (l, i), alors π vérifie toutes les contraintes (2.6) et les contraintes de faisabilité duale pour les colonnes incompatibles considérées.

Un dernier écueuil apparaît : le problème de plus court chemin ainsi défini peut comporter des cycles de longueur négative. En effet, si la solution λ du problème agrégé n'est pas optimale pour le problème original, il se peut qu'il soit impossible de trouver une solution duale complémentaire et réalisable pour toutes les colonnes incompatibles considérées. Pour contourner cet obstacle, une approche possible est la suivante : le

problème de plus court chemin est résolu par un algorithme de plus court chemin qui détecte les cycles de longueur négatives (par exemple Ford-Bellman). Si un tel cycle est trouvé, on retire un des arcs du cycle (c'est-à-dire qu'on oublie la contrainte de faisabilité duale pour une des colonnes incompatibles), puis on réapplique l'algorithme de plus court chemin.

2.3 La stabilisation des variables duales

Cette section présente la méthode de stabilisation des variables duales. Cette approche combine l'idée de pénalisation provenant des algorithmes de point proximal (Rockafellar, 1976; Güler, 1991) et le concept de zone de confiance de la méthode Boxstep (Marsten *et al.*, 1975). Nous utilisons ici les travaux de du Merle *et al.* (1999) et Oukil *et al.* (2007) où la stabilisation des variables duales est exprimée dans le cadre de la programmation linéaire et appliquée à des problèmes très dégénérés.

Comme mentionné plus haut, un problème est dégénéré lorsqu'il existe beaucoup de solutions basiques comportant peu de variables non nulles par rapport au nombre de contraintes. Si une solution nécessite peu de colonnes non nulles, il faut trouver des colonnes complétant la solution pour former une base. Dans le cadre de la génération de colonnes, le nombre de colonnes est énorme. Il existe donc de nombreuses façons de compléter une solution pour former une base. Chaque base est associée de manière unique à sa solution duale complémentaire. Lorsque l'algorithme du simplexe effectue un pivot dégénéré, la base change mais la solution basique ne change pas.

L'idée de la stabilisation est de pénaliser une grande partie de l'espace des solutions duales. Cela revient à augmenter le coût des bases associées à une solution duale pénalisée. Les bases les plus pénalisées ne seront donc pas explorées, ce qui réduira le nombre de pivots dégénérés. Malheureusement, il est possible que l'on ait trop pénalisé les bases optimales. Il faudra donc modifier dynamiquement la zone pénalisée pour obtenir une solution optimale au problème.

La pénalisation de l'espace dual peut aussi s'interpréter du point de vue du sous-problème. En effet la direction de l'objectif du sous-problème $c - A^T\alpha$ est fonction de la solution duale courante α. Pénaliser une partie de l'espace dual revient donc à

défavoriser certaines des directions possibles de l'objectif du sous-problème. C'est-à-dire que l'on va favoriser la génération de points extrêmes optimaux pour les directions non pénalisées.

Dans un premier temps, on présentera l'approche adoptée pour pénaliser l'espace des solutions duales. Ensuite on expliquera comment faire évoluer cette pénalité pour s'assurer d'obtenir une solution optimale au problème.

2.3.1 Pénalisation de l'espace dual

Considérons le problème original et son dual :

$$(PMR_\Omega) \quad \begin{aligned} \min_{\lambda} \quad & \sum_{x\in\Omega} c^T x \lambda_x \\ & \sum_{x\in\Omega} Ax\lambda_x = b \\ & \lambda_x \geq 0 \ \ \forall x \in \Omega \end{aligned} \quad \Bigg| \quad (DMR_\Omega) \quad \begin{aligned} \max_{\alpha} \quad & b^T\alpha \\ & \alpha^T Ax \leq c^T x \qquad \forall x \in \Omega. \end{aligned}$$

La stratégie adoptée pour pénaliser une partie de l'espace des solutions duales est la suivante :

- pour chaque variable duale, on définit une « zone de confiance » sans pénalité ;
- plus une variable duale est éloignée de sa zone de confiance, plus on pénalise le coût de la solution.

La fonction de pénalité choisie est linéaire en trois morceaux, comme représenté à la figure 2.1. Si α est compris entre δ^- et δ^+ aucune pénalité ne s'applique. Si α est plus grand que δ^+, on applique une pénalité de valeur $\varepsilon^+\omega^+$ où $\omega^+ = \alpha - \delta^+$ est une variable mesurant la distance entre α et la zone de confiance. De manière similaire, si α est plus petit que δ^-, on applique une pénalité de valeur $\varepsilon^-\omega^-$ où $\omega^- = \delta^- - \alpha$.

La version stabilisée du problème dual est :

$$(DMRS_\Omega) \quad \begin{aligned} \max_{\alpha,\omega^+,\omega^-} \quad & b^T\alpha - {\varepsilon^-}^T\omega^- - {\varepsilon^+}^T\omega^+ \\ & \alpha^T Ax \leq c^T x \qquad \forall x \in \Omega \\ & \alpha \leq \delta^+ + \omega^+ \\ & \delta^- - \omega^- \leq \alpha \\ & \omega^+, \omega^- \geq 0, \end{aligned}$$

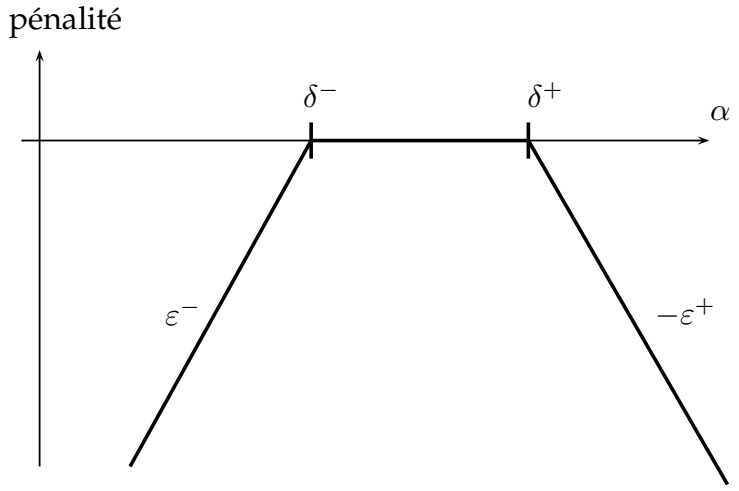


FIGURE 2.1 Fonction de pénalisation des variables duales

avec $\varepsilon^+ > 0$, $\varepsilon^- > 0$ et $\delta^+ > \delta^-$.

On en déduit le problème primal stabilisé :

$$
(PMRS_\Omega) \qquad
\begin{aligned}
\min_{\lambda, y^+, y^-} \quad & \sum_{x \in \Omega} c^T x \lambda_x - {\delta^-}^T y^- + {\delta^+}^T y^+ \\
& \sum_{x \in \Omega} A x \lambda_x - y^- + y^+ = b \\
& y^+ \leq \varepsilon^+ \\
& y^- \leq \varepsilon^- \\
& y^+, y^- \geq 0 \\
& \lambda_x \geq 0 \qquad \forall x \in \Omega.
\end{aligned}
$$

Remarquez que le problème primal stabilisé est une relaxation du problème primal original. En effet, toute solution λ de (PMR_Ω) induit une solution réalisable de même coût $(\lambda, 0, 0)$ dans $(PMRS_\Omega)$.

2.3.2 Critère de convergence

La résolution du problème stabilisé ne donne qu'une borne inférieure sur la solution optimale du problème initial. Il faut donc résoudre une suite de problèmes stabilisés $(PMRS^k_\Omega)_{k \in \mathbb{N}}$ pour atteindre la solution optimale du problème initial. Reste à savoir

comment définir cette suite pour assurer la convergence. La proposition ci-dessous nous donne une condition suffisante. On appelera *itération majeure* la résolution d'un problème stabilisé ($PMRS^k_\Omega$).

Notons z_k la valeur optimale de ($PMRS^k_\Omega$). Par dualité forte, c'est aussi la valeur optimale de ($DMRS^k_\Omega$). La proposition suivante donne une condition suffisante pour assurer la convergence de la suite $(z_k)_{k\in\mathbb{N}}$ vers la solution optimale de (PMR_Ω) :

Proposition 4. *Soit $(\alpha_k, \omega_k^+, \omega_k^-)$ une solution optimale de ($DMRS^k_\Omega$). Si les paramètres de ($DMRS^{k+1}_\Omega$) vérifient :*

$$\delta_{k+1}^{-T} e_t < \alpha_k^T e_t < \delta_{k+1}^{+T} e_t \quad \forall t \in T,$$

alors

- *Soit $z_{k+1} > z_k$.*
- *Soit $z_{k+1} = z_k$ et α_k est une solution optimale de (DMR_Ω).*

Démonstration. On a :

$$z_k = b^T \alpha_k - \varepsilon_k^{-T} \omega_k^- - \varepsilon_k^{+T} \omega_k^+.$$

Si la condition de la proposition est vérifiée, alors $(\alpha_k, 0, 0)$ est une solution réalisable pour ($DMRS^{k+1}_\Omega$) de coût $b^T\alpha_k$. En conséquence :

$$\begin{aligned} z_{k+1} &\geq b^T \alpha_k \\ &\geq b^T \alpha_k - \varepsilon_k^{-T} \omega_k^- - \varepsilon_k^{+T} \omega_k^+ \\ &\geq z_k, \end{aligned}$$

car $\omega_k^+, \omega_k^-, \varepsilon_k^+$ et ε_k^+ sont tous positifs.

Si $z_{k+1} = z_k$, alors la solution $(\alpha_k, 0, 0)$ est optimale pour ($DMRS^{k+1}_\Omega$) car c'est une solution réalisable et son coût est $b^T\alpha_k \geq z_k$.

De plus α_k est optimale pour (DMR_Ω). En effet supposons qu'il existe une solution α' réalisable pour (DMR_Ω) telle que $b^T\alpha' > b^T\alpha_k$. Alors $(\alpha', 0, 0)$ est une solution réalisable pour $(DMRS_\Omega^{k+1})$ de coût strictement supérieur à $b^T\alpha_k$, ce qui contredit l'optimalité de $(\alpha_k, 0, 0)$. □

2.3.3 Mise à jour des paramètres de stabilisation

La proposition 4 étant assez peu restrictive, on a une grande liberté sur la mise à jour des paramètres de stabilisation $\varepsilon^+, \varepsilon^-, \delta^+, \delta^-$. Deux stratégies de mise à jour, inspirées de la stratégie dynamique présentée dans Oukil *et al.* (2007), ont été testées. Dans les deux cas, la zone de confiance à l'itération $k+1$ est centrée sur la solution duale obtenue à l'itération k. Les régles de mise à jour modifient dynamiquement, pour chaque composante $t \in T$, la largeur de la zone de confiance $\Delta_t^k = \delta_k^{+T} e_t - \delta_k^{-T} e_t$ et les pénalités $\varepsilon_k^{+T} e_t$, $\varepsilon_k^{-T} e_t$.

Stratégie 1 : expansion-seulement

On regarde chaque variable duale indépendamment les unes des autres : si une variable duale est à l'intérieur de la zone de confiance, on peut penser que cette valeur est « bonne ». On ne change pas la largeur de la zone de confiance et on augmente les pénalités. Si au contraire une variable duale est à l'extérieur de la zone de confiance, on considère que la zone de confiance était trop contraignante pour cette variable. On diminue donc les pénalités et on augmente la largeur de la zone. De manière plus formelle :

- Si $\delta_k^{-T} e_t < \alpha_k^T e_t < \delta_k^{+T} e_t$, la largeur de la zone de confiance ne change pas. Les pénalités sont doublées si leur valeurs ne dépassent pas 1 :

$$\begin{aligned} \Delta_t^{k+1} &= \Delta_t^k \\ \varepsilon_{k+1}^{+T} e_t &= \min\left\{2\varepsilon_k^{+T} e_t, 1.0\right\} \\ \varepsilon_{k+1}^{-T} e_t &= \min\left\{2\varepsilon_k^{-T} e_t, 1.0\right\}. \end{aligned}$$

- Si $\alpha_k^T e_t \leq \delta_k^{-T} e_t$ ou $\delta_k^{+T} e_t \leq \alpha_k^T e_t$, la largeur de la zone de confiance est multipliée par deux. Les pénalités sont divisées par deux si leur valeurs ne tombent pas en

dessous de la précision machine ϵ :

$$\begin{aligned}
\Delta_t^{k+1} &= 2\Delta_t^k \\
\varepsilon_{k+1}^{+T} e_t &= \max\left\{0.5\varepsilon_k^{+T} e_t,\ \epsilon\right\} \\
\varepsilon_{k+1}^{-T} e_t &= \max\left\{0.5\varepsilon_k^{-T} e_t,\ \epsilon\right\}.
\end{aligned}$$

Stratégie 2 : expansion-contraction

On part du même principe que précédemment, mais pour éviter une trop grande expansion de la zone de confiance, on la contracte autour des variables duales « bien » approchées.

- Si $\delta_k^{-T} e_t < \alpha_k^T e_t < \delta_k^{+T} e_t$, la largeur de la zone de confiance est divisée par deux si sa valeur ne tombe pas en dessous de 0.1. Les pénalités sont doublées si leur valeurs ne dépassent pas 1 :

$$\begin{aligned}
\Delta_t^{k+1} &= \max\{\ 0.5\Delta_t^k, 0.1\} \\
\varepsilon_{k+1}^{+T} e_t &= \min\left\{2\varepsilon_k^{+T} e_t, 1.0\right\} \\
\varepsilon_{k+1}^{-T} e_t &= \min\left\{2\varepsilon_k^{-T} e_t, 1.0\right\}.
\end{aligned}$$

- Si $\alpha_k^T e_t < \delta_k^{-T} e_t$ ou $\delta_k^{+T} e_t < \alpha_k^T e_t$, la largeur de la zone de confiance est multipliée par quatre. Les pénalités sont divisées par deux si leur valeurs ne tombent pas en dessous de la précision machine ϵ :

$$\begin{aligned}
\Delta_t^{k+1} &= 4\Delta_t^k \\
\varepsilon_{k+1}^{+T} e_t &= \max\left\{0.5\varepsilon_k^{+T} e_t,\ \epsilon\right\} \\
\varepsilon_{k+1}^{-T} e_t &= \max\left\{0.5\varepsilon_k^{-T} e_t,\ \epsilon\right\}.
\end{aligned}$$

- Si $\alpha_k^T e_t = \delta_k^{-T} e_t$ ou $\delta_k^{+T} e_t = \alpha_k^T e_t$, la largeur de la zone de confiance et les

pénalités ne sont pas modifiées :

$$\begin{aligned}\Delta_t^{k+1} &= \Delta_t^k \\ \varepsilon_{k+1}^{+T} e_t &= \varepsilon_k^{+T} e_t \\ \varepsilon_{k+1}^{-T} e_t &= \varepsilon_k^{-T} e_t.\end{aligned}$$

Pour les deux stratégies, la largeur de la zone de confiance ainsi que les pénalitées sont initialisées avec une valeur de 0.1.

Dans la première stratégie, la zone de confiance sera élargie pour les variables duales « mal » évaluées. La deuxième stratégie ajoute un mécanisme de contraction de la zone de confiance pour les variables duales « bien » évaluées. Comme la première stratégie ne fait qu'étendre la zone de confiance, elle est susceptible de contenir rapidement une solution duale optimale. Mais une zone de confiance trop grande va mener à un problème très dégénéré. On s'attend donc à converger en faisant « peu » de mises à jour, mais chaque problème stabilisé sera long à résoudre. La deuxième stratégie crée des zones de confiance plus petites, donc moins susceptibles de contenir une solution duale optimale. Cependant le problème stabilisé sera peu dégénéré. On s'attend donc à converger en faisant « beaucoup » de mises à jour, mais chaque problème stabilisé sera rapide à résoudre.

Chapitre 3

Couplage de l'agrégation dynamique de contraintes et de la stabilisation des variables duales

Ce chapitre décrit la contribution mathématique apportée au cours de cette maîtrise. Il s'agit de tirer parti des deux méthodes présentées au chapitre 2, l'agrégation dynamique de contraintes et la stabilisation des variables duales, pour réduire au maximum les effets négatifs de la dégénérescence. Plus précisément, les problèmes stabilisés seront maintenant résolus par agrégation dynamique de contraintes. La difficulté est d'écrire un problème maître agrégé et stabilisé qui permet toujours de désagréger les variables duales en résolvant un problème de plus court chemin.

Dans un premier temps, le problème stabilisé et agrégé sera défini et ses propriétés étudiées. Dans un second temps, la façon de désagréger les variables duales sera abordée dans le détail. Dans tout ce chapitre, on parlera exclusivement de problèmes de partitionnement d'ensemble.

