Titre: Title:	Une approche de génération de colonnes pour un problème de localisation-affectation avec contraintes de capacités dans un contexte de déneigement urbain
Auteur: Author:	Sylvain Maréchal
Date:	1997
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Maréchal, S. (1997). Une approche de génération de colonnes pour un problème de localisation-affectation avec contraintes de capacités dans un contexte de déneigement urbain [Master's thesis, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/6684/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/6684/
Directeurs de recherche: Advisors:	Brigitte Jaumard
Programme: Program:	Unspecified

UNIVERSITÉ DE MONTRÉAL

UNE APPROCHE DE GÉNÉRATION DE COLONNES POUR UN PROBLÈME DE LOCALISATION-AFFECTATION AVEC CONTRAINTES DE CAPACITÉS DANS UN CONTEXTE DE DÉNEIGEMENT URBAIN

SYLVAIN MARÉCHAL

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)
AOÛT 1997

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé:

UNE APPROCHE DE GÉNÉRATION DE COLONNES POUR UN PROBLÈME DE LOCALISATION-AFFECTATION AVEC CONTRAINTES DE CAPACITÉS DANS UN CONTEXTE DE DÉNEIGEMENT URBAIN

présenté par: MARÉCHAL Sylvain

en vue de l'obtention du diplôme de: Maîtrise ès sciences appliquées

a été dûment accepté par le jury d'examen constitué de :

M. SAVARD Gilles, Ph.D., président

Mme JAUMARD Brigitte, T. Doct., T.Hab., membre et directrice de recherche

M. HANSEN Pierre, D.Agr., membre

À Nathalie et Marie Rose.

"Aux rêveurs .... aux railleurs ..." (V. de L'I. A.)

REMERCIEMENTS

Je voudrais d'abord remercier Brigitte Jaumard pour la confiance dont elle a témoigné à mon égard en me confiant ce travail. Je remercie également toutes les personnes qui m'ont aidé et encouragé pendant toute la durée de mon séjour au GERAD. En particulier, je suis reconnaissant à Stéphane Alarie, Tsévi Vovor, Pascal Adjakplé et Olivier du Merle pour leurs aide, conseils et encouragements. Je remercie aussi tout le personnel du GERAD.

RÉSUMÉ

Dans un contexte de déneigement urbain, on considère le problème qui consiste à choisir les sites de déversement de neige et à leur affecter les secteurs de déneigement de manière à minimiser le coût total des opérations. Un modèle de génération de colonnes (dans lequel des contraintes de capacité de volume et horaire sont définies pour les sites) est utilisé. Un algorithme exact, qui combine la méthode de génération de colonnes et une procédure d'optimisation par séparation et évaluation progressive, est proposé pour le résoudre. Le problème auxiliaire (qui consiste à déterminer la colonne de coût réduit minimum) est un problème de sac-à-dos bidimensionnel avec des contraintes additionnelles résultant de la règle de division.

Notre méthode de résolution a été testée sur un ensemble de 11 classes de problèmes, chacune étant définie par un certain nombre de secteurs et de sites. Les résultats semblent valider l'approche de génération de colonnes pour la résolution de ce problème.

ABSTRACT

In the context of urban snow removal and disposal operations, we consider the problem of choosing snow disposal sites and assigning to them the snow from all the snow removal districts of a city in order to minimize the total operations cost. A column generation model is used, in which storage capacity and storage rate at the snow disposal sites are taken into account. An exact algorithm, which combines the column generation method with branch-and-bound, is proposed to solve it. The auxiliary problem (which consists in determining the column of minimum marginal cost) is a bidimensional knapsack problem, with additional constraints after branching.

The solution method has been tested on 11 classes of problems, each of which being defined by a certain number of districts and sites. The results show that a column generation approach is justified for the solution of this problem.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

INTRODUCTION

Plusieurs problèmes relevant de la recherche opérationnelle peuvent être rencontrés dans la planification d'un système de déneigement urbain. Campbell et Langevin [3] fournissent une liste de ces problèmes et les décrivent brièvement. Ils se sont plus particulièrement intéressés à l'un d'eux, central pour la planification du déneigement, pour lequel ils ont proposé un modèle: il s'agit du problème de localisation-affectation des sites de déversement de la neige. Ce modèle est une version augmentée (par l'ajout de l'aspect "localisation") d'un modèle qu'ils présentent dans [2], lui-même généralisant un modèle de Leclerc [13]. Campbell et Langevin n'ont pas tenté de résoudre optimalement ce modèle. Ils suggèrent plutôt une méthode incorporant une heuristique présentée dans [2]. Notre objectif est de proposer une méthode exacte de résolution pour ce problème, que nous nommerons PLACC (Problème de Localisation-Affectation avec Contraintes de Capacités pour les sites de déversement), en faisant appel à la programmation linéaire généralisée (génération de colonnes). La présentation formelle du problème sera faite au chapitre 1. Nous en donnons cependant déjà ici une brève description, de manière à bien situer notre sujet.

Un centre urbain est partitionné en *secteurs* de déneigement. Chaque secteur est caractérisé par un volume de neige à transporter et un niveau de service. Il existe un certain nombre de *sites* où il serait possible de déverser de la neige. Chaque site est caractérisé, pour sa part, par une capacité de volume et une capacité horaire. Les opérations de déneigement s'effectuent à peu près simultanément dans tous les secteurs. Ces opérations consistent à faire charger la neige (préalablement tassée) dans des camions par une souffleuse. Un camion rempli est dirigé vers un site de

déversement pendant qu'un autre le remplace. Chaque secteur est affecté à un seul site; un camion dans un secteur est donc dirigé vers le site auquel le secteur est affecté. Le problème consiste alors à déterminer les sites qui devraient être utilisés pour le déversement de la neige et, pour chaque site retenu, à identifier les secteurs qui devraient y être affectés de manière à minimiser le coût total de déneigement. Une affectation de secteurs à un site doit bien sûr se faire en tenant compte des capacités du site.

Campbell et Langevin [3] donnent un bon aperçu de l'ampleur des opérations de déneigement pour la Ville de Montréal. Mentionnons simplement que la ville est divisée en 60 secteurs de déneigement et qu'elle dispose pour le moment de 20 sites de déversement. Chaque année, 660 camions transportent en moyenne 7 millions de mètres cubes de neige, pour un total d'environ 300 000 chargements. On avance le chiffre de $ 60 millions pour le budget total d'opérations de déneigement à Montréal (en 1991). Il est difficile de quantifier les économies pouvant être réalisées par une planification rationnelle du système de déneigement. De même, on peut difficilement mesurer l'impact économique découlant de la mise en oeuvre d'une localisation-affectation optimale pour les sites de déversement. Notons toutefois que Campbell et Langevin [2] ont obtenu, pour Montréal, après la résolution de leur modèle par CPLEX (modèle qui, rappelons-le, est moins général que celui que nous étudions ici, donné dans [3]), une solution optimale d'environ 4.4% inférieure à celle utilisée actuellement par la ville.

Mais la motivation environnementale s'ajoute également aux motivations économiques. Dans plusieurs villes la neige est souvent simplement jetée dans les cours d'eau. Par exemple, on déverse à Montréal 31% de la neige dans le fleuve St-Laurent (cf.[2] et [3]). Il s'agit en effet du mode de déversement le plus "économique", mais

aussi le plus polluant. La volonté de tenir compte de motifs et griefs environnementaux dans la localisation et l'utilisation des sites de déversement diminue la marge de manoeuvre et favorise le recours à des solutions obtenues par les moyens de la recherche opérationnelle.