3.1 Méthode

Comme pour la stabilisation, une suite de problèmes stabilisés $(PMRS^k_\Omega)_{k\in\mathbb{N}}$ sera résolue. Par contre, l'agrégation dynamique de contraintes sera utilisée pour ces résolutions. Pour cela on construira un problème stabilisé et agrégé $(PMRSA^k_{\Omega,V})$. Il sera défini de telle sorte que, à partir d'un couple de solutions primale et duale optimales de coût z_k, $(\widehat{\lambda}, \widehat{y}^+, \widehat{y}^-)$ pour $(PMRSA^k_{\Omega,V})$ et $(\widehat{\alpha}, \widehat{\omega}^+, \widehat{\omega}^-)$ pour son dual $(DMRSA^k_{\Omega,V})$, il soit facile de :

- construire une solution (λ, y^+, y^-) réalisable pour $(PMRS^k_\Omega)$ de coût z_k ;
- constuire, par la résolution d'un problème de plus court chemin, une solution

réalisable $(\alpha, \omega^+, \omega^-)$ pour $(DMRS^k_\Omega)$ qui soit complémentaire à (λ, y^+, y^-). Par complémentarité, on aura ainsi résolu $(PMRS^k_\Omega)$ à l'optimalité. La construction du problème stabilisé et agrégé $(PMRSA^k_{\Omega,V})$ étant identique pour à chaque itération majeure k, cet indice sera omis par la suite.

3.2 Un problème maître agrégé et stabilisé

Prenons un problème stabilisé :

$$
(PMRS_\Omega) \qquad
\begin{aligned}
\min_{\lambda, y^+, y^-} \quad & \sum_{x \in \Omega} c^T x \lambda_x - \delta^{-T} y^- + \delta^{+T} y^+ \\
& \sum_{x \in \Omega} A x \lambda_x - y^- + y^+ = b \\
& y^+ \leq \varepsilon^+ \\
& y^- \leq \varepsilon^- \\
& y^+, y^- \geq 0 \\
& \lambda_x \geq 0 \qquad \forall x \in \Omega,
\end{aligned}
$$

avec $\varepsilon^+ > 0$, $\varepsilon^- > 0$ et $\delta^+ > \delta^-$.

Comme décrit à la section 2.2, on part d'une solution réalisable au problème original (PMR_Ω). Un sous-espace vectoriel V contenant la solution réalisable est choisi et une base orthogonale Q de V formée de colonnes binaires est explicitée. Cette base induit une partition $\{W_l \mid l \in L\}$ de l'ensemble de contraintes T. Définissons alors le problème stabilisé et agrégé :

$$
(PMRSA_{\Omega,V}) \qquad
\begin{aligned}
\min \quad & \sum_{x \in \Omega} c^T x \lambda_x - \widehat{\delta}^{-T} \widehat{y^-} + \widehat{\delta}^{+T} \widehat{y^+} \\
& \sum_{x \in \Omega} H^V A x \lambda_x - \widehat{y^-} + \widehat{y^+} = 1 \\
& \widehat{y^-} \leq \widehat{\varepsilon^-} \\
& \widehat{y^+} \leq \widehat{\varepsilon^+} \\
& \widehat{y^+}, \widehat{y^-} \geq 0 \\
& \lambda_x \geq 0 \qquad \forall x \in \Omega,
\end{aligned}
$$

avec

$$\begin{aligned}
&\forall l \in L \qquad \widehat{\delta_l^-} = \sum_{t \in W_l} \delta_t^- \\
&\forall l \in L \qquad \widehat{\delta_l^+} = \sum_{t \in W_l} \delta_t^+ \\
&\forall l \in L \qquad \widehat{\varepsilon_l^-} = \min_{t \in W_l} \varepsilon_t^- \\
&\forall l \in L \qquad \widehat{\varepsilon_l^+} = \min_{t \in W_l} \varepsilon_t^+.
\end{aligned}$$

Proposition 5. *Soit $(\lambda, \widehat{y}^+, \widehat{y^-})$ une solution de $(PMRSA_{\Omega,V})$ de coût z. En choisissant :*

$$\begin{aligned}
y_t^+ &= \widehat{y_l^+} && \forall l \in L, \qquad \forall t \in W_l \\
y_t^- &= \widehat{y_l^-} && \forall l \in L, \qquad \forall t \in W_l,
\end{aligned}$$

la solution (λ, y^+, y^-) est réalisable pour $(PMRS_\Omega)$ et a un coût z.

Démonstration. Vérifions d'abord que (λ, y^+, y^-) est réalisable pour $(PMRS_\Omega)$. Soit T_l une contrainte de $(PMRSA_{\Omega,V})$. On a vu dans la section 2.2.4 que toutes les contraintes $t \in W_l$ de $(PMRS_\Omega)$ sont identiques à T_l. Cela signifie que, pour tout $t \in W_l$:

$$(\sum_{x \in \Omega} Ax\lambda_x - b)^T e_t = \widehat{y_l^-} - \widehat{y_l^+} = y_t^- - y_t^+.$$

Donc la contrainte $\sum_{x \in \Omega} Ax\lambda_x - y^- + y^+ = b$ est vérifiée.

Soit $t \in T$ une contrainte et l l'indice de la partie W_l contenant t. On a :

$$\begin{aligned} y_t^+ = \widehat{y_l}^+ &\leq \widehat{\varepsilon}_l^+ \\ &\leq \min_{t' \in W_l} \varepsilon_{t'}^+ \\ &\leq \varepsilon_t^+. \end{aligned}$$

Donc $y^+ \leq \varepsilon^+$. De même, $y^- \leq \varepsilon^-$.

Le coût de la solution (λ, y^+, y^-) est :

$$\begin{aligned} &\sum_{x \in \Omega} c^T x \lambda_x - {\delta^-}^T y^- + {\delta^+}^T y^+ \\ &= \sum_{x \in \Omega} c^T x \lambda_x + \sum_{t \in T} (-\delta_t^- y_t^- + \delta_t^+ y_t^+) \\ &= \sum_{x \in \Omega} c^T x \lambda_x + \sum_{l \in L} \sum_{t \in W_l} (-\delta_t^- y_t^- + \delta_t^+ y_t^+) \\ &= \sum_{x \in \Omega} c^T x \lambda_x + \sum_{l \in L} (-\widehat{y_l^-} \sum_{t \in W_l} \delta_t^- + \widehat{y_l^+} \sum_{t \in W_l} \delta_t^+) \\ &= \sum_{x \in \Omega} c^T x \lambda_x + \sum_{l \in L} (-\widehat{y_l^-}\, \widehat{\delta_t^-} + \widehat{y_l^+} \widehat{\delta_t^+}) \\ &= z. \end{aligned}$$

□

Remarquez que cette démonstration nécessite de travailler sur les problèmes de partitionnement d'ensemble. Dans le cas général, désagréger les variables d'écart $\widehat{y^+}$ et $\widehat{y^-}$ de manière à obtenir une solution réalisable semble moins évident.

3.3 Désagrégation des variables duales

Pour prouver l'optimalité de la solution (λ, y^+, y^-) du problème primal désagrégé $(PMRS_\Omega)$, une solution $(\alpha, \omega^+, \omega^-)$ réalisable pour le problème dual $(DMRS_\Omega)$ et complémentaire à (λ, y^+, y^-) est construite. Les variables d'écart introduites par la stabilisation impliquent de nouvelles contraintes de complémentarité. Dans un premier temps, ces nouvelles contraintes seront présentées. Ensuite on montrera que, modulo le changement de variables (2.7) introduit pour l'agrégation dynamique de contraintes,

elles ont la structure de conditions d'optimalité d'un problème de plus court chemin. Après une étude détaillée du graphe ainsi induit, la méthode choisie pour la résolution du problème de plus court chemin sera expliquée.

3.3.1 Contraintes de complémentarité

Pour que la solution $(\alpha, \omega^+, \omega^-)$ soit complémentaire à (λ, y^+, y^-), il faut qu'elle vérifie :

$$\forall x \in \Omega \qquad \lambda_x > 0 \;\Rightarrow\; \alpha^T A x = c^T x \tag{3.1}$$

$$\forall t \in T \qquad y_t^+ > 0 \;\Rightarrow\; \alpha_t = \delta_t^+ + \omega_t^+ \tag{3.2}$$

$$\forall t \in T \qquad y_t^- > 0 \;\Rightarrow\; \delta_t^- - \omega_t^- = \alpha_t \tag{3.3}$$

$$\forall t \in T \qquad \omega_t^+ > 0 \;\Rightarrow\; y_t^+ = \varepsilon_t^+ \tag{3.4}$$

$$\forall t \in T \qquad \omega_t^- > 0 \;\Rightarrow\; y_t^- = \varepsilon_t^-. \tag{3.5}$$

Les conditions (3.2) et (3.4) couplées aux conditions $\omega^+ \geq 0$ impliquent :

$$y_t^+ = \varepsilon_t^+ \Rightarrow \begin{cases} \alpha_t \geq \delta_t^+ \\ \omega_t^+ = \alpha_t - \delta_t^+ \end{cases}$$

$$0 < y_t^+ < \varepsilon_t^+ \Rightarrow \begin{cases} \alpha_t = \delta_t^+ \\ \omega_t^+ = 0 \end{cases}$$

$$y_t^+ = 0 \Rightarrow \begin{cases} \alpha_t \leq \delta_t^+ \\ \omega_t^+ = 0. \end{cases}$$

De manière similaire, les conditions (3.3) et (3.5) couplées aux conditions $\omega^- \geq 0$ impliquent :

$$
\begin{aligned}
y_t^- = \varepsilon_t^- &\Rightarrow \begin{cases} \alpha_t \leq \delta_t^- \\ \omega_t^- = \delta_t^- - \alpha_t \end{cases} \\
0 < y_t^- < \varepsilon_t^- &\Rightarrow \begin{cases} \alpha_t = \delta_t^- \\ \omega_t^- = 0 \end{cases} \\
y_t^- = 0 &\Rightarrow \begin{cases} \alpha_t \geq \delta_t^- \\ \omega_t^- = 0. \end{cases}
\end{aligned}
$$

Les contraintes de complémentarités s'écrivent finalement ainsi :

$$
\left|
\begin{array}{ll}
\forall x \in \Omega \text{ tel que } \lambda_x > 0 & \alpha^T A x = c^T x \\
\forall t \in T \text{ tel que } y_t^+ = \varepsilon_t^+ & \alpha_t \geq \delta_t^+ \\
\forall t \in T \text{ tel que } 0 < y_t^+ < \varepsilon_t^+ & \alpha_t = \delta_t^+ \\
\forall t \in T \text{ tel que } y_t^+ = 0 & \alpha_t \leq \delta_t^+ \\
\forall t \in T \text{ tel que } y_t^- = \varepsilon_t^- & \alpha_t \leq \delta_t^- \\
\forall t \in T \text{ tel que } 0 < y_t^- < \varepsilon_t^- & \alpha_t = \delta_t^- \\
\forall t \in T \text{ tel que } y_t^- = 0 & \alpha_t \geq \delta_t^-.
\end{array}
\right. \tag{3.6}
$$

La recherche de la solution duale revient donc à résoudre le problème de faisabilité en α défini par les contraintes (3.6). Les valeurs de ω^+ et ω^- seront entièrement déterminées par la donnée de y^+, y^- et α.

Comme pour l'agrégation dynamique de contraintes, on remplace la contrainte (3.1) par la condition suffisante (2.6), on effectue le changement de variables (2.7) et on ajoute les contraintes de faisabilité duale pour une certaine classe de colonnes

incompatibles. Le problème de faisabilité devient alors :

$$\left|\begin{array}{ll}
\forall l \in L & \pi^l_{|W_l|} = \widehat{\alpha}_l \\
\forall t \in T \text{ tel que } y^+_t = \varepsilon^+_t & \pi^l_j - \pi^l_i \geq \delta^+_t \\
\forall t \in T \text{ tel que } 0 < y^+_t < \varepsilon^+_t & \pi^l_j - \pi^l_i = \delta^+_t \\
\forall t \in T \text{ tel que } y^+_t = 0 & \pi^l_j - \pi^l_i \leq \delta^+_t \\
\forall t \in T \text{ tel que } y^-_t = \varepsilon^-_t & \pi^l_j - \pi^l_i \leq \delta^-_t \\
\forall t \in T \text{ tel que } 0 < y^-_t < \varepsilon^-_t & \pi^l_j - \pi^l_i = \delta^-_t \\
\forall t \in T \text{ tel que } y^-_t = 0 & \pi^l_j - \pi^l_i \geq \delta^-_t \\
\forall (l,r) \in L^2, (i,j) \in E_{l,r} & \pi^l_j - \pi^l_i \leq \tilde{c}^{l,r}_{i,j}.
\end{array}\right.$$

où les ensembles $E_{l,r}$ indexent les contraintes induites par les colonnes incompatibles retenues.

Remarquez que le même raisonnement peut se faire sur une solution optimale $(\lambda, \widehat{y}^+, \widehat{y}^-)$ du problème stabilisé et agrégé et sa solution duale complémentaire $(\widehat{\alpha}, \widehat{\omega}^+, \widehat{\omega}^-)$. Celles-ci vérifient donc :

$$\left|\begin{array}{ll}
\forall l \in L \text{ tel que } \widehat{y}^+_l = \widehat{\varepsilon}^+_l & \widehat{\alpha}_l \geq \widehat{\delta}^+_l \\
\forall l \in L \text{ tel que } 0 < \widehat{y}^+_l < \widehat{\varepsilon}^+_l & \widehat{\alpha}_l = \delta^+_l \\
\forall l \in L \text{ tel que } \widehat{y}^+_l = 0 & \widehat{\alpha}_l \leq \widehat{\delta}^+_l \\
\forall l \in L \text{ tel que } \widehat{y}^-_l = \widehat{\varepsilon}^-_l & \widehat{\alpha}_l \leq \widehat{\delta}^-_l \\
\forall l \in L \text{ tel que } 0 < \widehat{y}^-_l < \widehat{\varepsilon}^-_l & \widehat{\alpha}_l = \widehat{\delta}^-_l \\
\forall l \in L \text{ tel que } \widehat{y}^-_l = 0 & \widehat{\alpha}_l \geq \widehat{\delta}^-_l.
\end{array}\right. \tag{3.7}$$

3.3.2 Résolution du problème de plus court chemin

Ce problème de faisabilité a encore la structure des conditions d'optimalité d'un problème de plus court chemin. Malheureusement, il est encore possible que ce problème comporte des cycles de longueur négative. Dans l'agrégation dynamique de contraintes, on détectait ces cycles et on pouvait enlever n'importe quel arc du cycle. Ici, on ne peut pas enlever des arcs correspondant aux contraintes de complémentarité, sinon on perdrait la preuve d'optimalité sur le problème original.

L'approche adoptée pour la résolution du sous-problème se base sur l'algorithme de Floyd-Warshall (Ahuja *et al.*, 1993) qui permet de détecter les plus courts chemins entre toutes les paires de sommets du graphe. Ici, on calcule la matrice $M = [m_{ij}]$ qui contient la longueur des plus courts chemins *ne passant pas par le noeud source* entre toutes les paires de noeuds du graphe. Ceci permet d'avoir une matrice beaucoup plus creuse (c'est-à-dire comportant de nombreux coefficients de valeur $+\infty$) et ce sans aucune perte d'information, puisque l'on connaît la longueur des plus courts chemins entre le noeud source et tous les autres noeuds.

Dans un premier temps, la matrice est initialisée en utilisant uniquement les arcs provenant des conditions de complémentarité. Cette étape peut se réaliser en $O(|T|^2)$, et on montrera que les conditions de complémentarité ne peuvent pas induire de cycle de longueur négative. Ensuite, les arcs induits par les colonnes incompatibles sont ajoutés un à un. La matrice des plus courts chemins permet de détecter immédiatement si l'ajout d'un tel arc crée un cycle de longueur négative. Si c'est le cas, l'arc incriminé est abandonné. Sinon, les plus courts chemins sont mis à jour, et on passe à l'arc suivant. Si I indexe l'ensemble des colonnes incompatibles considérées, cette étape prend $O(|I||T|^2)$ opérations. De manière plus formelle, l'algorithme s'énonce comme suit.
Algorithme de résolution du problème de plus court chemin :

1. Initialiser la matrice M des plus courts chemins en utilisant uniquement les arcs provenant des conditions de complémentarité.
2. Choisir $(k, l) \in I$ un arc provenant d'une colonne incompatible de coût c_{kl}.
3. $I = I \setminus \{(k, l)\}$.
4. Si $c_{kl} + m_{lk} < 0$, cet arc forme un cycle de longueur négative ne passant pas par le noeud source. Aller en 7.
5. Si $m_{l,source} + m_{source,k} + c_{kl} < 0$, cet arc forme un cycle de longueur négative passant par le noeud source. Aller en 7.
6. Sinon, pour tout $i, j \in T \cup \{source\}$, poser $m_{ij} = \min\{m_{ij}, m_{ik} + c_{kl} + m_{lj}\}$.
7. Si I est vide, arrêter.
8. Retourner en 2.

La section suivante analysera d'un peu plus près le graphe induit par les contraintes de complémentarité. Cela montrera que ce graphe ne comporte pas de cycles de longueur

négative, et permettra d'initialiser la matrice des plus courts chemins en un temps $O(|T|^2)$.

3.3.3 Graphe induit par les contraintes de complémentarité

Commençons par une remarque simple sur les variables d'écart $(\widehat{y^+}, \widehat{y^-})$ du problème agrégé.