Ce mémoire est divisé comme suit: le premier chapitre donne deux modèles mathématiques pour PLACC et discute de leur équivalence. Le premièr modèle est celui de Campbell et Langevin (tel qu'énoncé dans [3]), alors que le second est formulé en vue d'une résolution utilisant la génération de colonnes. La décomposition pour la génération de colonnes est présentée au chapitre 2. La méthode de résolution, qui combine les techniques de séparation et évaluation progressive (*branch-and-bound*) et de génération de colonnes, est exposée au chapitre 3. Des résultats sont présentés aux chapitres 4 et 5. La conclusion suit le chapitre 5.

CHAPITRE 1
Modélisation

Ce chapitre a pour but de décrire le problème et d'en fournir deux modèles (ou formulations) mathématiques. Le premier modèle, qui a l'avantage d'être assez explicite, permet de bien comprendre la nature du problème étudié. Le second modèle est formulé en vue d'une résolution utilisant la génération de colonnes. C'est la résolution de ce modèle qui nous intéresse dans ce travail. On termine le chapitre en démontrant l'équivalence des deux modèles présentés.

1.1 Le problème

1.1.1 Données

On dispose de m **secteurs** de déneigement ($i = 1, 2, \ldots, m$), chacun étant caractérisé par un volume de neige à enlever et un niveau de service:

- v_i: volume annuel de neige à enlever dans le secteur i (m^3/an);
- r_i: taux horaire d'enlèvement de la neige dans le secteur i ($m^3/heure$).

Le volume annuel de neige à enlever v_i peut être estimé à partir de données antérieures. Il dépend bien sûr aussi du nombre et de la dimension des rues à déneiger. Le taux horaire d'enlèvement de la neige, quant à lui, découle d'une décision administrative. Il est entendu, par exemple, que certains secteurs doivent

être déneigés plus rapidement que d'autres, moins prioritaires. Il dépend de la quantité et du type d'équipement utilisé dans le secteur.

On dispose d'un ensemble de n **sites** potentiels ($j = 1.2.....n$) pour le déversement de la neige. L'utilisation d'un site pour le déversement entraine un coût fixe annuel et un coût variable selon la quantité de neige qui y est déposée. Un site est également caractérisé par des limites de capacités:

- f_j: coût fixe annuel d'utilisation du site j ($\$/an$);
- b_j: coût variable d'utilisation du site j ($\$/m^3$);
- V_j: capacité annuelle de stockage de neige du site j (m^3/an);
- R_j: capacité horaire de stockage de neige du site j ($m^3/heure$).

Le coût fixe annuel d'utilisation du site est en quelque sorte un coût de localisation généralisé, pouvant contenir, comme son nom l'indique, toutes les formes de coûts fixes reliés à l'utilisation du site. Le coût variable d'utilisation du site, pour sa part, comprend toutes sortes de coûts qui varient selon la quantité de neige qui y est déversée, et particulièrement le coût d'élimination de cette neige. La capacité annuelle d'un site est la quantité maximum de neige pouvant y être déversée annuellement. Il est à noter que certains sites ont une capacité annuelle pratiquement illimitée (par exemple, un cours d'eau). La capacité horaire d'un site est une limite sur la quantité de neige que ce site peut recevoir ou traiter par heure. Elle peut aussi découler des caractéristiques physiques du site, qui limitent l'accès aux camions.

Le **coût total** de déneigement (annuel) comporte alors les coûts fixe et variable d'utilisation des sites retenus pour l'affectation des secteurs, et, bien sûr, le coût de transport de la neige des secteurs à ces sites. On doit donc disposer du coût unitaire de transport de la neige pour chaque paire secteur-site:

- c_{ij}: coût de transport d'un m^3 de neige du secteur i au site j ($\$/m^3$).

Ce coût est généralement considéré comme étant proportionnel à la distance entre le secteur et le site.

On trouvera dans Campbell et Langevin [2], à titre d'exemple, des données relatives au déneigement à Montréal.

1.1.2 Énoncé

On veut déterminer, parmi un ensemble de sites potentiels ($j = 1, 2, \ldots, n$) de déversement de neige, 1) ceux qu'il faut utiliser et 2) comment leur affecter les différents secteurs ($i = 1, 2, \ldots, m$) d'un centre urbain (en respectant les contraintes de capacités des sites) de façon à minimiser le coût annuel total de déneigement pour tous les secteurs. On suppose qu'on ne peut affecter un secteur qu'à un seul site et que le déneigement se fait simultanément sur l'ensemble des secteurs.

Il s'agit d'un problème de localisation-affectation avec contraintes de capacités (PLACC).

1.2 Premier modèle

1.2.1 Modèle

Le premier modèle que nous présentons comporte $n + mn$ variables et $m + 2n + mn$ contraintes. Il correspond en tous points à celui que Campbell et Langevin proposent

dans [3]:

$$\min \sum_{j=1}^{n} f_j y_j + \sum_{i=1}^{m} \sum_{j=1}^{n} (b_j + c_{ij}) v_i x_{ij} \tag{1.1}$$

(M1)

$$\text{s.c.} \sum_{j=1}^{n} x_{ij} = 1 \qquad i = 1, 2, \ldots, m \tag{1.2}$$

$$\sum_{i=1}^{m} v_i x_{ij} \leq V_j \qquad j = 1, 2, \ldots, n \tag{1.3}$$

$$\sum_{i=1}^{m} r_i x_{ij} \leq R_j \qquad j = 1, 2, \ldots, n \tag{1.4}$$

$$x_{ij} \leq y_j \qquad i = 1, 2, \ldots, m;\ j = 1, 2, \ldots, n \tag{1.5}$$

$$x_{ij} \in \{0, 1\} \qquad i = 1, 2, \ldots, m;\ j = 1, 2, \ldots, n \tag{1.6}$$

$$y_j \in \{0, 1\} \qquad j = 1, 2, \ldots, n \tag{1.7}$$

où la variable x_{ij} est égale à 1 si la neige du secteur i est déversée dans le site j (autrement dit si le secteur i est affecté au site j) et à 0 sinon. La variable y_j est égale à 1 si le site j est utilisé pour le déversement de la neige et à 0 sinon. La notation est telle que définie à la section 1.1.1.

La fonction-objectif (1.1) se compose des trois coûts définissant le coût annuel total de déneigement: le coût de localisation des sites (coût fixe d'utilisation), le coût variable d'utilisation des sites et enfin le coût de transport de la neige. Il s'agit de minimiser ce coût annuel total. Les contraintes (1.2) et (1.6) nous assurent que chaque secteur est affecté à un seul site (les contraintes (1.2) sont des contraintes de partitionnement). Les contraintes (1.3) et (1.4) expriment le fait que les affectations doivent être faites en respectant les limites de capacités (annuelle et horaire) de chaque site utilisé. Finalement, les contraintes (1.5) et (1.7) font en sorte qu'il y a affectation d'un secteur à un site seulement si celui-ci est utilisé. La résolution de ce modèle fournit la liste des sites utilisés et les affectations optimales des secteurs à ces sites.

Une variante de (M1), comportant $m+2n$ contraintes, est obtenue en intégrant les contraintes (1.5) aux contraintes de capacités (1.3) et/ou (1.4): il suffit en effet de considérer le produit de la capacité et de y_j dans le membre de droite de ces dernières.

1.2.2 Notation

Dans le but d'alléger la notation, on pose

$$w_{ij} \equiv (b_j + c_{ij})$$

pour le reste de ce mémoire. On peut dès lors considérer w_{ij} comme le coût unitaire (en $\$/m^3$) d'affectation de la neige du secteur i au site j.

1.2.3 Discussion

Ce modèle généralise celui que Campbell et Langevin présentent dans [2], en introduisant l'aspect "localisation". Par ailleurs, le modèle de [2] est la formulation mathématique du problème d'affectation généralisé à deux ressources. Gavish et Pirkul ont étudié le problème d'affectation généralisé à plusieurs ressources dans [9]. Ils y proposent des méthodes de résolution heuristiques et un algorithme de séparation et évaluation progressive (*branch-and-bound*). Cependant, Campbell et Langevin n'ont pas tenté de résoudre de manière exacte leur modèle. Ils ont plutôt proposé une heuristique basée sur des échanges de type *k-opt*. Notons qu'ils ont obtenu une solution optimale pour Montréal après 4.5 heures de calcul sur une station SUN Sparc10 en ayant recours à l'optimiseur CPLEX (cf. [2]).

Le problème d'affectation généralisé à plusieurs ressources (MRGAP) est un problème apparemment peu étudié. Il s'agit d'une généralisation du problème

d'affectation généralisé (GAP) qui a soulevé, quant à lui, plus d'intérêt. Gavish et Pirkul [9] estiment que la difficulté du problème, et, en particulier, le manque de méthodes efficaces pour la résolution de problèmes de "sac-à-dos" (*knapsack*) à plusieurs contraintes (qui découlent de la relaxation lagrangienne du problème), expliqueraient le peu de recherches sur le MRGAP. Ces auteurs ont d'ailleurs proposé des algorithmes pour la résolution de problèmes de "sac-à-dos" à plusieurs contraintes (cf. [8]). Mentionnons pour finir que le MRGAP est NP-dur.

Le modèle présenté ici est donc en un sens une généralisation du problème d'affectation généralisé à deux ressources. Campbell et Langevin font mention dans [3], sans plus de détails, d'une approche heuristique permettant de le résoudre. Notre but n'est pas de résoudre ce modèle, mais un modèle équivalent, formulé en vue d'une résolution par la programmation linéaire généralisée (génération de colonnes).

1.3 Modèle de génération de colonnes

1.3.1 Préambule

Dans ce qui suit, on appellera "colonne" un vecteur décrivant une affectation de secteurs à un site (on suppose dans ce cas que chaque colonne est associée à un site). Une affectation (décrite par une colonne k) est donc représentée comme un vecteur de dimension m $(a_{1k}, a_{2k}, \ldots, a_{ik}, \ldots, a_{mk})$ où a_{ik} est égal à 1 si le secteur i est affecté au site associé à la colonne k et à 0 sinon. (Formellement, il est utile de distinguer le vecteur qui représente une colonne du vecteur qui représente l'affectation. En fait, le vecteur représentant l'affectation forme les m premières composantes du vecteur représentant la colonne). Chaque colonne est caractérisée par un coût c_k qui est le

coût de l'affectation (coût de transport de la neige) auquel s'ajoutent les coûts fixe et variable d'utilisation du site qui lui est associé. Notre problème consiste donc à identifier l'ensemble de colonnes de coût minimum, tout en respectant les contraintes de capacités et de partitionnement. C'est ce que la résolution du modèle que nous proposons ici nous permet de faire.

1.3.2 Modèle

Soient $K = \{1, 2, \ldots, k, \ldots, n(2^m - 1)\}$, l'ensemble des indices associés à toutes les colonnes non vides décrivant des affectations de secteurs à un site, et $K_j \subset K$ l'ensemble des indices de colonnes qui sont associées à un site j. On propose ici un programme mathématique dont la matrice des contraintes se compose des $|K|$ colonnes. Ce modèle de génération de colonnes comporte alors $|K|$ variables (colonnes) et $m + 2n$ contraintes:

$$\min \sum_{k \in K} c_k z_k \tag{1.8}$$

$$\text{s.c.} \sum_{k \in K} a_{ik} z_k = 1 \qquad i = 1, 2, \ldots, m \tag{1.9}$$

(M2)

$$\sum_{k \in K_j} \left(\sum_{i=1}^{m} v_i a_{ik}\right) z_k \leq V_j \qquad j = 1, 2, \ldots, n \tag{1.10}$$

$$\sum_{k \in K_j} \left(\sum_{i=1}^{m} r_i a_{ik}\right) z_k \leq R_j \qquad j = 1, 2, \ldots, n \tag{1.11}$$

$$z_k \in \{0, 1\} \qquad \forall k \in K \tag{1.12}$$

où la variable z_k est égale à 1 si la colonne k est retenue dans la solution et à 0 sinon.

La fonction-objectif (1.8) évalue le coût d'un ensemble de colonnes (ensemble de coût minimum) décrivant des affectations optimales. Pour que ce coût représente

le coût total annuel de déneigement, on doit définir le coût c_k d'une colonne k comme suit:

$$c_k \equiv f_{j(k)} + \sum_{i=1}^{m} w_{ij(k)} v_i a_{ik} \tag{1.13}$$

où $j(k)$ est le site j associé à la colonne. On remarque aussi que pour que le coût calculé soit correct, on ne doit avoir, dans la solution optimale, qu'au plus une colonne choisie par site (et c'est d'ailleurs toujours le cas: voir ci-dessous). Les contraintes (1.9) sont les contraintes de partitionnement. Les contraintes (1.10) et (1.11) sont les contraintes de capacités. La résolution de ce modèle donne un ensemble de colonnes, chacune étant associée à un site, fournissant du même coup les affectations optimales des secteurs aux sites.

La proposition qui suit nous permet d'affirmer que dans la solution optimale (entière) de (M2), on trouve au plus une colonne choisie par site, et ce, sans qu'aucune contrainte spécifique ne soit nécessaire (ce qui ne sera pas le cas dans un programme présenté plus loin). Bien que les contraintes (1.10) et (1.11) n'empêchent pas la possibilité de choisir plus d'une colonne par site, le choix de plus d'une colonne par site est plus coûteux que le choix d'une seule colonne. Et puisque l'on cherche à minimiser le coût, une seule colonne sera toujours choisie.

Proposition 1.3.1 *La solution optimale z^* de (M2) satisfait implicitement la contrainte*

$$\sum_{k \in K_j} z_k \leq 1 \tag{1.14}$$

pour tout site j.

Preuve: Considérons deux colonnes k_1 et k_2 (disjointes), choisies dans une solution de (M2), décrivant une affectation (réalisable) de secteurs à un site j (les deux colonnes étant associées à ce même site). Le coût total de ces colonnes est de $c_{k_1} + c_{k_2} = f_{j(k_1)} + f_{j(k_2)} + \sum_{i=1}^{m} w_{ij(k_1)} v_i a_{ik_1} + \sum_{i=1}^{m} w_{ij(k_2)} v_i a_{ik_2}$, où $f_{j(k_1)} = f_{j(k_2)}$.