Proposition 6. *Soit $(\widehat{\lambda}, \widehat{y^+}, \widehat{y^-})$ une solution optimale de $(PMRSA_{\Omega,V})$, alors :*

$$\widehat{y_l^+}\widehat{y_l^-} = 0 \qquad \forall l \in L.$$

Démonstration. Supposons qu'il existe $l \in L$ tel que $\widehat{y_l^+}\widehat{y_l^-} \neq 0$. Cela signifie que $\widehat{y_l^+} > 0$ et $\widehat{y_l^-} > 0$. On va montrer qu'il existe une solution réalisable au problème de coût inférieur à la solution $(\widehat{\lambda}, \widehat{y^+}, \widehat{y^-})$. Le coût induit par ces variables est :

$$z = -\widehat{\delta_l^-}\widehat{y_l^-} + \widehat{\delta_l^+}\widehat{y_l^+}.$$

Posons :

$$\widehat{y'_l}^- = \max\{0, \widehat{y_l^-} - \widehat{y_l^+}\}$$
$$\widehat{y'_l}^+ = -\widehat{y_l^-} + \widehat{y_l^+} + \widehat{y'_l}^-.$$

Comme $-\widehat{y'_l}^- + \widehat{y'_l}^+ = -\widehat{y_l^-} + \widehat{y_l^+}$, la solution $(\widehat{\lambda}, \widehat{y'}^+, \widehat{y'}^-)$ est réalisable. Le coût induit par ces variables d'écart est :

$$z' = -\widehat{\delta_l^-}\widehat{y'_l}^- + \widehat{\delta_l^+}\widehat{y'_l}^+.$$

- Si $\widehat{y'_l}^- = \widehat{y_l^-} - \widehat{y_l^+}$, nous avons :

$$\begin{aligned} z' &= -\widehat{\delta_l}^- \widehat{y'}_l^- \\ &= -\widehat{\delta_l}^- \widehat{y_l}^- + \widehat{\delta_l}^- \widehat{y_l}^+ \\ &< -\widehat{\delta_l}^- \widehat{y_l}^- + \widehat{\delta_l}^+ \widehat{y_l}^+ \\ &< z, \end{aligned}$$

car $\widehat{\delta_l}^+ > \widehat{\delta_l}^-$ et $\widehat{y_l}^+ > 0$.

- Si $\widehat{y'}_l^- = 0$, nous avons :

$$\begin{aligned} z' &= \widehat{\delta_l}^+ \widehat{y'}_l^+ \\ &= -\widehat{\delta_l}^+ \widehat{y_l}^- + \widehat{\delta_l}^+ \widehat{y_l}^+ \\ &< -\widehat{\delta_l}^- \widehat{y_l}^- + \widehat{\delta_l}^+ \widehat{y_t}^+ l \\ &< z, \end{aligned}$$

car $\widehat{\delta_l}^+ > \widehat{\delta_l}^-$ et $\widehat{y_l}^- > 0$.
Le coût de la solution $(\widehat{\lambda}, \widehat{y'}^+, \widehat{y'}^-)$ est donc strictement inférieur au coût de la solution $(\widehat{\lambda}, \widehat{y}^+, \widehat{y}^-)$. Ceci contredit l'optimalité de $(\widehat{\lambda}, \widehat{y}^+, \widehat{y}^-)$. □

Cette proposition signifie que pour chaque partie l, on est toujours dans l'un des cinq cas suivant :

$$\begin{aligned} \widehat{y_l}^+ = 0 \text{ et } \widehat{y_l}^- &= 0 \\ \widehat{\varepsilon_l}^+ > \widehat{y_l}^+ > 0 \text{ et } \widehat{y_l}^- &= 0 \\ \widehat{\varepsilon_l}^+ = \widehat{y_l}^+ > 0 \text{ et } \widehat{y_l}^- &= 0 \\ \widehat{y_l}^+ = 0 \text{ et } \widehat{\varepsilon_l}^- > \widehat{y_l}^- &> 0 \\ \widehat{y_l}^+ = 0 \text{ et } \widehat{\varepsilon_l}^- = \widehat{y_l}^- &> 0. \end{aligned}$$

Analysons les arcs induits par les contraintes de compatibilité dans les trois premiers cas. Le raisonnement étant identique pour les deux derniers cas, il ne sera pas présenté.

Cas $\widehat{y_l^+} = 0$ et $\widehat{y_l^-} = 0$

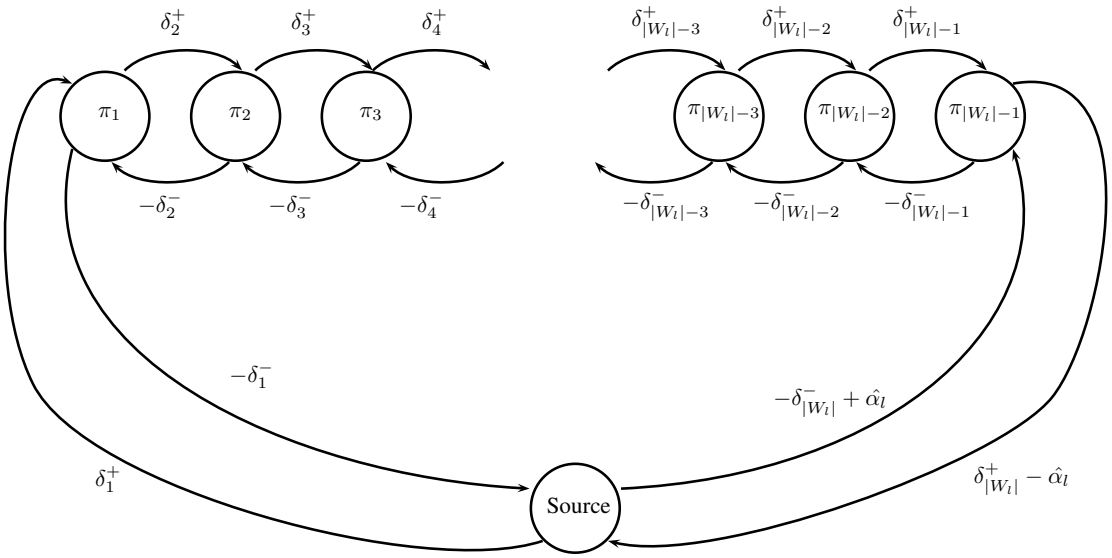


FIGURE 3.1 Graphe induit par les contraintes de complémentarité. Cas $\widehat{y_l^+} = 0$ et $\widehat{y_l^-} = 0$

Dans ce cas, pour tout $t \in W_l$, $y_t^+ = 0$ et $y_t^- = 0$. Comme $\varepsilon_t^+ > 0$ et $\varepsilon_t^- > 0$, les seules contraintes de compatibilité qui s'appliquent sont :

$$\begin{aligned} & \pi^l_{|W_l|} = \widehat{\alpha}_l \\ \forall t \in W_l \qquad & \pi^l_j - \pi^l_i \leq \delta_t^+ \\ \forall t \in W_l \qquad & \pi^l_j - \pi^l_i \geq \delta_t^-. \end{aligned}$$

Le graphe résultant est représenté à la figure 3.1.

Une remarque importante s'impose : l'optimalité du problème agrégé (3.7) implique :

$$\sum_{k=1}^{|W_l|} \delta_k^- \leq \widehat{\alpha}_l \leq \sum_{k=1}^{|W_l|} \delta_k^+.$$

De plus $\delta^+ > \delta^-$. On en déduit que tous les cycles dans ce graphe ont des longueurs positives ou nulles. Il suffit donc de regarder les chemins élémentaires pour initialiser

la matrice M. Ainsi les longueurs des plus courts chemins ne passant pas par le noeud source entre les noeuds $i, j \in \{1, ..., |W_l|\}$, $i < j$ sont données par :

$$m_{ij} = \sum_{k=i+1}^{j} \delta_k^+$$
$$m_{ji} = \sum_{k=i+1}^{j} -\delta_k^-.$$

Pour la longueur des plus courts chemins entre la source et le noeud $i \in \{1, ..., |W_l|\}$, on trouve :

$$m_{source,i} = \min \left\{ \sum_{k=1}^{i} \delta_k^+,\ \widehat{\alpha}_l + \sum_{k=i+1}^{|W_l|} -\delta_k^- \right\}$$
$$m_{i,source} = \min \left\{ -\widehat{\alpha}_l + \sum_{k=i+1}^{|W_l|} \delta_k^+,\ \sum_{k=1}^{i} -\delta_k^- \right\}.$$

En utilisant ces formules, on peut initialiser la matrice M pour les noeuds de la partie W_l en un temps $O(|W_l|^2)$.

Cas $\widehat{\varepsilon}_l^+ > \widehat{y}_l^+ > 0$ et $\widehat{y}_l^- = 0$

Dans ce cas, pour tout $t \in W_l$, on a $\varepsilon_l^+ > y_t^+ > 0$ et $y_t^- = 0$. Les seules contraintes de compatibilité qui s'appliquent sont :

$$\pi^l_{|W_l|} = \widehat{\alpha}_l$$
$$\forall t \in W_l \qquad \pi^l_j - \pi^l_i = \delta_t^+.$$

Le graphe résultant est représenté à la figure 3.2.

L'optimalité du problème agrégé 3.7 implique ici que $\widehat{\alpha}_l = \sum_{k=1}^{|W_l|} \delta_k^+$. Donc tous les cycles du graphe ont une longueur nulle. Il suffit donc de regarder les chemins

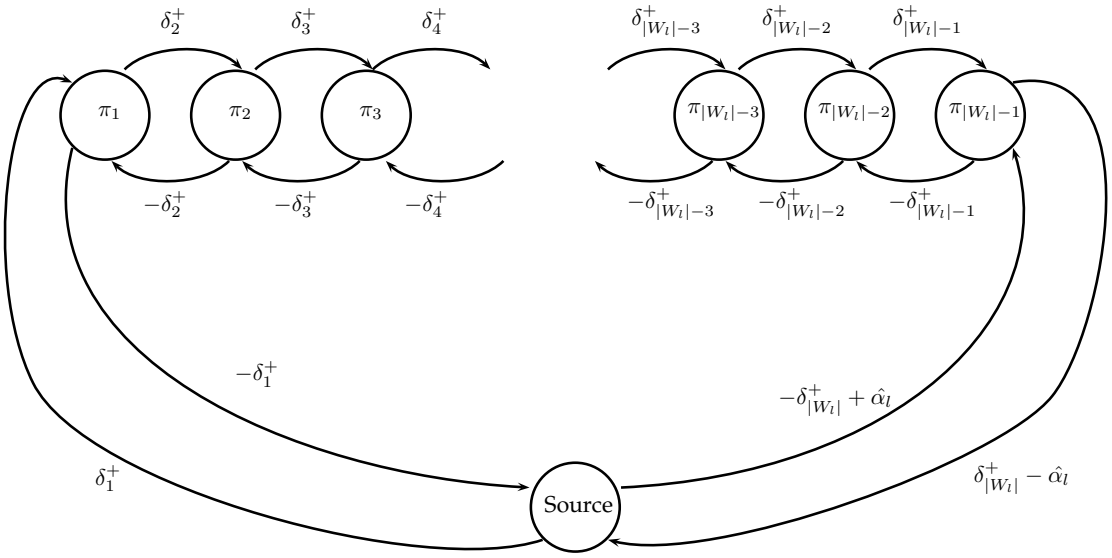


FIGURE 3.2 Graphe induit par les contraintes de complémentarité. Cas $\widehat{\varepsilon_l^+} > \widehat{y_l^+} > 0$ et $\widehat{y_l^-} = 0$

élémentaires pour initialiser la matrice M. Ainsi les longueurs des plus courts chemins ne passant pas par le noeud source entre les noeuds $i, j \in \{1, ..., |W_l|\}$, $i < j$ sont données par :

$$m_{ij} = \sum_{k=i+1}^{j} \delta_k^+$$
$$m_{ji} = \sum_{k=i+1}^{j} -\delta_k^-.$$

Pour la longueur des plus courts chemins entre la source et le noeud $i \in \{1, ..., |W_l|\}$, on obtient :

$$m_{source,i} = \min \left\{ \sum_{k=1}^{i} \delta_k^+,\ \widehat{\alpha}_l + \sum_{k=i+1}^{|W_l|} -\delta_k^+ \right\}$$
$$m_{i,source} = \min \left\{ -\widehat{\alpha}_l + \sum_{k=i+1}^{|W_l|} \delta_k^+,\ \sum_{k=1}^{i} -\delta_k^+ \right\}.$$

Or l'optimalité du problème agrégé implique :

$$\widehat{\alpha}_l = \widehat{\delta}_l^+ = \sum_{k=1}^{l} \delta_k^+.$$

Donc les deux termes de la fonction minimum sont égaux.

En utilisant ces formules, on peut initialiser la matrice M pour les noeuds de la partie W_l en un temps $O(|W_l|^2)$.

Cas $\varepsilon_l^+ = \widehat{y_l^+}$ et $\widehat{y_l^-} = 0$

Dans ce cas, pour tout $t \in W_l$, on a $\varepsilon_t^+ \geq y_t^+ > 0$ et $y_t^- = 0$. Les seules contraintes de compatibilité qui s'appliquent sont :

$$\pi^l_{|W_l|} = \widehat{\alpha}_l$$
$$\forall t \in W_l \text{ tel que } y_t^+ = \varepsilon_t^+ \qquad \pi^l_j - \pi^l_i \geq \delta_t^+$$
$$\forall t \in W_l \text{ tel que } y_t^+ < \varepsilon_t^+ \qquad \pi^l_j - \pi^l_i = \delta_t^+.$$

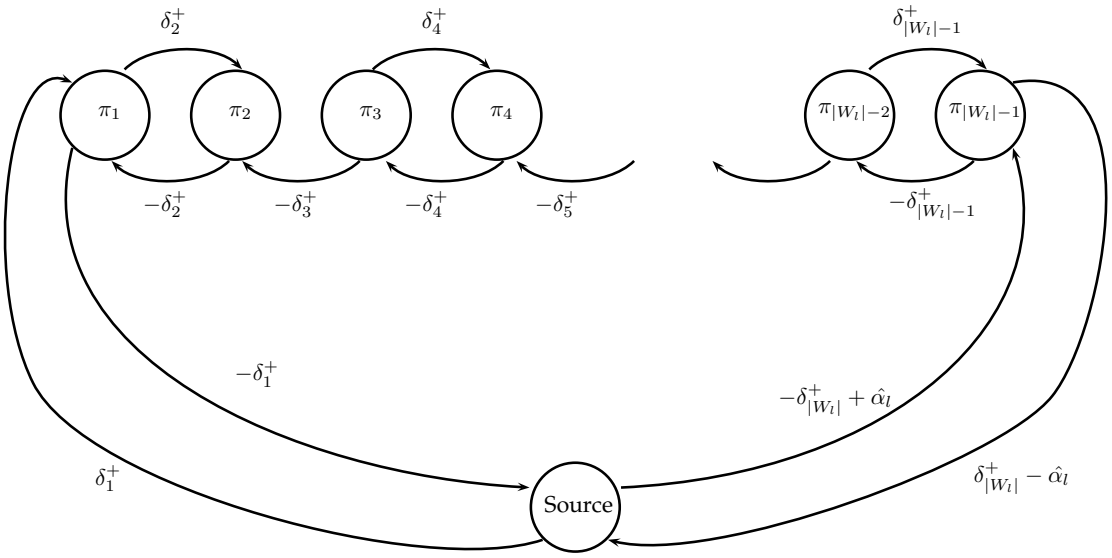


FIGURE 3.3 Graphe induit par les contraintes de complémentarité. Cas $\varepsilon_l^+ = \widehat{y_l^+}$ et $\widehat{y_l^-} = 0$

ce qui se reformule en :

$$\begin{aligned} & & \pi^l_{|W_l|} &= \widehat{\alpha}_l \\ \forall t \in W_l & & \pi^l_i - \pi^l_j &\leq -\delta^+_t \\ \forall t \in W_l \text{ tel que } y^+_t < \varepsilon^+_t & & \pi^l_j - \pi^l_i &\leq \delta^+_t. \end{aligned}$$

Le graphe résultant est représenté à la figure 3.3.

Pour $i, j \in \{1, ..., |W_l|\}$, $i < j$, le graphe comportera toujours un chemin allant de j vers i sans passer par la source, et éventuellement un chemin allant de i vers j sans passer par la source si la condition $y_t^+ < \varepsilon_t^+$ est vérifiée pour toutes les contraintes d'indice k avec $i < k \leq j$.

L'optimalité du problème agrégé (3.7) implique $\sum_{k=1}^{|W_l|} \delta_k^+ \leq \widehat{\alpha}_l$. Tous les cycles présents dans ce graphe ont donc une longueur positive ou nulle. Il suffit donc de regarder les chemins élémentaires pour initialiser la matrice M. Ainsi les longueurs des plus courts chemins ne passant pas par le noeud source entre les noeuds $i, j \in$

$\{1, ..., |W_l|\}$, $i < j$ sont données par :

$$m_{ij} = \begin{cases} \sum_{k=i+1}^{j} \delta_k^+ & \text{s'il existe un chemin de } i \text{ à } j \\ +\infty & \text{sinon,} \end{cases}$$

et

$$m_{ji} = \sum_{k=i+1}^{j} -\delta_k^+.$$

Intéressons-nous à la longueur du plus court chemin partant de la source vers le noeud $i \in \{1, ..., |W_l|\}$. Il existe toujours un chemin passant par le noeud $\pi_{|W_l|-1}$ de longueur $\widehat{\alpha}_l + \sum_{k=i+1}^{|W_l|} -\delta_k^+$. Il existe éventuellement un chemin passant par le noeud π_1 de longueur $\sum_{k=1}^{i} \delta_k^+$. Par optimalité du problème agrégé (3.7), on sait que :

$$\widehat{\alpha}_l \geq \widehat{\delta}_l^+ = \sum_{k=1}^{l} \delta_k^+,$$

ce qui implique :

$$\sum_{k=1}^{i} \delta_k^+ \leq \widehat{\alpha}_l + \sum_{k=i+1}^{|W_l|} -\delta_k^+ ?$$

S'il existe un chemin passant par le noeud π_1, celui-ci est donc le plus court. Finalement :

$$m_{source,i} = \begin{cases} \sum_{k=1}^{i} \delta_k^+ & \text{s'il existe un chemin de la source à } i \text{ contenant } \pi_1 \\ \widehat{\alpha}_l + \sum_{k=i+1}^{|W_l|} -\delta_k^+ & \text{sinon.} \end{cases}$$

Par un raisonnement analogue, la longueur du plus court chemin arrivant à la source en partant du noeud $i \in \{1, ..., |W_l|\}$ est :

$$m_{i,source} = \begin{cases} \sum_{k=i+1}^{|W_l|} \delta_k^+ - \widehat{\alpha}_l & \text{s'il existe un chemin de } i \text{ à la source contenant } \pi_{|W_l|-1} \\ -\sum_{k=1}^{i} \delta_k^+ & \text{sinon.} \end{cases}$$

Une implémentation intelligente permet encore d'initialiser la matrice M pour les noeuds de la partie W_l en un temps $O(|W_l|^2)$.