Notons z le coût $\sum_{i=1}^{m} w_{ij(k_1)} v_i a_{ik_1} + \sum_{i=1}^{m} w_{ij(k_2)} v_i a_{ik_2}$. On a donc un coût total de $c_{k_1} + c_{k_2} = 2f_{j(k_1)} + z$ pour ces deux colonnes. Il est toujours possible, dans cette solution de (M2), de remplacer ces deux colonnes par une seule colonne k_3 décrivant la même affectation. Le coût c_{k_3} de cette colonne serait alors de $f_{j(k_3)} + z$, avec $f_{j(k_3)} = f_{j(k_1)} = f_{j(k_2)}$, soit $f_{j(k_1)} + z$, qui est plus petit que le coût total des deux colonnes k_1 et k_2 (si $f_{j(k_1)} > 0$). □

1.3.3 Discussion

Ce modèle, bien que comportant un nombre exponentiel de colonnes, peut être résolu dans le cadre de la programmation linéaire généralisée. Plus précisément, la méthode de résolution consiste en un algorithme de séparation et évaluation progressive à l'intérieur duquel on a recours à la technique de génération de colonnes. La technique de génération de colonnes, qui est proche du principe de décomposition de Dantzig-Wolfe, a été utilisée pour la première fois en 1961 par Gilmore et Gomory [10] pour résoudre la relaxation continue du problème de découpe unidimensionnelle (*cutting-stock problem*). On peut la considérér comme une extension de l'algorithme du simplexe permettant de traiter implicitement des problèmes linéaires de grandes taille (nombre de colonnes) pour lesquels il n'est évidemment pas question, ni possible, de travailler sur une matrice de contraintes explicite. L'efficacité de la méthode réside dans le fait que la "colonne entrante" est trouvée par la résolution d'un programme mathématique appelé *problème auxiliaire* (aussi appelé "problème de génération de colonnes" ou "sous-problème"). Le calcul de la colonne se fait à partir des variables duales obtenues de la résolution du *problème maître*.

La technique a été utilisée à partir des années 80 conjointement avec des algorithmes d'évaluation et de séparation pour obtenir des solutions optimales entières

(voir par exemple Desrosiers *et al.* [4], une des toutes premières applications, pour un problème de routage de véhicules). Plus récemment, des algorithmes généraux de génération de colonnes en variables entières ont été proposés. Citons à ce sujet les travaux de Barnhart *et al.* [1], Hansen *et al.* [11] ainsi que ceux de Vanderbeck et Wolsey [18].

La décomposition de notre modèle (M2) pour la génération de colonnes et l'algorithme utilisé pour sa résolution (en nombres entiers) sont décrits aux chapitres 2 et 3.

1.4 Équivalence des modèles

Nous voulons montrer, dans cette section, que le modèle de génération de colonnes (M2) est équivalent au modèle (M1). Pour ce faire, on montre qu'une solution optimale de (M2) permet de déterminer une solution réalisable de (M1) de même valeur et qu'une solution optimale de (M1) permet de déterminer une solution réalisable de (M2) de même valeur. Rappelons d'abord que dans la solution optimale de (M2) on a au plus une colonne choisie ($z_k^* = 1$) par site. C'est-à-dire que pour tout site j, la contrainte (1.14) est satisfaite. Rappelons aussi que $j(k)$ désigne le site j associé à la colonne k.

- Soit z^* la solution optimale de (M2). Soient également les ensembles d'indices de colonnes et de sites $K^* = \{k \in K \mid z_k^* = 1\}$ et $J^* = \{j(k) \mid k \in K^*\}$. Construisons une solution réalisable de (M1):

$\forall j \in J^*$, poser $y_j = 1$;

$\forall j \in J \setminus J^*$, poser $y_j = 0$;

$\forall j(k) \in J^*$, poser $x_{ij(k)} = a_{ik} \quad \forall i$;

$\forall j \in J \setminus J^*$, poser $x_{ij} = 0 \quad \forall i$.

Il est aisé de vérifier que les contraintes (1.5) de (M1) sont satisfaites. Sachant que

$$\sum_{k \in K_j} z_k^* = 1 \qquad \forall j \in J^*$$

et

$$\sum_{k \in K_j} z_k^* = 0 \qquad \forall j \in J \setminus J^*.$$

les contraintes (1.10) donnent

$$\begin{aligned} & \sum_{i=1}^{m} v_i a_{ik} \leq V_{j(k)} \qquad \forall k \in K^* \\ \Rightarrow \quad & \sum_{i=1}^{m} v_i x_{ij(k)} \leq V_{j(k)} \quad \forall j(k) \in J^* \\ \Rightarrow \quad & \sum_{i=1}^{m} v_i x_{ij} \leq V_j \qquad \forall j \in J \end{aligned}$$

et les contraintes (1.3) de (M1) sont donc satisfaites. Il en est de même pour les contraintes (1.4). Les contraintes (1.9) donnent

$$\begin{aligned} & \sum_{k \in K} a_{ik} z_k^* = 1 \qquad \forall i \\ \Rightarrow \quad & \sum_{k \in K^*} a_{ik} = 1 \qquad \forall i \\ \Rightarrow \quad & \sum_{j(k) \in J^*} x_{ij(k)} = 1 \quad \forall i \\ \Rightarrow \quad & \sum_{j \in J} x_{ij} = 1 \qquad \forall i \end{aligned}$$

et les contraintes (1.2) de (M1) sont donc satisfaites. La valeur de la solution de (M1)

$$v_1 = \sum_{j \in J} (f_j y_j + \sum_{i=1}^{m} w_{ij} v_i x_{ij})$$

$$
\begin{aligned}
&= \sum_{j(k)\in J^*} (f_{j(k)} + \sum_{i=1}^{m} w_{ij(k)} v_i x_{ij(k)}) \\
&= \sum_{k\in K^*} (f_{j(k)} + \sum_{i=1}^{m} w_{ij(k)} v_i a_{ik}) \\
&= \sum_{k\in K^*} c_k \\
&= \sum_{k\in K} c_k z_k^* \\
&= v_2^*
\end{aligned}
$$

- Soient maintenant (y^*, x^*), la solution optimale de (M1), et $J^* = \{j \mid y_j^* = 1\}$, l'ensemble des sites utilisés dans la solution. Construisons une solution réalisable de (M2):

$\forall j \in J^*$, identifier la colonne $k \in K_j$ telle que $a_{ik} = x_{ij}^* \ \ \forall i$, poser $z_k = 1$, mettre k dans K^* (initialement vide) et poser $j(k) = j$;

$\forall k \in K \setminus K^*$, poser $z_k = 0$.

Les contraintes (1.3) donnent

$$
\begin{aligned}
& \sum_{i=1}^{m} v_i x_{ij}^* \le V_j && \forall j \in J^* \\
\Rightarrow \ & \sum_{i=1}^{m} v_i a_{ik} \le V_{j(k)} && \forall j(k) \in J^* \\
\Rightarrow \ & \sum_{k\in K_j} (\sum_{i=1}^{m} v_i a_{ik}) z_k \le V_j && \forall j
\end{aligned}
$$

où la dernière implication est vérifiée puisque

$$\sum_{k\in K_j} z_k = 1 \qquad \forall j \in J^*$$

et

$$\sum_{k\in K_j} z_k = 0 \qquad \forall j \in J \setminus J^*.$$

Les contraintes (1.10) de (M2) sont donc satisfaites. Il en est de même pour les contraintes (1.11). Les contraintes (1.2) donnent

$$\begin{aligned} & \sum_{j \in J^*} x_{ij}^* = 1 \qquad \forall i \\ \Rightarrow & \sum_{k \in K^*} a_{ik} = 1 \qquad \forall i \\ \Rightarrow & \sum_{k \in K} a_{ik} z_k = 1 \qquad \forall i \end{aligned}$$

et les contraintes (1.9) de (M2) sont donc satisfaites. La valeur de la solution de (M2)

$$\begin{aligned} v_2 &= \sum_{k \in K} z_k (f_{j(k)} + \sum_{i=1}^{m} w_{ij(k)} v_i a_{ik}) \\ &= \sum_{k \in K^*} f_{j(k)} + \sum_{k \in K^*} \sum_{i=1}^{m} w_{ij(k)} v_i x_{ij(k)}^* \\ &= \sum_{j=1}^{n} f_j y_j^* + \sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij} v_i x_{ij}^* \\ &= v_1^* \end{aligned}$$

Nous venons ainsi de montrer l'équivalence des deux modèles.

CHAPITRE 2
Décomposition

Comme nous l'avons déjà suggéré à la section 1.3.3, la méthode de résolution de (M2) repose en premier lieu sur sa décomposition en un problème maître et un problème auxiliaire, la fonction de ce dernier étant de générer, à chaque itération de l'algorithme de génération de colonnes, la "colonne entrante" pour le problème maître. Par abus de langage, on assimilera donc la résolution de (M2) à la résolution du problème maître, étant cependant bien entendu que cette résolution fait intervenir le problème auxiliaire. Ce chapitre a pour but de préciser la décomposition utilisée pour (M2) et de comparer les valeurs des bornes que fournissent les relaxations continues de (M1) et du problème maître.

2.1 Décomposition de (M2) pour la génération de colonnes

La décomposition consiste essentiellement à transférer les contraintes de capacités (1.10) et (1.11) de (M2) au problème auxiliaire. Ces contraintes s'appliquant à chaque site, on considérera un problème auxiliaire pour chaque site. Dans ces conditions, les problèmes auxiliaires deviennent des problèmes de "sac-à-dos" à deux contraintes (*bidimensional knapsack*). Le problème auxiliaire associé au site j calcule, à chaque itération de l'algorithme de génération de colonnes, une colonne entrante réalisable pour ce site (si elle existe).

Le problème maître, quant à lui, se compose alors de (M2), sans les contraintes

de capacités, auquel on doit cependant ajouter les contraintes (1.14) de la proposition 1.3.1, devenues nécessaires en raison du transfert des contraintes de capacités aux problèmes auxiliaires. Rappelons qu'on ne doit avoir, dans la solution optimale, qu'au plus une colonne choisie par site. Ceci était toujours le cas dans (M2) d'après la proposition 1.3.1. Cette proposition ne s'applique cependant pas au problème maître, car il n'y est pas toujours possible de remplacer, dans une solution, deux colonnes disjointes décrivant une affectation (à un même site) par une seule colonne décrivant la même affectation. En effet, cette colonne pourrait ne pas être réalisable (c'est-à-dire qu'elle pourrait violer les contraintes de capacités), donc ne jamais être générée par le problème auxiliaire.

2.1.1 Le problème maître

Suite au transfert des contraintes de capacités aux problèmes auxiliaires, on obtient le problème maître suivant, qui comporte $|K|$ variables (colonnes) implicites et $m+n$ contraintes:

$$\min \sum_{k \in K} c_k z_k \tag{2.1}$$

(PM)

$$\text{s.c.} \sum_{k \in K} a_{ik} z_k = 1 \qquad i = 1, 2, \ldots, m \tag{2.2}$$

$$\sum_{k \in K_j} z_k \leq 1 \qquad j = 1, 2, \ldots, n \tag{2.3}$$

$$z_k \in \{0, 1\} \qquad \forall k \in K \tag{2.4}$$

où les c_k, a_{ik} et z_k ont la même signification que dans le modèle (M2). Le coût c_k d'une colonne k est défini à l'équation (1.13).

La fonction-objectif (2.1) et les contraintes (2.2) restent inchangées (par rapport à (M2)), alors que les contraintes (1.14) de la proposition 1.3.1 sont ici explicites:

il s'agit des contraintes (2.3).

Nous voulons insister ici sur le fait que, contrairement au modèle (M2), les $|K|$ colonnes de (PM) sont ici *implicites*, c'est-à-dire qu'elles sont générées au besoin par les problèmes auxiliaires décrits ci-dessous.

2.1.2 Les problèmes auxiliaires

Soient

$$u = ([u_i]_{i=1}^{m}, [u_{m+j}]_{j=1}^{n}),$$

le vecteur représentant les variables duales associées à la solution courante du (PM), où les $u_i \in \mathbb{R}$ et les $u_{m+j} \in \mathbb{R}_-$, et

$$\alpha_k^j = ([a_{ik}]_{i=1}^{m}, [\delta_{jr}]_{r=1}^{n})^T,$$

le vecteur représentant une colonne k de (PM) associée au site j, où le delta de Kroneker δ_{jr} est égal à 1 si $r = j$ et à 0 sinon. Le coût réduit de cette colonne s'écrit

$$c_k - u\alpha_k^j$$

ou bien, en utilisant l'équation (1.13),

$$f_{j(k)} + \sum_{i=1}^{m} w_{ij(k)} v_i a_{ik} - u\alpha_k^j$$

et on obtient, en développant le produit scalaire $u\alpha_k^j$, un coût réduit de

$$f_{j(k)} + \sum_{i=1}^{m} w_{ij(k)} v_i a_{ik} - \sum_{i=1}^{m} u_i a_{ik} - u_{m+j}$$

c'est-à-dire

$$f_{j(k)} + \sum_{i=1}^{m} (v_i w_{ij(k)} - u_i) a_{ik} - u_{m+j}. \tag{2.5}$$

Toute colonne (associée au site j) pour laquelle le coût réduit, tel que défini par l'expression (2.5), est négatif peut être considérée comme une candidate à l'entrée

dans la base du problème maître (PM). En particulier, la colonne de coût réduit le plus négatif est un bon choix. Le problème auxiliaire (associé au site j) revient dans ce cas à calculer la colonne réalisable (qui respecte les contraintes de capacités du site j) de coût réduit minimum. Si cette colonne est de coût réduit négatif, alors on pourra la considérer pour l'entrée en base de (PM). Le problème auxiliaire associé au site j se formule donc comme suit:

$$\min \quad f_j - u_{m+j} + \sum_{i=1}^{m} (w_{ij}v_i - u_i)a_i \tag{2.6}$$

$$\textbf{(PA)}_j \qquad \text{s.c.} \quad \sum_{i=1}^{m} v_i a_i \leq V_j \tag{2.7}$$

$$\sum_{i=1}^{m} r_i a_i \leq R_j \tag{2.8}$$

$$a_i \in \{0,1\} \qquad i = 1,2,\ldots,m \tag{2.9}$$

où la variable a_i est égale à 1 si la neige du secteur i est déversée dans le site j et à 0 sinon.

Il s'agit d'un problème de "sac-à-dos" à deux contraintes, dont la fonction-objectif (2.6) représente le coût réduit de la colonne entrante associée au site j. Les affectations des secteurs à ce site sont définies par les variables a_i, qui forment les m premières composantes de la colonne, lorsque celle-ci est insérée dans la base de (PM).

2.2 Quelques remarques sur la méthode de résolution

Le moment est venu de donner quelques éclaircissements au sujet de la méthode de résolution. Rappelons que notre but est de résoudre le modèle (M2) présenté à la section 1.3.2. Pour ce faire, nous l'avons décomposé en un problème maître (PM) et des problèmes auxiliaires $(\text{PA})_j$. La résolution de (PM) donne donc la

solution recherchée, qui est celle de (M2). L'algorithme utilisé pour le résoudre, faisant intervenir la génération de colonnes, est présenté au prochain chapitre.