Conclusion de l'analyse du graphe

L'étude du graphe induit par les contraintes de complémentarité pour chaque partie a montré que tous les cycles présents ont une longueur positive ou nulle. De plus la matrice M peut être initialisée en un temps $O(|W_l|^2)$ pour chaque partie, soit au total $O(\sum_{l \in L} |W_l|^2) = O(|T|^2)$. Dans le graphe complet, les éventuels cycles de longueur négative contiendront donc forcément un arc induit par une colonne incompatible.

Chapitre 4

Application au problème d'horaires de véhicules avec dépôts multiples

L'approche présentée a été appliquée au problème d'horaires de véhicules avec dépôts multiples (ou MDVSP pour Multiple Depot Vehicle Scheduling Problem). C'est un problème qui est étudié depuis de nombreuses années (Bertossi *et al.*, 1987; Carpaneto *et al.*, 1989; Ribeiro et Soumis, 1994; Löbel, 1998; Hadjar *et al.*, 2006; Kliewer *et al.*, 2006; Desaulniers et Hickman, 2007). Il est rencontré par exemple par les services d'autobus pour le transport public urbain. Le transporteur assure à ses clients qu'un autobus sera présent aux horaires indiqués et qu'il effectuera un certain trajet. Il dispose d'un certain nombre de véhicules répartis dans quelques dépots. Chaque véhicule doit retourner dans son dépôt à la fin de la journée. Le transporteur doit construire des parcours pour ses véhicules afin d'effectuer tous les trajets promis aux clients tout en minimisant ses coûts de fonctionnement (en terme de carburant, d'usure des véhicules, etc).

Une première section présentera deux modélisations du MDVSP en programme linéaire en nombres entiers. Une deuxième section explicitera une relaxation du MDVSP. Cette relaxation sera utilisée non seulement pour trouver une solution initiale permettant d'initialiser la méthode d'agrégation dynamique de contraintes, mais aussi pour estimer une solution duale au MDVSP permettant d'initialiser les zones de confiance de la méthode de stabilisation des variables duales.

4.1 Modélisation du problème

Commençons par présenter la notation du MDVSP :

- Soit T l'ensemble des tâches (trajets) à réaliser. Chaque tâche $i \in T$ débute à l'heure h_i.

- Soit S un ensemble de stations. La tâche $i \in T$ consiste à partir de la station o_i pour rejoindre la station d_i.
- Soit K l'ensemble des dépôts. Le dépôt $k \in K$ possède n^k véhicules.
- Soit $i, j \in S \cup K$. Les temps et les coûts de parcours entre les lieux i et j sont connus et notés τ_{ij} et c_{ij}, respectivement.
- Notons C_f le coût fixe engendré par l'utilisation d'un véhicule.

4.1.1 Modèle à connexions directes

Le MDVSP peut se modéliser comme un problème de programmation en nombres entiers basé sur une approche multi-flot. Pour chaque dépôt $k \in K$, définissons le graphe orienté $G^k = (N^k, A^k)$.

- L'ensemble des noeuds N^k est composé d'un noeud par tâche $i \in T$, d'un noeud O^k représentant le dépôt comme point de départ des véhicules et d'un noeud D^k représentant le dépôt comme point de retour des véhicules.
- A^k contient un arc reliant les noeuds-tâches i et j si, après avoir effectué la tâche i, un véhicule peut effectuer la tâche j, c'est-à-dire si :

$$h_i + \tau_{o_i,d_i} + \tau_{d_i,o_j} \leq h_j.$$

  Dans ce cas, l'arc a un coût c_{d_i,o_j}. On dira que l'enchaînement des tâches i et j est *admissible*.
- A^k contient un arc reliant le noeud dépôt-origine O^k avec chaque tâche $i \in T$, de coût $c_{d_i,k} + C_f$.
- A^k contient un arc reliant chaque tâche $i \in T$ avec le noeud dépôt-destination D^k de coût $c_{d_i,k}$.

Introduisons les variables X_{ij}^k qui représentent la valeur du flot passant sur l'arc (i, j) du graphe G^k. Le MDVSP peut alors se formuler comme le programme linéaire en

nombres entiers suivant :

$$\begin{aligned}
\min \quad & \sum_{k\in K}\sum_{(i,j)\in A^k} c_{ij}X^k_{ij} & & \\
\text{sujet à : } & \sum_{k\in K}\sum_{j|(i,j)\in A^k} X^k_{ij} = 1 & \forall i \in T \qquad & (4.1)\\
& \sum_{j|(O^k,j)\in A^k} X^k_{O^k,j} \leq n^k & \forall k \in K \qquad & (4.2)\\
& \sum_{j|(i,j)\in A^k} X^k_{ij} = \sum_{j|(j,i)\in A^k} X^k_{ji} & \forall k \in K,\ \forall i \in N^k \qquad & (4.3)\\
& X^k_{ij} \in \mathbb{N} & \forall k \in K,\ \ \forall (i,j) \in A^k \qquad & (4.4)
\end{aligned}$$

Les contraintes (4.1) imposent que chaque tâche soit réalisée exactement un fois. Le nombre de véhicules disponibles par dépôt est contraint par (4.2). La conservation de flot, c'est-à-dire la conservation à chaque noeud du nombre de véhicules provenant d'un dépot, est assurée par (4.3). Enfin les contraintes (4.4) empêchent l'utilisation de fractions de véhicules et assure la non-négativité.

Les contraintes (4.2), (4.3) et (4.4) font apparaître, pour chaque dépôt k, la structure d'un problème de flot avec au plus n^k unités de flot circulant entre les noeuds O^k et D^k. Comme les arcs de ce problème de flot ne comportent aucune borne supérieure sur la capacité, il existe toujours une solution optimale composée de n^k unités de flot circulant sur un seul chemin reliant les noeuds O^k et D^k. Ce problème de flot est donc équivalent à un problème de plus court chemin. Cette structure nous permet d'appliquer la décomposition de Dantzig-Wolfe, et d'obtenir une formulation adéquate pour la génération de colonnes, comportant un sous-problème par dépôt.

Pour chaque dépôt $k \in K$, notons Δ_k l'ensemble des points extrêmes du polytope défini par les contraintes :

$$\begin{aligned}
&\sum_{j|(O^k,j)\in A^k} X^k_{O^k,j} = 1 \\
&\sum_{j|(j,D^k)\in A^k} X^k_{j,D^k} = 1 \\
&\sum_{j|(i,j)\in A^k} X^k_{ij} = \sum_{j|(j,i)\in A^k} X^k_{ji} && \forall i \in N^k \\
&X^k_{ij} \in \mathbb{N} && \forall (i,j) \in A^k.
\end{aligned}$$

Ce polyèdre est borné car $\|X\|_\infty = 1$. C'est donc un polytope. Chaque point extrême $x \in \Delta_k$ représente le parcours d'un véhicule partant et revenant au dépôt k. Le MDVSP se reformule en :

$$\begin{aligned}
\min_{\lambda} \quad & \sum_{x\in\Delta} c^T x \lambda_x \\
& \sum_{x\in\Delta} a_{ix}\lambda_x = 1 && \forall i \in T \\
& \sum_{x\in\Delta} a_{kx}\lambda_x \leq n^k && \forall k \in K \\
& \sum_{x\in\Delta} x\lambda_x \text{ entier} \\
& \lambda_x \geq 0 && \forall x \in \Delta,
\end{aligned}$$

où

- $\Delta = \cup_{k\in K}\Delta_k$
- $a_{ix} = 1$ si le parcours associé au point extrême x couvre la tâche $i \in T$, $a_{ix} = 0$ sinon.
- $a_{kx} = 1$ si le parcours associé au point extrême x débute et s'achève au dépôt $k \in K$, $a_{kx} = 0$ sinon.

4.1.2 Modèle espace-temps

Les sous-problèmes définis précedemment comportent beaucoup d'arcs. Ribeiro et Soumis (1994); Gintner *et al.* (2005) et Kliewer *et al.* (2006) décrivent une formulation

plus compacte via des réseaux espace-temps. Elle est présentée dans cette section. Les graphes des sous-problèmes étant construits de manière identique, dans la suite on parlera de manière générique *du* graphe du sous-problème.

Redéfinissons le graphe $G = (N, A)$ du sous-problème. N contient encore un noeud représentant le dépôt-origine et un noeud représentant le dépôt-arrivée. Pour chaque tâche $i \in T$, on introduit maintenant deux noeuds :

- un noeud de coordonnée (o_i, h_i) représentant la position initiale dans l'espace-temps d'un véhicule sur le point d'effectuer la tâche i.
- un noeud de coordonnée $(d_i, h_i + \tau_{o_i,d_i})$ représentant la position dans l'espace-temps d'un véhicule venant de terminer la tâche i.

Ces deux noeuds sont reliés par un arc, naturellement associé à la réalisation de la tâche i. Il est possible que deux tâches distinctes induisent deux noeuds ayant les mêmes coordonnées dans l'espace-temps. Dans ce cas, ces noeuds sont fusionnés en un seul.

Cette représentation des noeuds dans un espace-temps permet d'introduire facilement la notion d'arcs d'attente. Soit s une station. Chaque noeud ayant la coordonnée spatiale s est relié par un arc de coût nul au noeud suivant le plus proche dans le temps ayant la coordonnée spatiale s. Autrement dit, pour chaque noeud (s, θ), on crée un arc de coût nul partant du noeud (s, θ) vers l'unique noeud réalisant le minimum suivant, s'il existe :

$$\min\{\theta' \mid (s, \theta') \in N,\ \theta \leq \theta'\}.$$

Un petit exemple est présenté à la figure 4.1. Il y a trois stations et trois tâches. Les arcs d'attente sont les arcs horizontaux en pointillé. Les arcs correspondant aux tâches sont en trait plein. Ces arcs d'attente permettent de relier toutes les tâches finissant en s avec toutes les tâches partant de s, avec un nombre d'arcs réduit par rapport au modèle précédent.

Le graphe tel que défini permet aux véhicules d'effectuer une tâche i arrivant à la station d_i puis d'enchaîner avec une tâche j uniquement si j part de la station $d_i = o_j$. Il ne permet pas les passages à vide (« dead-head ») entre deux stations. Par exemple, dans la figure 4.1, un véhicule pourrait effectuer la tâche 1, se rendre à vide jusqu'à

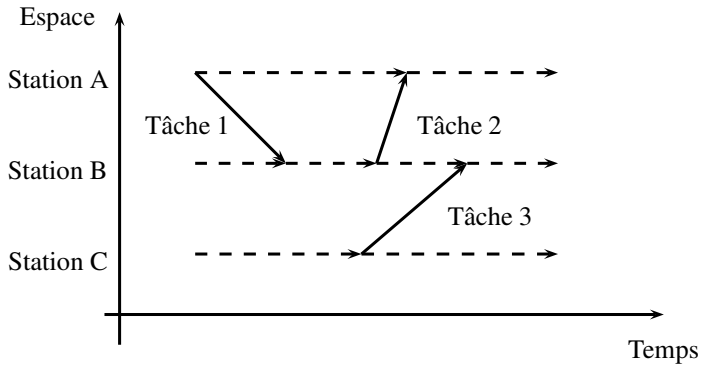


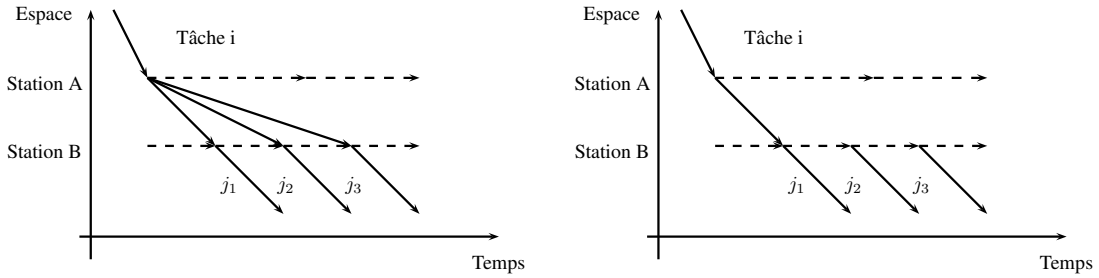
FIGURE 4.1 Réseau espace-temps. Tâches et arcs d'attente.

la station C puis effectuer la tâche 3. Voyons comment permettre ces changements de stations à vide sans utiliser trop d'arcs.

Prenons une tâche $i \in T$ finissant à la station A. Après la fin de la tâche i, un véhicule a le temps de se rendre à la station B avant l'heure de départ des tâches j_1, j_2 et j_3 (avec $h_{j_1} < h_{j_2} < h_{j_3}$). Plutôt que de relier naïvement la fin de la tâche i aux noeuds de départ de j_1, j_2 et j_3 (figure 4.2a), il suffit de relier la fin de la tâche i au noeud de départ de j_1 (figure 4.2b). En effet, les tâches j_2 et j_3 restent accessibles via les arcs d'attente de la station B. De manière plus formelle, pour chaque tâche $i \in T$ et chaque station $s \in S$ avec $s \neq d_i$, on ajoute un arc partant du noeud $(d_i, h_i + \tau_{o_i,d_i})$ vers l'unique noeud réalisant le minimum suivant, s'il existe :

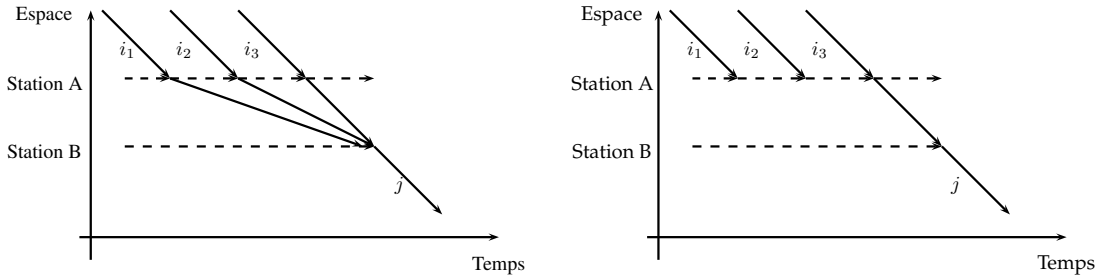
$$\min\{\theta' \mid (s, \theta') \in N,\ h_i + \tau_{o_i,d_i} + \tau_{d_i,s} \leq \theta'\}.$$

Le graphe défini jusqu'à maintenant permet toutes les connections, même celles nécessitant des passages à vide entre les stations. Cependant on peut encore réduire le nombre d'arcs tout en permettant toutes les connections. Supposons que les tâches i_1, i_2 et i_3, finissant à la station A ont été reliées à la tâche j de la station B par des arcs de passage à vide lors de l'étape précédente (figure 4.2a). Les arcs reliant i_1 et i_2 à j peuvent être enlevés car j reste accesible via les arcs d'attente de la station A et l'arc de passage à vide entre i_3 et j (figure 4.3a).



(a) Approche naïve (b) Meilleure approche

FIGURE 4.2 Arcs de passage à vide entre les stations



(a) Avant réduction (b) Après réduction

FIGURE 4.3 Réduction du nombre d'arcs de passage à vide entre les stations

Il ne reste plus qu'à ajouter des arcs reliant le noeud dépot-origine aux noeuds de départ des tâches, et des arcs reliant les noeuds de fin des tâches au noeud dépot-destination.

4.2 Solution initiale

L'utilisation de l'agrégation dynamique de contraintes nécessite une partition initiale des contraintes. Pour cela une heuristique peut permettre de construire une solution réalisable au MDVSP, ce qui induit naturellement une partition des contraintes. La stabilisation, elle, demande des valeurs duales pour « bien » initialiser les zones de confiance. Les variables duales d'une relaxation du MDVSP serviront de centres pour les zones de confiance.

Dans un premier temps, une relaxation du MDVSP sera présentée. Ensuite, à partir d'une solution optimale de cette relaxation, on construira une solution réalisable au MDVSP. Enfin les variables duales obtenues lors de la résolution de la relaxation serviront pour estimer une solution duale au MDVSP.

4.2.1 Relaxation du MDVSP en un problème de flot

La contrainte difficile du MDVSP est la nécessité pour les véhicules de revenir au dépôt initial. Montrons que la relaxation de cette contrainte tranforme le MDVSP en un problème de flot à coût minimum avec contraintes de capacité. Définissons le graphe $G = (N, A)$ ainsi :

- N comporte deux noeuds pour chaque tâche : un noeud « début de tâche » et un noeud « fin de tâche ».
- On demande un flot sortant de valeur 1 sur tous les noeuds « début de tâche » et on impose un flot entrant de valeur 1 sur tous les noeuds « fin de tâche ».
- De plus N comporte deux noeuds pour chaque dépot : un noeud « dépot-origine » et un noeud « dépot-destination ».
- Comme précédemment, A contient les arcs admissibles reliant les tâches entre elles (on peut utiliser le modèle à connexions directes ou le modèle espace-temps).
- A contient un arc reliant chaque noeud dépôt-origine à chaque noeud « début de tâche » et un arc reliant chaque noeud « fin de tâche » à chaque dépot-destination.
- Enfin pour chaque dépot k, A contient un arc reliant le noeud dépôt-destination au noeud dépôt-origine. Cet arc a une capacité maximale de n^k.

Les flots externes imposent la réalisation de chaque tâche. La capacité sur les arcs-retour entre les noeud-dépots réalise la contrainte de capacité sur le nombre de véhicules dans chaque dépot. La seule contrainte du MDVSP qui peut être violée est celle qui impose aux véhicules de revenir à leur dépôt de départ.

4.2.2 Obtention d'une partition initiale

Une solution au problème de flot à coût minimum sur le graphe ainsi défini se décompose, en « oubliant » les dépôts de départ et d'arrivée, en un ensemble de parcours couvrant chacun un certain nombre de tâches. Cet ensemble de parcours prove-

nant d'une solution entière, il couvre exactement une fois chaque tâche du problème. Il reste donc juste à « affecter » chaque parcours à un dépôt, tout en respectant les contraintes sur le nombre de véhicules présents pour chaque dépôt. Ce problème se modélise encore comme un problème de flot à coût minimum.

Indexons l'ensemble des parcours obtenus par l'ensemble P. Soit $G' = (N', A')$ un graphe défini ainsi :

- N' contient un noeud source avec un flot entrant de $|P|$ et un noeud puits avec un flot sortant de $|P|$.
- N' contient un noeud pour chaque parcours et un noeud pour chaque dépôt.
- Pour chaque parcours $p \in P$, A' contient un arc de coût nul et de capacité 1 reliant le noeud source au noeud-parcours p.
- Pour chaque dépôt $k \in K$, A' contient un arc de coût nul et de capacité n^k reliant le noeud-dépot k au noeud puits.
- A' contient un arc reliant chaque noeud-parcours p à chaque noeud-dépôt k, avec un coût correspondant au coût d'affectation du parcours p au dépôt k (c'est-à-dire le coût pour aller du dépôt k à la première station du parcours p ajouté au coût pour relier la dernière station du parcours p et le dépôt k).

Une solution du problème de flot à coût minimum sur le graphe G' affectera chaque parcours à un dépot exactement une fois (à cause des capacités unitaires sur les arcs sortant du noeud source). De plus à chaque dépôt, on n'affectera pas plus de parcours que le nombre de véhicules disponibles (à cause des capacités sur les arcs sortant des noeuds-dépôt). On a donc obtenu une solution réalisable au MDVSP.

Une solution réalisable définit naturellement une partition de l'ensemble des tâches : à chaque parcours $p_l \in P$, on associe la partie W_l comportant toutes les tâches couvertes par le parcours p_l. Comme l'ensemble des parcours $(p_l)_{l \in L}$ forme une solution réalisable, l'ensemble des parties $(W_l)_{l \in L}$ forme une partition de l'ensemble des tâches.

4.2.3 Obtention d'une solution duale initiale

Le premier problème de flot à coût minimum (section 4.2.1) est une relaxation du MDVSP. La solution duale associée à une de ses solutions optimales est utilisée comme estimation des variables duales du MDVSP. Dans ce problème de flot, chaque tâche est associée à deux noeuds : un noeud « début de tâche » et un noeud « fin de tâche ».

Notons π_i^d et π_i^f les variables duales du problème de flot associées respectivement au noeud « début de tâche » et au noeud « fin de tâche » de la tâche i. L'estimation de la variable duale α_i associée à la tâche i est donnée par $\alpha_i = -\pi_i^d + \pi_i^f$.

Chaque solution optimale au problème de flot induit une estimation des variables duales α. Oukil *et al.* (2007) et Rousseau *et al.* (2007) suggèrent qu'une combinaison convexe de ces estimations donne de meilleurs résultats. Pour obtenir plusieurs solutions optimales au problème de flot, la procédure suivante est utilisée : une première solution optimale est obtenue. Ensuite le problème est réoptimisé en imposant l'utilisation d'un arc ayant un flot nul dans la solution et un coût réduit de zéro. En effectuant cette opération avec tous les arcs ayant un flot nul dans la solution initiale et un coût réduit de zéro, on trouve éventuellement plusieurs solutions optimales.

Les méthodes de stabilisation et de stabilisation avec agrégation dynamique de contraintes ont été testées avec deux estimations initiales pour les variables duales initiales :

- L'estimation dite *première*, associée à la première solution du problème de flot.
- L'estimation dite *barycentrique*, qui est le barycentre obtenu en affectant un poids identique à toutes les solutions optimales obtenues avec la procédure décrite ci-dessus.

4.3 Générateur d'instances

Les calculs ont été menés sur des instances générées aléatoirement selon une procédure introduite par Carpaneto *et al.* (1989) puis modifiée par Oukil *et al.* (2007) pour fabriquer des problèmes plus dégénérés. Cette procédure est décrite en détail dans cette section.

Les instances sont générées selon trois paramètres : le nombre n de tâches à effectuer, le nombre m de dépôts, le nombre de jours d sur lesquels les tâches sont réparties.

- Le nombre de stations est un entier choisi aléatoirement dans l'intervalle $[n/2, n/3]$ selon une loi uniforme.

- Chaque station et chaque dépôt sont des points du plan euclidien, générés selon une loi uniforme dans un carré de taille 60x60.
- Le nombre de véhicules disponibles à chaque dépôt est un entier choisi aléatoirement dans l'intervalle $[3 + n/(3m), 3 + n/(2m)]$ selon une loi uniforme.
- Chaque trajet est assigné à une journée selon une loi uniforme sur les entiers de l'intervalle $[1, d]$. Les trajets sont répartis dans deux classes : les trajets long (60%) et les trajets courts (40%).
- Chaque trajet long commence et finit à la même station (choisie aléatoirement). L'heure de départ est choisie aléatoirement dans les 24h composant la journée choisie. La durée est choisie aléatoirement entre 180 minutes et 300 minutes.
- Pour les trajets courts, les stations de départ et d'arrivée i et j sont toutes deux choisies aléatoirement. La durée est choisie aléatoirement dans l'intervalle $[\tau_{ij}+5, \tau_{ij}+40]$ où τ_{ij} est la distance euclidienne entre les stations i et j. L'heure de départ est choisie uniformément dans la période de la journée à laquelle le trajet est assigné. Chaque trajet peut appartenir à :
  - la période creuse du matin entre 0h et 7h avec une probabilité de 0.1.
  - la période de pointe du matin entre 7h et 8h avec une probabilité de 0.15.
  - la période creuse de la journée entre 8h et 17h avec une probabilité de 0.5.
  - la période de pointe du soir entre 17h et 18h avec une probabilité de 0.15.
  - la période creuse du soir entre 18h et 24h avec une probabilité de 0.1.

Chapitre 5

Résultats et analyse

Cette section compare les performances de 5 méthodes appliquées à la résolution de la relaxation linéaire d'instances du MDVSP. Les cinq méthodes testées sont :

- la génération de colonnes
- l'agrégation dynamique de contraintes
- la stabilisation des variables duales
- l'agrégation dynamique de contraintes stabilisée
- la méthode du simplexe dual

Dans la suite, on se référera à ces méthodes en utilisant les acronymes *gencol*, *dca*, *stab*, *stab-dca* et *cplex* respectivement. L'implémentation utilisée pour la méthode de génération de colonnes est le logiciel GENCOL 4.5 utilisant CPLEX 10.1.1. Les méthodes dca, stab et stab-dca sont des surcouches de GENCOL 4.5 utilisant aussi CPLEX 10.1.1. La méthode cplex réfère au solveur commercial CPLEX dans sa version 10.1.1. Les méthodes stab et stab-dca ont été testées avec les deux stratégies de mise à jour des paramètres de stabilisation présentées dans la section 2.3.3 et avec les deux estimations pour la solution duale initiale décrites en 4.2.3. En l'absence de précision, les méthodes stab et stab-dca réfèrent à la stratégie expansion-contraction avec l'estimation barycentrique.

Ces méthodes ont été testées sur des instances du MDVSP, générées aléatoirement selon la procédure présentée à la section 4.3. Ces instances comportent 500, 1000, 1500, 2000, 2500 ou 3000 tâches, étalées sur 2 ou 5 jours et toujours 3 dépôts. Six instances, identifiées par leur germe, ont été générées pour chaque couple (nombre de tâches, nombre de jours).

Dans un premier temps, on verra que le modèle à connections directes n'est pas compétitif face au modèle espace-temps. Avant de s'attaquer aux instances de grande taille, on montrera que les méthodes gencol et dca sont beaucoup plus lentes que les méthodes stab et stab-dca sur les petites instances. Dans un deuxième temps, on

comparera les différentes options possibles pour les méthodes basées sur la stabilisation : l'estimation duale initiale et la procédure de mise à jour des paramètres de stabilisation. Enfin on verra quel est l'impact de l'agrégation de contraintes sur la méthode stab, et comment ces méthodes se comparent à cplex sur des instances de grande taille.

5.1 Comparaison du modèle à connections directes avec le modèle espace-temps

Cette section mesure l'impact du modèle espace-temps sur les perfomances. Chaque méthode a été testée sur 18 instances exprimées selon ces deux modélisations. Le tableau 5.1 présente le rapport entre le temps de calcul pour le modèle à connections directes et pour le modèle espace-temps. On constate que l'impact est important pour les méthodes stab et stab-dca avec une accélération de 3 en moyenne. L'impact est encore significatif pour les méthodes dca et gencol avec un facteur d'accélération moyen supérieur à 2. Les performances de la méthode cplex sont améliorées, mais de façon moins significative. Une analyse plus poussée permet d'expliquer les différences d'accélération entre les méthodes.

Pour les quatre méthodes basées sur la génération de colonnes (gencol, dca, stab, stab_dca), le réseau espace-temps a un impact surtout au niveau de la résolution des sous-problèmes. Comme il présente beaucoup moins d'arcs que le modèle à connections directes, la résolution des sous-problèmes est accélérée d'un facteur d'environ 4, comme le montre le tableau 5.2. Ces méthodes impliquant de nombreuses résolutions des sous-problèmes, l'impact est important. En effet, le tableau 5.3 indique que sur le modèle à connections directes, les méthodes stab et stab-dca passent 90% du temps à résoudre les sous-problèmes. Pour les méthodes dca et gencol, la résolution des sous-problèmes représente une part moins importante du temps de calcul. Elles profitent donc moins du réseau espace-temps.

La méthode cplex profite moins du changement de modèle que les autres méthodes. L'analyse est ici moins évidente. Soulignons tout de même que le modèle espace-temps a deux fois plus de contraintes que le modèle à connections directes, mais que le nombre de variables est bien moins important.

tâches	jours	germe	stab-dca	stab	dca	gencol	cplex
500	2	0	2.64	3.00	2.61	2.37	1.62
		1	3.12	3.49	3.04	2.34	1.13
		2	2.95	3.41	2.18	2.34	1.20
		3	3.56	3.30	2.94	2.56	2.25
		4	2.97	3.25	2.43	2.43	1.04
		5	2.95	3.02	2.78	2.38	3.08
500	5	0	2.30	2.99	1.23	1.89	1.17
		1	3.34	3.16	2.30	3.20	2.20
		2	3.28	3.08	1.16	2.03	0.94
		3	3.35	3.07	1.57	1.72	2.18
		4	3.17	3.32	2.44	1.97	2.99
		5	3.65	3.78	1.16	2.55	1.40
1000	2	0	3.09	2.94	2.79	1.91	1.36
		1	3.23	3.01	2.20	1.90	0.91
		2	3.15	2.84	2.08	1.99	0.91
		3	3.07	2.90	2.61	1.98	0.94
		4	3.32	3.21	2.25	1.92	0.84
		5	3.28	2.86	2.47	1.87	1.11
moyenne			3.13	3.15	2.23	2.19	1.51

TABLEAU 5.1 Rapport du temps de calcul du modèle à connections directes par rapport au modèle espace-temps.

tâches	jours	germe	stab-dca	stab	dca	gencol
500	2	0	3.47	3.92	3.38	3.85
		1	3.59	3.95	3.63	4.22
		2	3.57	4.30	3.66	4.19
		3	3.85	4.57	4.03	4.74
		4	3.79	4.52	3.78	4.44
		5	3.81	4.54	4.14	4.78
500	5	0	3.70	4.11	3.48	3.99
		1	3.60	4.50	4.23	4.47
		2	3.70	4.58	3.86	4.47
		3	3.86	4.63	4.32	4.77
		4	3.73	4.26	3.96	4.33
		5	3.78	4.44	4.02	4.47
1000	2	0	3.17	3.32	3.19	3.35
		1	3.62	3.70	3.54	3.83
		2	3.44	3.50	3.44	3.39
		3	3.44	3.61	3.49	3.64
		4	3.26	3.36	3.50	3.32
		5	3.34	3.78	3.53	3.74
moyenne			3.6	4.09	3.73	4.11

TABLEAU 5.2 Rapport du temps moyen de résolution des sous-problèmes du modèle à connections directes par rapport au modèle espace-temps.

tâches	jours	germe	stab-dca	stab	dca	gencol
500	2	0	92.10	89.86	92.82	78.56
		1	92.85	87.73	85.86	75.79
		2	94.14	91.98	76.25	77.08
		3	93.83	88.84	83.83	78.31
		4	91.04	88.01	82.23	76.86
		5	92.58	88.76	66.37	75.32
500	5	0	95.25	89.63	57.19	62.53
		1	95.65	89.30	44.40	52.55
		2	95.12	90.20	59.53	68.27
		3	91.72	85.56	55.18	67.43
		4	93.48	90.45	57.36	70.91
		5	95.92	91.28	51.12	59.49
1000	2	0	95.80	92.91	84.73	68.54
		1	94.53	89.87	83.02	64.26
		2	95.06	92.73	76.05	68.95
		3	95.23	92.69	86.04	69.47
		4	91.08	87.05	76.50	67.13
		5	94.70	90.82	85.72	66.29
moyenne			93.89	89.87	72.46	69.32

TABLEAU 5.3 Pourcentage du temps total passé à résoudre les sous-problèmes. Modèle à connections directes

En conclusion, le modèle espace-temps est préférable au modèle à connections directes, en particulier pour les méthodes basées sur la génération de colonnes. Dans toute la suite, seul le modèle espace-temps sera utilisé.

5.2 Comparaison de gencol, dca et stab

Avant de passer aux instances de plus grande taille, les méthodes dca et gencol ont été comparées à la stabilisation sur des instances modestes. Le tableau 5.4 montre que dca et gencol ne sont pas compétitifs par rapport à stab. Pour les instances sur 2 jours, stab est en moyenne 5 fois plus rapide que dca et 7 fois plus rapide que gencol. La différence est particulièrement frappante pour les instances sur 5 jours : ces facteurs deviennent 61 et 48, respectivement.