Il serait intéressant de s'interroger sur les valeurs des relaxations continues de (M1) et de (PM), qui fournissent des bornes inférieures sur la valeur de la solution entière. Nous en profitons d'ailleurs pour faire remarquer que les relaxations continues de (PM) et de (M2) ne sont pas équivalentes puisque, rappelons-le, nous avons dû ajouter la contrainte (2.3) à (PM), contrainte qui n'apparait pas dans (M2), bien qu'elle soit satisfaite (implicitement) à l'optimalité de (M2) (cf. proposition 1.3.1). On peut donc conclure, *a priori*, que la valeur de la relaxation continue de (M2) est inférieure ou égale à la valeur de la relaxation continue de (PM). Quoi qu'il en soit, les bornes qui nous intéressent ici sont celles que nous donnent les relaxations continues de (M1) et de (PM), et c'est ce dont il est question à la section suivante.

2.3 Comparaison théorique des valeurs des relaxations continues de (M1) et (PM)

Nous voulons comparer ici, de manière théorique, les valeurs des bornes inférieures que fournissent les relaxations continues de (M1) et (PM). On notera PL1 et PLM les relaxations continues des modèles (M1) et (PM). La démarche consiste à montrer que, disposant d'une solution réalisable de PLM, il est possible de construire une solution réalisable de PL1 de même valeur. La conséquence de ceci est que la résolution de PL1 ne peut pas fournir de meilleure borne inférieure que la résolution de PLM.

Proposition 2.3.1 *L'application des formules*

$$x_{ij} = \sum_{k \in K_j} a_{ik} z_k \qquad \forall i, j \tag{2.10}$$

$$y_j = \sum_{k \in K_j} z_k \qquad \forall j \tag{2.11}$$

permet de construire, à partir d'une solution réalisable de PLM, une solution réalisable de PL1 de même valeur.

Preuve: Montrons d'abord que la solution obtenue est réalisable pour PL1. Les contraintes (1.2) donnent

$$\sum_{j=1}^{n} \sum_{k \in K_j} a_{ik} z_k = \sum_{k \in K} a_{ik} z_k = 1 \qquad \forall i$$

puisque les contraintes (2.2) de (PM) sont satisfaites. Les contraintes (1.3) donnent

$$\sum_{i=1}^{m} v_i \sum_{k \in K_j} a_{ik} z_k = \sum_{k \in K_j} z_k \sum_{i=1}^{m} v_i a_{ik} \leq \sum_{k \in K_j} z_k V_j \leq V_j \quad \forall j,$$

la première inégalité découlant du fait que toutes les colonnes satisfont les contraintes de capacité (2.7), la seconde découlant de la satisfaction de la contrainte (2.3) de (PM). Il en est de même pour les contraintes (1.4). Les contraintes (1.5) sont visiblement satisfaites puisque $a_{ik} z_k \leq z_k \ \forall i, k$. Finalement, puisque les contraintes (2.2) de (PM) sont satisfaites, on a forcément

$$x_{ij} = \sum_{k \in K_j} a_{ik} z_k \leq 1 \quad \forall i, j$$

et, puisque les contraintes (2.3) de (PM) sont satisfaites, on a $y_j \leq 1 \ \ \forall j$.

Montrons maintenant que la solution obtenue est de même valeur que la solution de PLM. Soient v_{PL1} et v_{PLM} les valeurs des solutions de PL1 et PLM.

$$v_{PL1} = \sum_{j=1}^{n} f_j y_j + \sum_{j=1}^{n} \sum_{i=1}^{m} w_{ij} v_i x_{ij}$$

$$
\begin{aligned}
&= \sum_{j=1}^{n} f_j \Big(\sum_{k \in K_j} z_k \Big) + \sum_{j=1}^{n} \sum_{i=1}^{m} w_{ij} v_i \Big(\sum_{k \in K_j} a_{ik} z_k \Big) \\
&= \sum_{j=1}^{n} \sum_{k \in K_j} f_j z_k + \sum_{j=1}^{n} \sum_{k \in K_j} \sum_{i=1}^{m} w_{ij} v_i a_{ik} z_k \\
&= \sum_{k \in K} \Big(f_{j(k)} + \sum_{i=1}^{m} w_{ij(k)} v_i a_{ik} \Big) z_k \\
&= \sum_{k \in K} c_k z_k \\
&= v_{PLM} \quad \square
\end{aligned}
$$

La proposition suivante est une conséquence immédiate de la proposition 2.3.1:

Proposition 2.3.2 *Soient v^*_{PL1} et v^*_{PLM} les solution optimales de PL1 et PLM. Soit aussi v_{PL1}, la solution réalisable de PL1 construite avec les formules (2.10) et (2.11) à partir de la solution optimale de PLM. On a alors*

$$v^*_{PLM} = v_{PL1} \geq v^*_{PL1}.$$

En d'autres mots, la résolution de PL1 ne donne pas de meilleure borne que la résolution de PLM.

Notons que ce résultat est partiel, puisqu'il n'exclut pas la possibilité que l'égalité $v^*_{PLM} = v^*_{PL1}$ soit toujours satisfaite. Les résultats expérimentaux que nous présentons au chapitre 4 montrent cependant que ce n'est pas le cas.

CHAPITRE 3
Résolution

Ce chapitre a pour but de présenter la démarche adoptée pour la résolution du problème maître (PM). La solution continue de (PM) est obtenue par génération de colonnes. Un algorithme de séparation et évaluation progressive permet d'en obtenir la solution entière, qui est aussi la solution du modèle (M2).

3.1 Résolution de la relaxation continue de (PM)

La solution continue de (PM) est obtenue en appliquant l'algorithme de génération de colonnes sur le modèle décomposé présenté à la section précédente, où la contrainte (2.4) est remplacée par $0 \leq z_k \leq 1 \quad \forall k$. En d'autres mots, on considère ici la relaxation continue du problème maître, notée PLM.

3.1.1 Algorithme de génération de colonnes

Dans ce qui suit, on appellera PLM *restreint* le problème PLM où seul un nombre restreint de colonnes (explicites) est considéré. Voici l'algorithme, suivi de quelques commentaires:

Initialisation: Construire un PLM restreint en fournissant un ensemble de colonnes réalisables.

Étape 1: Résoudre le PLM restreint courant.

Étape 2: Obtenir les variables duales associées à la solution courante du PLM restreint.

Étape 3: Pour $j = 1$ à n, construire et résoudre $(PA)_j$. Chaque $(PA)_j$ fournit la colonne de coût réduit minimum pour le site j.
Si aucune colonne de coût réduit négatif n'a été obtenue, **FIN**: la solution du PLM restreint courant est optimale; il s'agit de la solution continue de (PM).
Sinon, aller à l'Étape 4.

Étape 4: Construire et introduire les colonnes de coût réduit négatif (parmi les n possibles) dans la base du PLM restreint courant et aller à l'Étape 1.

Il n'est pas difficile, à l'étape d'initialisation, de fournir un ensemble de colonnes permettant de démarrer l'algorithme. Il faut seulement s'assurer que les colonnes soient réalisables (c'est-à-dire qu'elles doivent satisfaire les contraintes de capacités). Il s'agit en fait de définir la matrice de contraintes du PLM restreint, à partir de laquelle une solution initiale sera obtenue. On peut d'ailleurs insérer directement des colonnes représentant une solution réalisable à PLM. Ceci peut être fait à l'aide d'une heuristique, plus ou moins sophistiquée selon l'objectif recherché. Nous avons choisi une heuristique sommaire pour obtenir une solution initiale réalisable.