Le tableau 5.5 permet de comparer les méthodes plus en détail. La stabilisation est gagnante sur tous les points : le nombre d'itérations de génération de colonnes est réduit, ainsi que le temps de résolution moyen du problème maître et même des sous-problèmes. La réduction du temps consacré au problème maître était le but recherché. L'amélioration du temps de résolution des sous-problèmes est quant à elle plutôt inattendue. Le facteur d'accélération n'est pas très grand, mais l'amélioration est présente sur toutes les instances. On peut conclure que, pour ces instances, la stabilisation réduit les effets de la dégénérescence de manière très efficace.

tâches	jours	germe	dca	gencol
500	2	0	2.54	3.68
		1	4.67	6.73
		2	8.18	8.91
		3	3.48	4.14
		4	5.93	5.51
		5	8.36	7.40
moyenne 500 tâches			5.53	6.06

tâches	jours	germe	dca	gencol
500	5	0	99.73	60.17
		1	88.99	90.07
		2	46.22	33.25
		3	25.57	17.87
		4	18.92	11.28
		5	88.27	79.39
moyenne 500 tâches			61.28	48.67

1000	2	0	3.61	6.53
		1	6.31	8.91
		2	7.07	8.39
		3	3.45	5.68
		4	9.04	11.16
		5	5.55	8.92
moyenne 1000 tâches			5.84	8.26

1000	5	0	53.19	105.36
		1	74.09	1005.09
		2	44.20	72.28
		3	87.50	1272.07
		4	36.58	55.84
		5	109.15	268.37
moyenne 1000 tâches			67.45	463.17

moyenne globale	5.68	7.16

moyenne globale	64.37	255.92

TABLEAU 5.4 Rapport du temps de résolution pour les méthodes gencol et dca par rapport à la méthode stab.

			itérations mineures		temps moyen sous-problèmes		temps moyen problème maître	
tâches	jours	germe	dca	gencol	dca	gencol	dca	gencol
500	2	0	1.09	0.74	2.58	3.52	1.78	8.38
		1	2.39	1.79	2.07	2.56	1.77	5.74
		2	2.23	2.00	2.30	2.72	7.37	9.22
		3	1.42	0.94	2.30	2.92	2.75	7.20
		4	1.93	1.23	2.45	3.04	4.22	7.16
		5	2.51	1.91	2.28	2.60	4.95	5.86
1000	2	0	1.33	0.83	2.31	3.85	4.32	23.96
		1	2.03	1.27	2.32	3.23	5.08	16.49
		2	2.20	1.16	2.08	3.55	6.71	18.60
		3	1.24	0.75	2.41	3.76	4.15	21.40
		4	3.12	1.74	1.92	3.18	5.20	14.08
		5	2.03	1.19	2.28	3.50	4.06	19.08
moyenne			1.96	1.30	2.28	3.20	4.36	13.10
500	5	0	19.03	17.21	1.67	1.65	12.89	7.48
		1	23.17	29.53	1.33	1.28	8.87	6.61
		2	10.27	9.10	1.72	1.79	10.59	7.75
		3	5.60	5.50	1.76	1.83	7.95	4.97
		4	3.63	2.39	2.35	2.49	10.78	9.06
		5	12.31	16.33	1.85	1.78	21.42	13.14
moyenne			12.34	13.34	1.78	1.80	12.08	8.17

TABLEAU 5.5 Rapport du nombre d'itérations mineures, du temps de résolution moyen par itération mineure des sous-problèmes et du problème maître pour les méthodes gencol et dca par rapport à la méthode stab.

5.3 Effet du choix de l'estimation duale initiale

Intéressons-nous maintenant aux méthodes basées sur la stabilisation. Cette section essaye de caractériser l'impact de l'estimation duale initiale. Le tableau 5.6 compare les performances des méthodes stab et stab-dca (utilisant la stratégie de mise à jour expansion-contraction) lorsque la première estimation duale ou l'estimation barycentrique sont utilisées. L'estimation barycentrique est clairement plus efficace, en particulier sur les instances de 5 jours où le temps de calcul est réduit par un facteur 10. D'où provient cette différence de performance ? Pour répondre à cette question, nous avons tracé la résolution par la méthode stab d'une instance de 1000 tâches étalées sur 5 jours avec les deux estimations duales (figure 5.1). Cette figure montre l'évolution du temps cumulatif consacré au problème maître, aux sous-problèmes ainsi que la valeur de l'objectif du problème maître. Des indicateurs moyens sur l'ensemble des instances sont rapportés dans le tableau 5.7.

Intéressons-nous d'abord à la première itération majeure : celle-ci est beaucoup plus longue à résoudre avec la première estimation duale. De plus, à la fin de cette première itération majeure, de nombreuses variables duales sont strictement à l'intérieur de la zone de confiance. Enfin l'objectif est plus proche de la valeur optimale avec l'estimation barycentrique. Ces observations peuvent s'analyser ainsi : la première estimation duale est située près de la frontière du polyèdre dual. Il est donc fort probable que cette frontière intersecte la zone de confiance. La pénalisation incite à chercher une bonne solution à l'intérieur de la zone de confiance, mais ceci est rendu difficile par la présence de la frontière du polyèdre. Il faut générer toutes les contraintes duales (c'est-à-dire les colonnes du primal) qui intersecte la zone de confiance pour résoudre la première itération majeure, d'où de nombreuses itérations mineures. Ces contraintes empêchent les variables duales d'améliorer l'objectif en s'approchant de la frontière de la zone de confiance, d'où le grand nombre de variables duales strictement à l'intérieur et une valeur plus faible de l'objectif.

Regardons ensuite les itérations suivantes : les graphiques 5.1a et 5.1b nous montrent qu'avec l'estimation barycentrique l'objectif s'améliore beaucoup entre les premières itérations majeures. Avec la première estimation, le pas est beaucoup plus petit. D'où provient cet effet ? Tout d'abord, avec la première estimation, beaucoup de variables duales se situent strictement à l'intérieur de la zone de confiance à la fin de la première

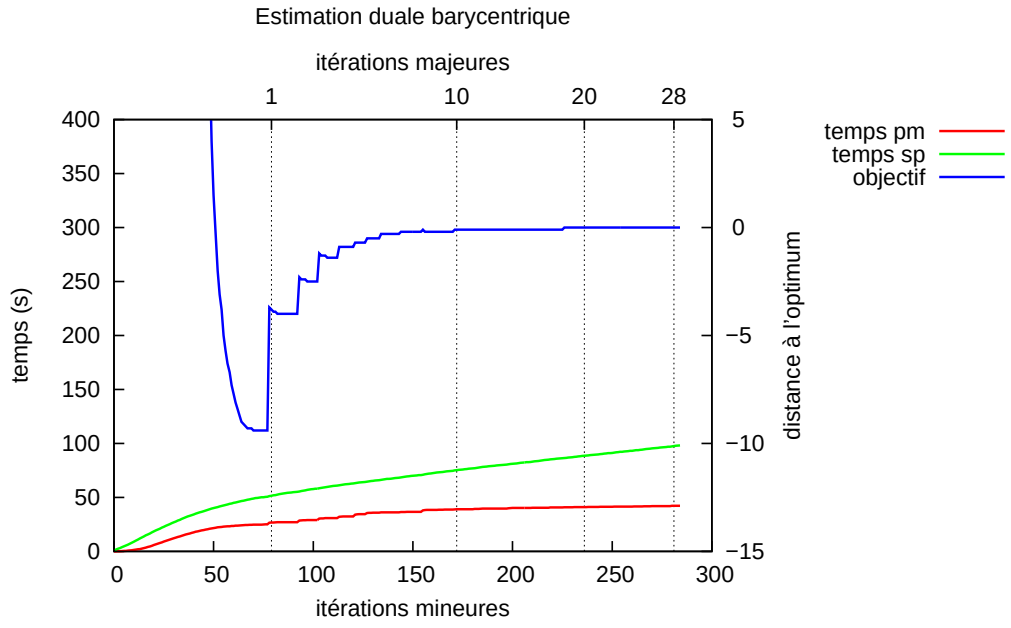
itération majeure. En conséquence, les pénalités sont augmentées sur beaucoup plus de variables. Il devient donc moins « rentable » de s'éloigner de la zone de confiance pour améliorer la fonction objectif. De plus, comme mentionné précédemment, beaucoup de variables duales sont « bloquées » par la frontière du polyèdre, empêchant d'améliorer l'objectif en restant dans la zone de confiance. A l'inverse avec l'estimation barycentrique, lors des premières itérations majeures, il y a moins de variables duales « bloquées » et moins de pénalités ont augmenté, ce qui permet de bonnes améliorations. Quand le nombre d'itérations majeures augmente, de plus en plus de variables duales se retrouvent « bloquées » et les pénalités augmentent de plus en plus, d'où des améliorations plus faibles de l'objectif.

Pour conclure, il semble qu'en utilisant l'estimation barycentrique, la stabilisation agit un peu comme une méthode de points intérieurs pour le dual. On part de l'intérieur du polyèdre et la pénalisation empêche de trop s'écarter de ce point de départ (et donc de trop s'approcher de la frontière du polyèdre). A l'inverse, avec la première estimation, il semble qu'on se déplace le long de la frontière. Cela cause deux effets négatifs : d'abord on rencontre beaucoup plus de points extrêmes pour le dual, c'est-à-dire plus de bases pour le primal ce qui rend le problème maître plus difficile à résoudre. Ensuite la proximité de la frontière empêchent d'améliorer l'objectif tout en restant dans la zone de confiance, ce qui amène de petites améliorations entre deux itérations majeures.

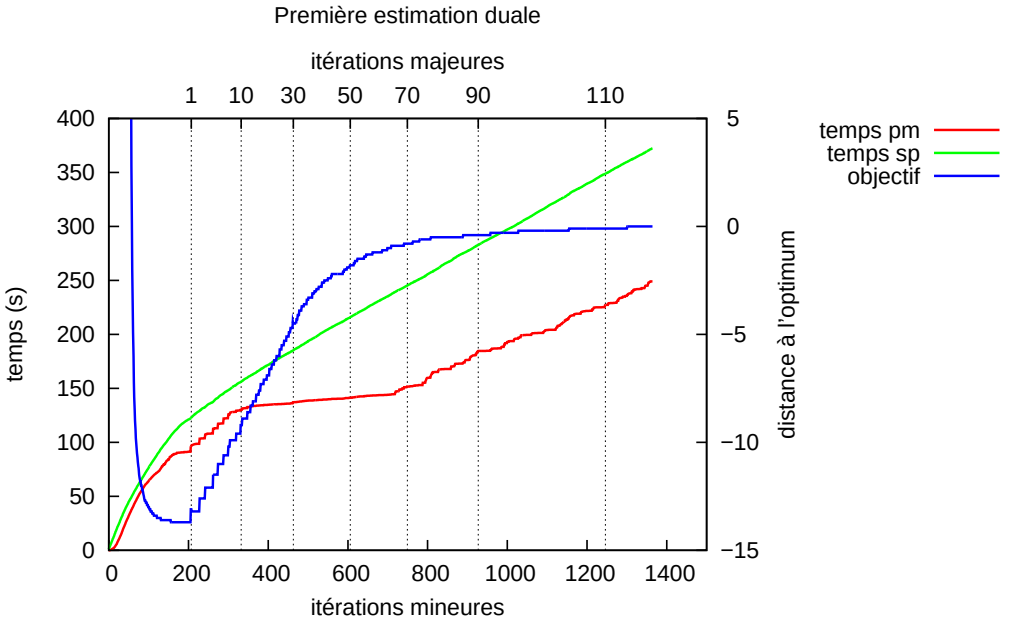
tâches	jours	germe	stab-dca	stab
500	2	0	2.00	2.12
		1	2.33	2.08
		2	2.44	2.78
		3	1.30	1.42
		4	1.97	2.05
		5	4.92	4.32
1000	2	0	1.80	1.87
		1	2.05	2.34
		2	2.01	2.22
		3	1.38	1.53
		4	1.97	2.45
		5	1.99	2.17
1500	2	0	1.61	1.82
		1	2.67	2.49
		2	1.65	1.85
		3	1.70	1.83
		4	2.39	2.36
		5	2.38	2.52
moyenne			2.14	2.23

tâches	jours	germe	stab-dca	stab
500	5	0	2.69	5.47
		1	6.61	6.01
		2	7.53	8.19
		3	3.62	4.50
		4	1.82	2.14
		5	13.16	15.07
1000	5	0	6.48	7.18
		1	11.81	14.41
		2	3.04	4.11
		3	9.48	11.99
		4	3.85	4.59
		5	11.09	12.36
1500	5	0	28.29	23.22
		1	3.35	3.26
		2	17.19	16.24
		3	17.16	17.69
		4	8.86	10.8
		5	20.97	13.32
moyenne			9.83	10.19

TABLEAU 5.6 Rapport du temps de résolution avec la première estimation duale par rapport à l'estimation barycentrique.



(a)



(b)

FIGURE 5.1 Instance de 1000 tâches sur 5 jours. Méthode stab. Évolution du temps de résolution cumulé du problème maître, des sous-problèmes ainsi que la valeur de l'objectif du problème maître.

		moyenne 2 jours	moyenne 5 jours
écart à la valeur optimale à la fin de la première itération majeure	stab	1.63	1.25
	stab-dca	1.63	1.25
nombre de variables duales strictement dans la zone de confiance à la fin de la première itération majeure	stab	3.19	3.52
	stab-dca	2.37	1.78
temps total pour résoudre la première itération majeure	stab	1.65	4.46
	stab-dca	1.69	4.21
distance à l'optimum à la fin de la seconde itération majeure	stab	2.85	3.19
	stab-dca	2.88	3.26
temps moyen pour la résolution du problème maître	stab	1.36	2.00
	stab-dca	2.00	6.80
temps moyen pour la résolution des sous-problèmes	stab	0.93	0.83
	stab-dca	0.92	0.90
nombre d'itérations mineures	stab	2.10	8.67
	stab-dca	2.00	7.46
nombre d'itérations majeures	Stab	2.25	4.26
	stab-dca	2.22	4.29

TABLEAU 5.7 Divers rapports entre les méthodes stab et stab-dca utilisées avec la première estimation duale par rapport à la même méthode utilisant l'estimation barycentrique.

5.4 Impact des stratégies de mise à jour des paramètres de stabilisation

Comparons les effets des deux stratégies de mise à jour des paramètres de stabilisation pour les méthodes stab et stab-dca en utilisant l'estimation duale barycentrique. Les résultats sont présentés au tableau 5.8. Ils indiquent que la stratégie d'expansion-contraction est meilleure pour les deux méthodes sur les instances plus dégénérées (sur 5 jours). Pour les instances moins dégénérées, les deux stratégies sont en moyenne équivalentes pour la méthode stab, et la stratégie d'expansion-seulement est légèrement avantageuse pour la méthode stab-dca. Avant d'analyser le comportement global sur toutes les instances, il est important de regarder un exemple pour bien comprendre le fonctionnement de la stabilisation. Les figures 5.2 et 5.3 donnent un aperçu du comportement induit par les deux stratégies. Ces figures représentent pour chacune des deux stratégies l'évolution du temps de résolution cumulé pour le problèms maître, pour les sous-problèmes ainsi que la valeur de l'objectif du problème maître lors de l'application des méthodes stab et stab-dca à la première instance de

2500 tâches étalées sur 5 jours.

En premier lieu, soulignons que le déroulement de la première itération majeure n'est pas influencé par le choix de la procédure de mise à jour. Cette observation est importante car cette première itération majeure représente une part importante du temps de résolution (46% en moyenne pour la méthode stab avec la stratégie expansion-contraction). Une fois passée la première itération majeure, la stratégie expansion-contraction effectue beaucoup d'itérations majeures qui sont résolues rapidement. À l'inverse, la stratégie expansion-seulement effectue peu d'itérations majeures et celles-ci sont difficiles. En particulier pour les dernières itérations majeures, le problème maître devient très long à résoudre. Cette analyse tend à confirmer l'intuition : plus la zone de confiance s'élargit, plus le nombre de bases non pénalisées augmente et donc plus le problème maître est dégénéré.

Cet exemple incite à penser que les meilleures performances de la stratégie expansion-contraction s'expliquent principalement par la réduction du temps consacré au problème maître : la contraction de la zone de confiance permet d'éviter la dégénérescence en fin de résolution. L'analyse de l'ensemble des instances présentée au tableau 5.9 confirme cette idée : la résolution du problème maître est 3 à 4 fois plus rapide sur les instances de 5 jours, mais l'accélération est faible sur les instances moins dégénérées de 2 jours. En contrepartie, la contraction implique de nombreuses itérations majeures, chacune étant sujette à l'effet « queue » : il faudra, à chaque itération majeure, quelques itérations mineures « inutiles » pour prouver l'optimalité (dans 45% des itérations majeures, la valeur est optimale dès le départ). Sur les instances de 5 jours, cet effet est largement compensé par la réduction de la dégénérescence du problème maître. En revanche sur les instances de 2 jours, les deux effets se compensent.

Remarquons enfin que la méthode stab-dca profite moins du changement de stratégie de mise à jour. Ceci s'explique par le fait que le problème maître est plus rapide à résoudre grâce à l'agrégation de contraintes. La proportion de temps qui lui est consacré est donc plus faible, d'où un impact moins important sur le temps total.