Comme le PLM restreint comporte un nombre raisonnable de colonnes, il est possible de le résoudre assez aisément, par exemple en ayant recours à un optimiseur commercial (CPLEX, par exemple), dont la base est l'algorithme du simplexe (c'est la démarche que nous avons employée). Il est peut-être utile de préciser ce que nous entendons par nombre "raisonnable" de colonnes: dans notre cas, on peut l'estimer à l'ordre de la centaine de colonnes. En fait, puisque les colonnes s'ajoutent d'itération en itération dans la matrice du problème, il peut devenir nécessaire d'en contrôler le nombre (c'est-à-dire d'en éliminer une partie) pendant la génération de colonnes, de

manière à limiter le temps de calcul (et l'espace mémoire). Une stratégie possible est d'éliminer, lorsque le nombre de colonnes devient trop grand, une certaine quantité de colonnes hors-base parmi celles dont le coût réduit est le plus positif.

La résolution des problèmes auxiliaires est commentée dans la section qui suit. Rappelons qu'il n'est pas nécessaire, pour assurer la convergence de l'algorithme, que les problèmes auxiliaires soient résolus à l'optimalité (sauf à la dernière itération). En effet, la seule exigence est que les colonnes produites soient de coût réduit négatif. Nous avons cependant choisi de résoudre les problèmes auxiliaires à l'optimalité puisque l'algorithme utilisé pour les résoudre est assez efficace, du moins pour la taille des problèmes qui nous intéressent. Nous reviendrons sur ce sujet au chapitre 4. De même, il n'est pas nécessaire d'introduire toutes les colonnes de coût réduit le plus négatif, comme il est écrit ci-dessus, à l'Étape 4. Cependant, des résultats préliminaires ont montré qu'il était souhaitable de le faire, comme nous le verrons aussi au chapitre 4.

3.1.2 Résolution des problèmes auxiliaires

La première étape dans la résolution d'un problème auxiliaire est de le mettre sous la forme d'un problème de maximisation (le problème de "sac-à-dos" est généralement défini et étudié sous cette forme). On peut aussi, après cette transformation, réduire la taille du problème en posant à 0 les variables a_i dont les coefficients $(u_i - w_{ij}v_i)$ sont négatifs. On retrouve alors un problème de la forme (BiSAD), tel que présenté plus loin. Il s'agit d'un problème de "sac-à-dos" à deux contraintes (ou bidimensionnel).

Le problème de "sac-à-dos" unidimensionnel a été passablement étudié et est

désormais considéré comme un problème classique de la programmation mathématique en variables 0-1. Bien que ce problème soit NP-dur, des méthodes exactes et efficaces en pratique, exploitant sa structure, existent pour le résoudre. On trouve dans Martello et Toth [14] un relevé des différents algorithmes permettant de résoudre le problème de "sac-à-dos" et quelques uns de ses dérivés.

Le cas multidimensionnel (et bidimensionnel, en particulier) a reçu moins d'attention. Nous avons déjà mentionné, à la section 1.2.3, les travaux de Gavish et Pirkul [8] sur le problème multidimensionnel. Ces auteurs proposent un algorithme de séparation et évaluation progressive pour la résolution de ce problème et comparent la qualité des bornes obtenues à partir des différentes relaxations utilisées (continue, lagrangienne, et relaxations avec contraintes agrégées (*surrogate relaxation*) et composite). Fréville et Plateau [7] s'intéressent à la résolution exacte de la relaxation avec contraintes agrégées pour le problème bidimensionnel, la méthode utilisée pouvant éventuellement être intégrée à un algorithme d'énumération implicite pour le résoudre, déjà proposé par ces mêmes auteurs (cf. [6]). Il existe somme toute apparemment peu de travaux sur ce problème, dont les applications sont nombreuses.

La résolution du problème de "sac-à-dos" à deux contraintes aurait donc pu faire, à elle seule, l'objet d'une étude complète. Notre ambition est plus modeste. Nous avons recours à un algorithme d'énumération implicite utilisant une relaxation avec contrainte agrégée, sans toutefois chercher à obtenir la meilleure borne possible pour celle-ci. Cette relaxation consiste en un problème de "sac-à-dos" classique qui est à son tour résolu par un algorithme d'énumération implicite. Nous commençons par décrire ci-dessous les caractéristiques principales de l'algorithme d'énumération implicite utilisé pour le problème à une seule contrainte, et nous complétons la section par la description de l'algorithme utilisé pour la résolution du problème à deux

contraintes.

Résolution du problème de "sac-à-dos" unidimensionnel

On considère ici le problème

$$\textbf{(SAD)} \qquad \begin{array}{lll} \max & \sum_{i=1}^{m} c_i x_i & \\ \text{s.c.} & \sum_{i=1}^{m} a_i x_i \leq b & \\ & x_i \in \{0,1\} & i = 1,2,\ldots,m \end{array}$$

pour lequel on suppose, sans perte de généralité, que $c_i > 0$ et $a_i > 0$ pour tout i et que $c_1/a_1 \geq c_2/a_2 \geq \ldots \geq c_m/a_m$.

La méthode utilisée pour résoudre ce problème est, tant au niveau de l'algorithme que de son implantation (usage d'un max-minimier), celle décrite par Jaumard et Gourdin dans [12]. Elle repose sur un algorithme d'énumération implicite qu'on résumera en précisant le calcul des bornes, la règle de division (décomposition) d'un sous-problème et la règle de branchement (sélection du sous-problème).

Une **borne supérieure** pour (SAD) est obtenue en résolvant sa relaxation continue. La proposition suivante, tirée de [12] et bien connue, décrit la solution optimale de cette relaxation:

Proposition 3.1.1 *Soit $r \in \{1,2,\ldots,m\}$ satisfaisant $\sum_{i=1}^{r-1} a_i \leq b < \sum_{i=1}^{r} a_i$. L'n vecteur optimal de la relaxation continue est défini comme suit:*

$$\begin{array}{rcll} x_i & = & 1 & i = 1,2,\ldots,r-1 \\ x_r & = & \dfrac{b - \sum_{i=1}^{r-1} a_i}{a_r} & \\ x_i & = & 0 & i = r+1, r+2, \ldots, m \end{array} \qquad (3.1)$$

où il est supposé, par convention, que $\sum_{i=1}^{0} a_i = 0$.

Une **borne inférieure** pour (SAD) est obtenue en balayant toutes les variables (de x_1 à x_m) pour poser le plus grand nombre possible de ces variables à 1 sans dépasser la contrainte de capacité. Pour un sous-problème donné, on devra bien sûr tenir compte, dans les calculs précédents, des contraintes additionnelles qui résultent de la règle de division.

La **division** utilisée est de type dichotomique sur une variable: il s'agit de la variable fractionnaire d'indice r résultant du cacul de la relaxation continue du problème courant (équation 3.1). La règle de division consiste donc à décomposer le problème en deux sous-problèmes: un pour lequel la variable $x_r = 1$ et un autre pour lequel la variable $x_r = 0$. Les contraintes de division s'appliquant à un sous-problème donné se ramènent donc à une liste de variables fixées.

La règle de **branchement** consiste à choisir, parmi la liste des sous-problèmes non-résolus, celui dont la borne supérieure est la plus élevée (meilleur d'abord).