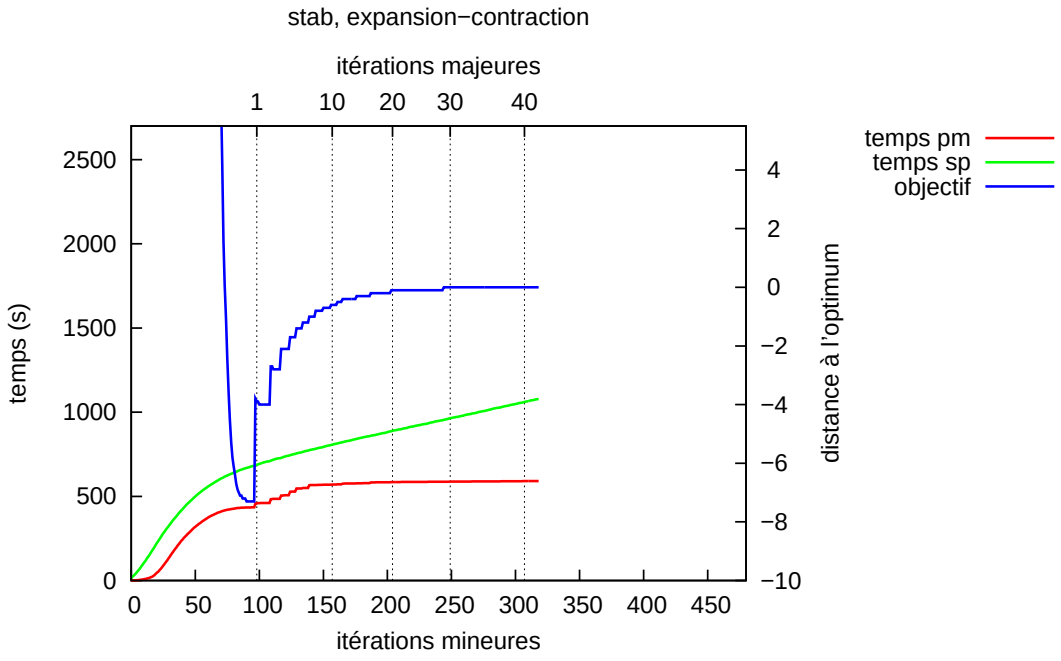
tâches	jours	germe	stab	stab-dca
500	2	0	0.87	0.72
		1	1.09	1.14
		2	1.05	1.12
		3	1.10	1.01
		4	1.38	1.12
		5	0.83	0.87
1000	2	0	1.03	0.88
		1	2.00	1.18
		2	1.31	1.35
		3	0.74	0.73
		4	1.31	1.22
		5	0.94	0.81
1500	2	0	0.81	0.73
		1	0.95	0.87
		2	0.82	0.73
		3	1.26	0.96
		4	1.05	0.81
		5	0.97	0.91
2000	2	0	0.84	0.72
		1	1.14	1.06
		2	0.89	0.77
		3	1.00	0.78
		4	1.04	0.85
		5	0.87	0.65
2500	2	0	0.99	0.68
		1	0.98	0.82
		2	0.92	0.87
		3	1.12	0.89
		4	1.01	0.87
		5	1.30	0.9
3000	2	0	1.03	0.88
		1	1.21	0.89
		2	1.29	0.95
		3	1.18	0.81
		4	1.25	0.92
		5	1.04	0.76
moyenne			1.07	0.90

tâches	jours	germe	stab	stab-dca
500	5	0	1.35	1.21
		1	1.48	1.30
		2	1.75	1.70
		3	1.12	1.20
		4	3.48	1.27
		5	1.87	1.39
1000	5	0	1.24	0.90
		1	1.29	1.06
		2	1.94	1.60
		3	1.50	1.31
		4	1.28	1.05
		5	1.38	1.31
1500	5	0	1.99	1.34
		1	1.70	0.96
		2	2.36	1.88
		3	2.89	1.35
		4	1.59	1.83
		5	2.49	1.89
2000	5	0	2.92	2.83
		1	1.32	0.95
		2	1.64	1.22
		3	2.55	1.94
		4	1.77	1.46
		5	3.12	2.06
2500	5	0	2.77	1.22
		1	3.25	2.87
		2	3.03	1.79
		3	1.97	1.19
		4	1.99	1.78
		5	1.90	1.58
3000	5	0	2.79	2.00
		1	2.88	1.82
		2	1.49	1.22
		3	3.02	1.61
		4	2.57	1.64
		5	2.81	1.85
moyenne			2.12	1.54

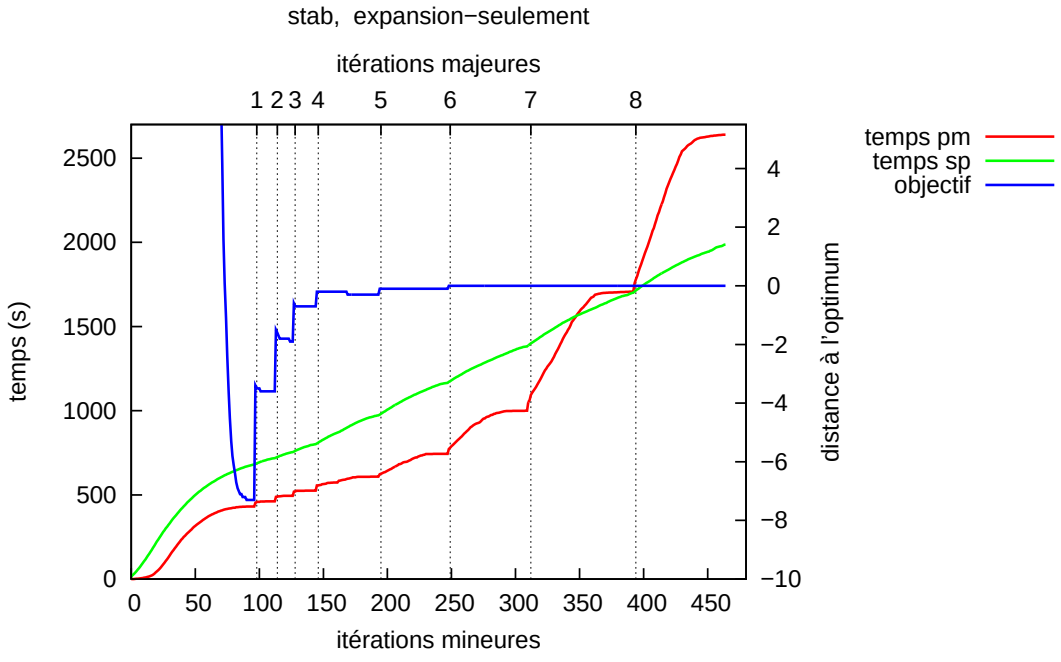
TABLEAU 5.8 Rapport du temps de résolution avec la stratégie expansion-seulement par rapport à la stratégie expansion-contraction.

		moyenne 2 jours	moyenne 5 jours
temps total	stab	1.07	2.12
	stab-dca	0.90	1.54
temps cumulé pour la résolution des sous-problèmes	stab	0.96	1.57
	stab-dca	0.86	1.34
temps cumulé pour la résolution du problème maître	stab	1.34	3.32
	stab-dca	1.25	4.02
nombre d'itérations mineures	stab	0.57	1.25
	stab-dca	0.56	1.02
nombre d'itérations majeures	stab	0.12	0.3
	stab-dca	0.14	0.32

TABLEAU 5.9 Moyennes de divers rapports entre la stratégie expansion-seulement par rapport à la stratégie expansion-contraction.

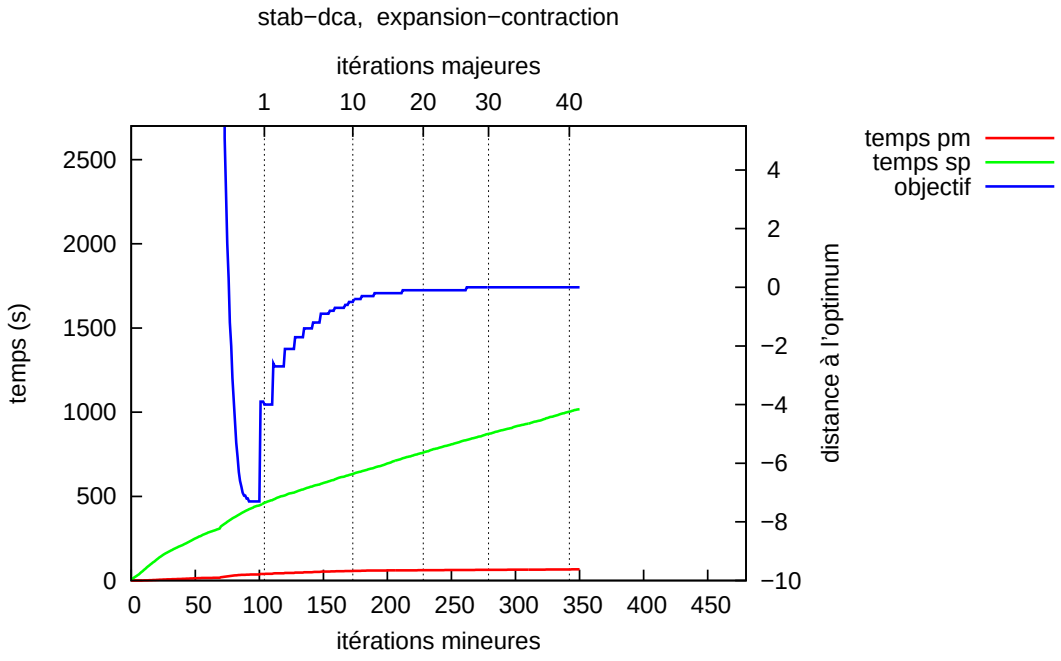


(a)

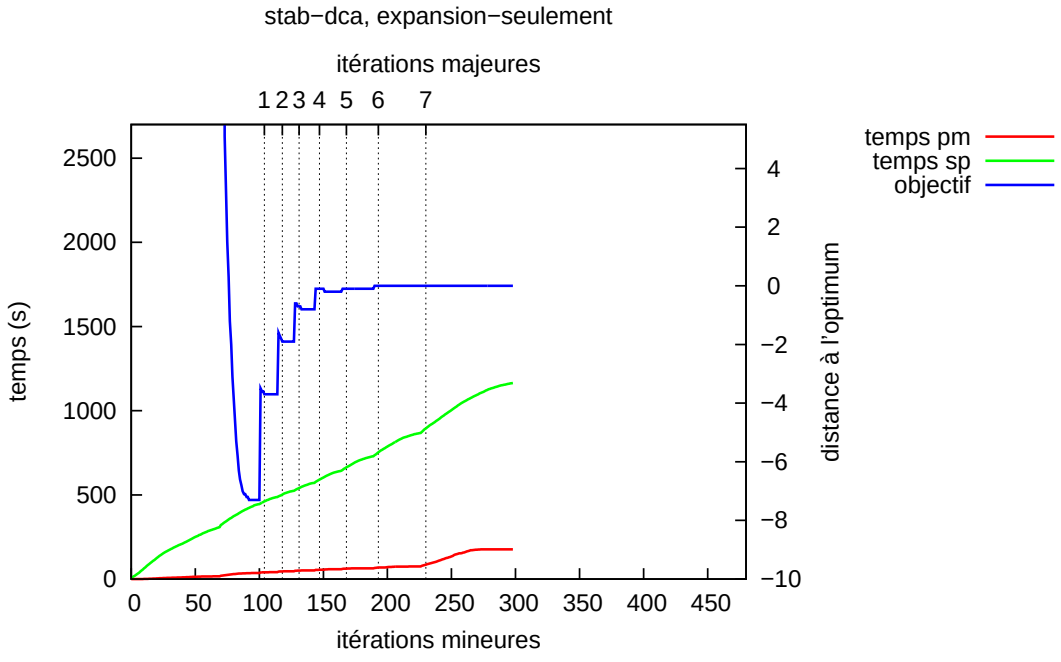


(b)

FIGURE 5.2 Instance de 2500 tâches sur 5 jours. Méthode stab. Évolution du temps de résolution du problème maître, des sous-problèmes ainsi que la valeur de l'objectif du problème maître.



(a)



(b)

FIGURE 5.3 Instance de 2500 tâches sur 5 jours. Méthode stab-dca. Évolution du temps de résolution du problème maître, des sous-problèmes ainsi que la valeur de l'objectif du problème maître.

5.5 Comparaison de stab et stab-dca

Enfin regardons quel est l'apport de l'agrégation dynamique de contraintes pour la méthode de stabilisation des variables duales. Le tableau 5.10 montre une amélioration légère des temps de calculs mais présente sur quasiment toutes les instances. L'accélération est plus importante lorsqu'on utilise la procédure de mise à jour expansion-seulement. Le tableau 5.11 et les figures 5.2 et 5.3 de la section précédente montrent que l'amélioration provient entièrement de la réduction du temps de calcul du problème maître. Cependant la résolution du problème maître ne représente que 30% du temps total avec la stratégie expansion-contraction et 45% avec la stratégie expansion-seulement, d'où les petits facteurs d'accélération sur le temps total. La procédure retenue pour la modification de la partition fait que le nombre de parties augmente très vite lors de la première itération majeure, puis augmente très peu jusqu'à la fin du processus (cf tableau 5.12).

Remarquons aussi que le nombre d'itérations mineures est plus ou moins identique avec les méthodes stab et stab-dca. L'amélioration globale des temps de calculs tient beaucoup à ce fait. En effet, si la méthode stab-dca nécessitait plus d'itérations mineures, elle serait globalement moins performante car la réduction du temps consacré au problème maître serait vite annulée par le temps nécessaire à la résolution des sous-problèmes supplémentaires. Pour préserver le nombre d'itérations mineures, les essais préliminaires ont montré qu'il était indispensable d'augmenter très rapidement la taille de la partition.

Dans un premier temps, on a voulu désagréger la partition uniquement lorsqu'il n'existait plus de colonnes compatibles de coût réduit négatif. Cette approche induisait un « effet queue » pour chaque taille de partition qui augmentait considérablement le nombre d'itérations mineures. Ensuite on a tenté de réagréger la partition lorsque celle-ci devenait trop grande (au cours de la première itération majeure, ou au début des itérations majeures suivantes). Ces essais nous ont montré que la méthode n'arrivait pas à trouver l'optimum d'une itération majeure sans désagréger fortement la partition. La raison de ce phénomène est peu claire. On peut tenter quelques explications :

- La structure des instances se prête peu à l'agrégation de contraintes. En effet ces instances comportent un grand nombre de stations, chacune étant reliées à

peu de tâches. Il est donc moins évident de trouver des groupes de tâches du type « multiples allers-retours entre deux stations ».

- La partition initiale provient d'une solution réalisable au problème non stabilisé. Il est possible que les variables de stabilisation, en permettant une surcouverture ou sous-couverture des tâches, change fortement les regroupements de tâches (voire casse complètement ces groupes).

Enfin on a tenté de réduire le temps de résolution des sous-problèmes en utilisant la méthode d'agrégation de contraintes bi-dynamique (Elhallaoui *et al.*, 2008). L'effet n'était pas positif pour plusieurs raisons. Tout d'abord, on a vu la nécessité de désagréger rapidement la partition. Or plus la partition est désagrégée, moins l'approche bi-dynamique est utile. De plus les sous-problèmes agrégés généraient souvent peu de colonnes, ce qui obligait à une deuxième résolution sur les sous-problèmes non-agrégés.

tâches	jours		expansion seulement	expansion contraction
500	2	0	1.13	0.94
		1	1.00	1.05
		2	0.79	0.85
		3	1.09	0.99
		4	1.06	0.87
		5	0.94	0.98
1000	2	0	1.18	1.00
		1	1.92	1.14
		2	0.95	0.98
		3	0.95	0.94
		4	1.03	0.96
		5	1.18	1.01
1500	2	0	1.20	1.08
		1	1.45	1.34
		2	1.25	1.11
		3	1.32	1.01
		4	1.48	1.15
		5	1.15	1.08
2000	2	0	1.35	1.17
		1	1.36	1.26
		2	1.31	1.14
		3	1.37	1.07
		4	1.49	1.21
		5	1.43	1.07
2500	2	0	1.63	1.12
		1	1.50	1.25
		2	1.31	1.24
		3	1.59	1.26
		4	1.56	1.35
		5	1.73	1.20
3000	2	0	1.42	1.22
		1	1.68	1.24
		2	1.93	1.42
		3	1.57	1.08
		4	1.91	1.41
		5	1.64	1.20
moyenne			1.36	1.12

tâches	jours		expansion seulement	expansion contraction
500	5	0	1.12	1.00
		1	1.55	1.36
		2	1.01	0.98
		3	1.27	1.37
		4	2.67	0.97
		5	1.33	0.99
1000	5	0	1.61	1.16
		1	1.77	1.45
		2	1.22	1.01
		3	1.48	1.29
		4	1.45	1.20
		5	1.19	1.14
1500	2	0	2.27	1.53
		1	2.43	1.37
		2	1.72	1.37
		3	2.36	1.10
		4	1.25	1.43
		5	1.79	1.36
2000	5	0	1.31	1.27
		1	2.80	2.01
		2	1.55	1.16
		3	1.55	1.17
		4	2.05	1.69
		5	1.83	1.20
2500	5	0	3.38	1.49
		1	1.48	1.31
		2	2.51	1.48
		3	3.00	1.81
		4	1.54	1.37
		5	1.82	1.52
3000	5	0	2.74	1.96
		1	2.38	1.51
		2	2.02	1.66
		3	2.75	1.47
		4	2.42	1.54
		5	2.11	1.39
moyenne			1.91	1.36

TABLEAU 5.10 Rapport du temps de résolution avec la méthode stab par rapport à la méthode stab-dca.

		moyenne 2 jours	moyenne 5 jours
temps total	expansion-seulement	1.36	1.91
	expansion-contraction	1.12	1.36
temps moyen pour la résolution du problème maître	expansion-seulement	3.19	4.40
	expansion-contraction	2.99	5.56
temps moyen pour la résolution des sous-problèmes	expansion-seulement	1.03	1.01
	expansion-contraction	0.94	1.07
nombre d'itérations mineures	expansion-seulement	1.00	1.21
	expansion-contraction	0.98	0.98
nombre d'itérations majeures	expansion-seulement	0.90	0.94
	expansion-contraction	1.00	1.02

TABLEAU 5.11 Divers ratios entre la méthode stab par rapport à la méthode stab-dca.

	taille de la partition à la fin de la première itération majeure		taille de la partition finale	
	expansion seulement	expansion contraction	expansion seulement	expansion contraction
moyenne 2 jours	77.36	77.36	89.42	85.44
moyenne 5 jours	60.86	60.86	74.58	69.38

TABLEAU 5.12 Taille de la partition en pourcentage du nombre de tâches. Méthode stab-dca.

5.6 Comparaison de cplex, stab et stab-dca

Pour terminer, regardons quels sont les facteurs d'accélération face au solveur cplex. Les résultats sont présentés dans les tableaux 5.13 et 5.14. On constate que les facteurs d'accélération sont importants : on atteint des facteurs de 30 sur les instances de 5 jours de grande taille. Sur quasiment toutes les instances, la méthode stab-dca est la plus rapide. Sur les instances de 2 jours, la procédure d'expansion-seulement est la meilleure avec un facteur d'accélération moyen de 10. L'expansion-contraction est préférable pour les instances de 5 jours avec en moyenne une amélioration d'un facteur 15.

Le tableau 5.15 présente les temps moyen de résolution des instances de chaque taille. Il est intéressant de constater que pour la méthode stab-dca utilisée avec l'expansion contraction, le temps moyen de résolution est quasiment identique pour les instances sur 2 jours et 5 jours. Cette méthode semble donc être efficace pour annuler les effets négatifs de la dégénérescence sur les temps de calcul.

tâches	jours	germe	stab-dca		stab	
			expansion contraction	expansion seulement	expansion contraction	expansion seulement
500	2	0	2.18	3.03	2.33	2.69
		1	2.98	2.61	2.85	2.61
		2	2.35	2.10	2.78	2.65
		3	1.08	1.07	1.08	0.99
		4	2.42	2.15	2.79	2.02
		5	1.47	1.70	1.50	1.80
moyenne 500 tâches			2.08	2.11	2.22	2.13
1000	2	0	3.07	3.49	3.06	2.96
		1	4.53	3.82	3.98	1.99
		2	3.69	2.74	3.78	2.88
		3	3.15	4.29	3.34	4.50
		4	4.99	4.07	5.19	3.96
		5	4.17	5.14	4.12	4.37
moyenne 1000 tâches			3.93	3.93	3.91	3.44
1500	2	0	5.32	7.31	4.94	6.10
		1	8.08	9.27	6.04	6.37
		2	5.58	7.63	5.03	6.11
		3	4.60	4.78	4.56	3.61
		4	6.83	8.39	5.96	5.65
		5	9.57	10.46	8.86	9.09
moyenne 1500 tâches			6.66	7.97	5.90	6.16
2000	2	0	9.16	12.70	7.85	9.39
		1	21.72	20.58	17.23	15.17
		2	9.63	12.45	8.48	9.53
		3	11.61	14.86	10.87	10.82
		4	11.36	13.38	9.38	8.98
		5	11.88	18.32	11.13	12.79
moyenne 2000 tâches			12.56	15.38	10.82	11.11
2500	2	0	16.05	23.53	14.27	14.42
		1	16.43	20.12	13.11	13.38
		2	21.32	24.37	17.17	18.60
		3	17.94	20.15	14.24	12.68
		4	17.87	20.47	13.22	13.09
		5	17.18	19.08	14.31	10.99
moyenne 2500 tâches			17.8	21.29	14.39	13.86
moyenne globale			8.61	10.14	7.45	7.34

TABLEAU 5.13 Facteur d'accélération des différentes méthodes face à cplex.

tâches	jours	germe	stab-dca		stab	
			expansion contraction	expansion seulement	expansion contraction	expansion seulement
500	5	0	3.02	2.50	3.01	2.22
		1	3.67	2.83	2.70	1.83
		2	4.84	2.85	4.94	2.82
		3	0.99	0.82	0.72	0.65
		4	0.71	0.56	0.73	0.21
		5	2.70	1.94	2.73	1.46
moyenne 500 tâches			2.65	1.92	2.47	1.53
1000	5	0	7.86	8.75	6.76	5.44
		1	11.69	11.03	8.04	6.23
		2	6.07	3.78	6.01	3.11
		3	9.50	7.26	7.35	4.91
		4	4.56	4.33	3.82	2.98
		5	9.42	7.17	8.27	6.01
moyenne 1000 tâches			8.18	7.05	6.71	4.78
1500	5	0	17.21	12.83	11.25	5.66
		1	16.95	17.60	12.33	7.24
		2	12.15	6.46	8.86	3.76
		3	12.20	9.02	11.06	3.82
		4	23.60	12.93	16.53	10.38
		5	11.10	5.88	8.17	3.28
moyenne 1500 tâches			15.53	10.79	11.37	5.69
2000	5	0	17.09	6.04	13.46	4.60
		1	50.38	52.86	25.02	18.89
		2	16.24	13.29	14.05	8.55
		3	12.81	6.61	10.92	4.27
		4	26.72	18.33	15.83	8.96
		5	23.6	11.47	19.61	6.28
moyenne 2000 tâches			24.47	18.10	16.48	8.59
2500	5	0	20.54	16.80	13.77	4.97
		1	21.66	7.56	16.59	5.11
		2	33.23	18.57	22.42	7.40
		3	37.20	31.30	20.58	10.42
		4	31.90	17.89	23.21	11.65
		5	20.06	12.69	13.24	6.96
moyenne 2500 tâches			27.43	17.47	18.3	7.75
moyenne globale			15.66	11.06	11.07	5.67

TABLEAU 5.14 Facteur d'accélération des différentes méthodes face à cplex.

tâches	jours	stab-dca		stab		cplex
		expansion contraction	expansion seulement	expansion contraction	expansion seulement	
500	2	21	20	20	21	43
1000		126	128	126	155	488
1500		342	286	380	373	2188
2000		677	533	776	737	8099
2500		1138	948	1403	1475	20205
3000		1942	1673	2401	2794	
500	5	18	24	20	39	38
1000		95	117	113	164	728
1500		280	431	376	828	4159
2000		653	1207	859	1961	13248
2500		1138	2042	1683	4171	30680
3000		1914	3226	3035	7752	

TABLEAU 5.15 Temps de résolution moyen en secondes pour chaque taille d'instance.

5.7 Conclusion de l'analyse des résultats

Les résultats montrent que le modèle espace-temps est plus efficace pour résoudre le problème d'horaires de véhicules avec dépôts multiples, car il permet de réduire substantiellement le nombre d'arcs dans les sous-problèmes. Ceci impacte positivement le temps de résolution. Les méthodes dca et gencol ne sont pas compétitives face à la méthode stab. Les mauvaises performances de la méthode dca peuvent s'expliquer par la structure des instances utilisées : celles-ci ne présente pas de groupes de tâches fortement susceptibles d'être présents dans une solution optimale. Il serait intéressant de comparer les méthodes stab et dca sur des problèmes plus propice à l'agrégation dynamique de contraintes, par exemple les problèmes de chauffeurs d'autobus.

Les performances de la méthode stab sont grandement impactées par l'estimation duale initiale choisie. La possibilité d'obtenir une « bonne » estimation duale initiale semble donc être un pré-requis à l'utilisation de cette méthode. Le choix de la stratégie de mise à jour a lui aussi un impact, même s'il est plus limité. Le critère de convergence (4) étant très peu restrictif, on risque de choisir une stratégie trop simple qui se révèlera peu performante. Mais cette grande liberté, couplée à une analyse du comportement de la méthode, peut mener à des stratégies spécifiques très

efficaces. Les stratégies choisies ici sont plutôt heuristiques, et une étude définissant de « bonnes » stratégies génériques rendrait sûrement la méthode de stabilisation plus simple à utiliser.

Le couplage de la stabilisation et de l'agrégation dynamique de contraintes a permis d'améliorer le temps de résolution du problème d'horaires de véhicules avec dépôts multiples. Cependant cette amélioration est très tributaire d'une bonne gestion de la désagrégation de la partition. On peut conjecturer que la méthode stab-dca serait plus robuste et plus efficace sur des problèmes présentant une structure plus propice à l'agrégation dynamique de contraintes.

Chapitre 6

Conclusion

Pour compléter ce mémoire, ce chapitre résume le travail qui a été effectué et présente quelques idées d'améliorations.

6.1 Synthèse des travaux

Le chapitre 2 a introduit la génération de colonnes qui est le cadre général de ce travail. Il s'est poursuivi sur une description détaillée des méthodes d'agrégation dynamique de contraintes et de stabilisation des variables duales. Avant de donner le cadre formel permettant de donner des descriptions complètes de ces deux méthodes, nous avons présenté une interprétation des idées sous-jacentes. Tout au long de cette présentation, nous nous sommes attachés à faire ressortir les difficultés rencontrées, qu'elles soient techniques ou conceptuelles. Le cadre de travail ayant été détaillé, le chapitre 3 a présenté la contribution théorique de cette maîtrise : comment coupler l'agrégation dynamique de contraintes et la stabilisation des variables duales. L'approche retenue est d'agréger dynamiquement les contraintes des problèmes stabilisés. Le principal défi de ce couplage était de désagréger l'information duale partielle obtenue par la résolution d'un problème de plus court chemin.

La méthode ayant été présentée, nous nous sommes intéressés à son application pratique sur le problème d'horaires de véhicules avec dépôts multiples. Le chapitre 4 a présenté ce type de problème et sa modélisation en termes de programmation linéaire. L'utilisation de l'agrégation dynamique de contraintes nécessite une solution primale initiale et la stabilisation des variables duales une estimation duale. Ces informations ont été obtenues grâce à une relaxation et une heuristique spécifiques au MDVSP, elles aussi détaillées au chapitre 4. Ce mémoire s'est achevé par la présentation des résultats obtenus lors de l'application de notre méthode sur des instances du MDVSP de tailles diverses. Le chapitre 5 compare les résultats obtenus avec les différentes méthodes

présentées et avec plusieurs paramétrisations de ces méthodes. Nous avons tenté d'analyser et d'expliquer au mieux les différences de résultats parfois marquantes.

Le premier apport de ce travail est avant tout théorique : il montre la possibilité de coupler deux méthodes efficaces mais différentes pour lutter contre les effets de la dégénérescence : la stabilisation des variables duales et l'agrégation dynamique de contraintes. Cependant ce couplage ne s'applique qu'aux problèmes de partitionnement d'ensemble. Étendre cette approche à une classe plus générale de problèmes semble plus difficile. Le deuxième aspect est plus pratique : il apporte une nouvelle amélioration des temps de calcul pour résoudre la relaxation linéaire des problèmes d'horaires de véhicules avec dépôts multiples par rapport à l'état de l'art. Le facteur d'accélération par rapport à la méthode de stabilisation n'est pas énorme, 2 au maximum, mais c'est déjà significatif pour les instances de grande taille. Le dernier aspect est plus exploratoire : il tente d'apporter des explications sur les différents comportement observés des méthodes testées. Les hypothèses ainsi obtenues tentent de donner une meilleure compréhension du fonctionnement des méthodes qui pourrait amener à de nouvelles idées d'amélioration. L'analyse conduite ici nous a d'ailleurs amené à formuler quelques propositions.

6.2 Idées d'amélioration

Au cours de l'interprétation du fonctionnement des méthodes et lors de l'analyse des résultats, quelques idées d'améliorations sont apparues.

- L'agrégation dynamique de contraintes permet de réduire substantiellement le temps de résolution du problème maître. Une amélioration supplémentaire des temps de calcul passera par la réduction du temps consacré aux sous-problèmes ou par une diminution du nombre d'itérations mineures.
- La comparaison des différentes estimations duales a montré qu'il est préférable de se situer à « l'intérieur » du polyèdre dual. Cela incite à penser qu'il serait intéressant d'utiliser la procédure de désagrégation des variables duales pour obtenir plusieurs solutions duales et travailler avec une combinaison convexe de celles-ci.
- Il serait aussi intéressant d'essayer d'autres partitions initiales. Ici la partition provient de la première solution réalisable rencontrée via l'heuristique. Peut-

être pourrait-on trouver une partition qui donne une information primale qui soit complémentaire de l'estimation duale initiale. Plus précisément, la stabilisation des variables duales définit un ensemble de directions non pénalisées pour la fonction objectif des sous-problèmes. Elle amène donc à générer les points extrêmes se situant dans ces directions. Est-il possible de définir une partition qui est compatible pour les colonnes associées à ces points extrêmes ?

- On peut remarquer qu'on effectue plusieurs itérations majeures, mais qu'une borne inférieure sur la valeur optimale du problème est obtenue dès la fin de la première itération majeure. Cette borne augmente peu lors des itérations majeures subséquentes. Dans une optique de réduction des temps de calcul, peut-être serait-il intéressant de se contenter de cette borne dans un schéma d'énumération implicite (branch & bound).

Références

AHUJA, R., MAGNANTI, T. et ORLIN, J. (1993). Network Flows : Theory, Algorithms and Applications, 147–150.

BARNHART, C., JOHNSON, E., NEMHAUSER, G., SAVELSBERGH, M. et VANCE, P. (1998). Branch-and-price : Column generation for solving huge integer programs. *Operations Research*, 46, 316–329.

BERTOSSI, A., CARRARESI, P. et GALLO, G. (1987). On some matching problems arising in vehicle scheduling models. *Networks*, 17, 271–281.

CARPANETO, G., DELL'AMICO, M., FISCHETTI, M. et TOTH, P. (1989). A branch and bound algorithm for the multiple depot vehicle scheduling problem. *Networks*, 19, 531–548.

CHARNES, A. (1986). Optimality and degeneracy in linear programming. *Econometrica*.

DANTZIG, G. et WOLFE, P. (1960). Decomposition principle for linear programs. *Operations Research*, 8, 101–111.

DESAULNIERS, G., DESROSIERS, J. et SOLOMON, M. (2005). *Column generation*. Springer NY.

DESAULNIERS, G. et HICKMAN, M. (2007). Public Transit. *Handbooks in Operations Research and Management Science*, 14, 69–127.

DESAULNIERS, G., J., D., IOACHIM, I., SOLOMON, M., SOUMIS, F. et VILLENEUVE, D. (1998). A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. *Dans T. Crainic et G. Laporte, Editeurrs, Fleet management and logistics, Kluwer, Norwell*, 57–93.

DESROSIERS, J., DUMAS, Y., SOLOMON, M. et SOUMIS, F. (1995). Time constrained routing and scheduling. *Handbooks in operations research and management science*, 8, 35–139.

DU MERLE, O., VILLENEUVE, D., DESROSIERS, J. et HANSEN, P. (1999). Stabilized column generation. *Discrete Mathematics*, 194, 229–237.

ELHALLAOUI, I., DESAULNIERS, G., METRANE, A. et SOUMIS, F. (2008). Bi-dynamic constraint aggregation and subproblem reduction. *Computers & Operations Research*, 35, 1713–1724.

ELHALLAOUI, I., METRANE, A., SOUMIS, F. et DESAULNIERS, G. (2010). Multi-phase dynamic constraint aggregation for set partitioning type problems. *Mathematical Programming*, 123, 345–370.

ELHALLAOUI, I., VILLENEUVE, D., SOUMIS, F. et DESAULNIERS, G. (2005). Dynamic aggregation of set partitioning constraints in column generation. *Operations Research*, 53, 632–645.

GAL, T. (1993). Selected bibliography on degeneracy. *Annals of Operations Research*, 46-47, 1–7.

GEOFFRION, A. (1974). Lagrangean relaxation for integer programming. *Mathematical Programming Study*, 2, 82–114.

GINTNER, V., KLIEWER, N. et SUHL, L. (2005). Solving large multiple-depot multiple-vehicle-type bus scheduling problems in practice. *OR Spectrum*, 27, 507–523.

GÜLER, O. (1991). On the convergence of the proximal point algorithm fo convex minimization. *SIAM Journal on Control and Optimization*, 29, 403–419.

HADJAR, A., MARCOTTE, O. et SOUMIS, F. (2006). A branch-and-cut algorithm for the multiple depot vehicle scheduling problem. *Operations Research*, 54, 130.

KLIEWER, N., MELLOULI, T. et SUHL, L. (2006). A time–space network based exact optimization model for multi-depot bus scheduling. *European Journal of Operational Research*, 175, 1616–1627.

LÖBEL, A. (1998). Vehicle scheduling in public transit and Lagrangean pricing. *Management Science*, 44, 1637–1649.

LÜBBECKE, M. et DESROSIERS, J. (2005). Selected Topics in Column Generation. *Operations Research*, 53, 1007–1023.

MARSTEN, R., HOGAN, W. W. et BLANKENSHIP, J. (1975). The Boxstep Method for Large-Scale Optimization. *Operations Research*, 23, 389–405.

OUKIL, A., BEN AMOR, H., DESROSIERS, J. et ELGUEDDARI, H. (2007). Stabilized column generation for highly degenerate multiple-depot vehicle scheduling problems. *Computers & Operations Research*, 34, 817–834.

PAN, P. (1998). A basis-deficiency-allowing variation of the simplex method for linear programming. *Computers & Mathematics with Applications*, 36, 33–53.

RIBEIRO, C. et SOUMIS, F. (1994). A column generation approach to the multiple-depot vehicle scheduling problem. *Operations Research*, 41–52.

ROCKAFELLAR, R. T. (1976). Monotone Operators and the Proximal Point Algorithm. *SIAM Journal on Control and Optimization*, 14, 877.

ROGERS, D. F., PLANTE, R. D., WONG, R. T. et EVANS, J. R. (1991). Aggregation and Disaggregation Techniques and Methodology in Optimization. *Operations Research*, 39, 553–582.

ROUSSEAU, L. M., GENDREAU, M. et FEILLET, D. (2007). Interior point stabilization for column generation. *Operations Research Letters*, 35, 660–668.

SHETTY, C. et TAYLOR, R. W. (1987). Solving large-scale linear programs by aggregation. *Computers & Operations Research*, 14, 385–393.

TERLAKY, T. (1993). Pivot rules for linear programming : A survey on recent theoretical developments. *Annals of Operations Research*, 46, 203–233.