Résolution du problème de "sac-à-dos" bidimensionnel

On considère ici le problème

$$\textbf{(BiSAD)} \qquad \begin{aligned} \max \quad & \sum_{i=1}^{m} c_i x_i \\ \text{s.c.} \quad & \sum_{i=1}^{m} a_i^1 x_i \leq b^1 \\ & \sum_{i=1}^{m} a_i^2 x_i \leq b^2 \\ & x_i \in \{0,1\} \qquad i = 1, 2, \ldots, m \end{aligned}$$

pour lequel on suppose, sans perte de généralité, que $c_i > 0$, $a_i^1 > 0$ et $a_i^2 > 0$ pour tout i et que $c_1/max\{a_1^1, a_1^2\} \geq c_2/max\{a_2^1, a_2^2\} \geq \ldots \geq c_m/max\{a_m^1, a_m^2\}$.

La méthode utilisée pour résoudre ce problème est sensiblement la même que celle décrite ci-dessus pour le cas unidimensionnel. L'implantation est identique et l'algorithme est peu modifié, sauf en ce qui concerne le calcul de la borne supérieure.

Dans un problème de la forme (BiSAD), il est possible qu'une des contraintes soit inactive à l'optimalité. Dans ce cas, cette contrainte peut être ignorée sans que la solution optimale en soit affectée. Considérons les problèmes de "sac-à-dos" unidimensionnels (SAD1) et (SAD2) obtenus de (BiSAD) en ignorant respectivement la deuxième contrainte de capacité et la première contrainte de capacité:

$$\textbf{(SAD1)} \qquad \begin{aligned} \max \quad & \sum_{i=1}^{m} c_i x_i \\ \text{s.c.} \quad & \sum_{i=1}^{m} a_i^1 x_i \leq b^1 \\ & x_i \in \{0,1\} \qquad i = 1,2,\ldots,m \end{aligned}$$

$$\textbf{(SAD2)} \qquad \begin{aligned} \max \quad & \sum_{i=1}^{m} c_i x_i \\ \text{s.c.} \quad & \sum_{i=1}^{m} a_i^2 x_i \leq b^2 \\ & x_i \in \{0,1\} \qquad i = 1,2,\ldots,m \end{aligned}$$

On propose la démarche suivante:

1. Résoudre (SAD1). Soit x^{1*} la solution optimale. Si

$$\sum_{i=1}^{m} a_i^2 x_i^{1*} \leq b^2$$

alors x^{1*} est optimal pour (BiSAD). Sinon

2. Résoudre (SAD2). Soit x^{2*} la solution optimale. Si

$$\sum_{i=1}^{m} a_i^1 x_i^{2*} \leq b^1$$

alors x^{2*} est optimal pour (BiSAD). Sinon

3. Considérer un problème de "sac-à-dos" avec contrainte agrégée pour obtenir une borne supérieure sur (BiSAD).

Une **borne supérieure** pour (BiSAD) est obtenue en résolvant le problème avec contrainte agrégée (résultant de la combinaison convexe des deux contraintes de capacités) suivant, qui est une relaxation de (BiSAD):

$$\textbf{(SAD}(\alpha)\textbf{)} \qquad \begin{aligned} \max \quad & \sum_{i=1}^{m} c_i x_i \\ \text{s.c.} \quad & \sum_{i=1}^{m} \alpha a_i^1 x_i + (1-\alpha) a_i^2 x_i \leq \alpha b^1 + (1-\alpha) b^2 \\ & x_i \in \{0,1\} \qquad i = 1, 2, \ldots, m \end{aligned}$$

où le facteur α $(0 < \alpha < 1)$ est défini comme suit:

$$\alpha \equiv \frac{\delta_1}{\delta_1 + \delta_2}$$

avec

$$\delta_1 \equiv \frac{\sum_{i=1}^{m} a_i^1 x_i^{2*} - b^1}{b^1} > 0$$

et

$$\delta_2 \equiv \frac{\sum_{i=1}^{m} a_i^2 x_i^{1*} - b^2}{b^2} > 0.$$

Si $\delta_1 > \delta_2$, alors α est plus près de 1 que de 0 et la contrainte 1 est favorisée dans la combinaison. Inversement, si $\delta_2 > \delta_1$, alors α est plus près de 0 que de 1 et la contrainte 0 est favorisée dans la combinaison. On peut considérer δ_i comme le degré de violation de la containte i par la solution optimale du problème de "sac-à-dos" dans lequel cette contrainte est ignorée. La combinaison proposée favorise donc la contrainte dont le degré de violation est le plus grand.

Le choix de α, donc des multiplicateurs α et $(1-\alpha)$, ne garantit cependant pas que la borne obtenue soit la meilleure (minimale). Mais cette façon de calculer

la contrainte agrégée donne d'assez bons résultats en pratique. Nous avons voulu éviter, faute de temps, l'implantation d'une méthode plus complexe qui permettrait d'obtenir la borne minimale. Les travaux de Gavish et Pirkul [8], ainsi que ceux de Fréville et Plateau [7], déjà mentionnés, proposent des méthodes de recherche dichotomiques qui permettent d'obtenir les multiplicateurs optimaux pour une contrainte agrégée résultant de la combinaison convexe de deux contraintes de capacités.

Une **borne inférieure** pour (BiSAD) est obtenue de la manière suivante:

1. Appliquer la technique décrite plus haut pour le cas unidimensionnel à chaque contrainte de (BiSAD). Soient x^1 et x^2 les vecteurs solutions obtenus.
2. Le vecteur retenu pour la borne inférieure est obtenu en faisant x^1 ET x^2.

Bien sûr, on tiendra compte, dans ces calculs, des contraintes additionnelles résultant de la règle de division énoncée ci-dessous. Il s'agit, comme dans le cas unidimensionnel, de prendre en compte une liste de variables fixées.

La **division** est ici aussi de type dichotomique sur une variable. On considère la variable fractionnaire x_r obtenue de la résolution de la relaxation du problème (SAD(α)) courant. La règle de division consiste donc à décomposer le problème en deux sous-problèmes: un pour lequel la variable $x_r = 1$ et un autre pour lequel la variable $x_r = 0$. Les contraintes qui résultent de la division se résument donc encore ici à une liste de variables fixées.

La règle de **branchement** consiste à choisir, parmi la liste des sous-problèmes non-résolus, celui dont la borne supérieure est la plus élevée ("meilleur d'abord").

3.2 Résolution en nombres entiers de (PM)

Nous allons voir dans cette section comment la technique de génération de colonnes est imbriquée dans un algorithme de séparation et évaluation progressive pour résoudre (PM) en nombres entiers. L'algorithme est résumé par la description de la règle de division et du calcul de la borne inférieure d'un sous-problème.

3.2.1 Schéma de la méthode

Nous proposons un algorithme de séparation et évaluation progressive (B&B) pour résoudre (PM). À chaque noeud de l'arbre de B&B, le sous-problème est le problème maître (PM) initial avec des contraintes additionnelles résultant de l'application successive de la règle de division. Dans notre cas, la règle de division pour un sous-problème implique, sur chaque branche, l'addition d'une contrainte qui peut être représentée implicitement en éliminant certaines colonnes dans la matrice du sous-problème et en ajoutant des contraintes aux problèmes auxiliaires de manière à empêcher la génération de colonnes qui sont interdites. La structure des problèmes auxiliaires s'en trouve donc affectée et leur résolution doit être modifiée en conséquence. L'algorithme de génération de colonnes est utilisé à chaque noeud pour résoudre la relaxation continue du sous-problème. La règle de branchement consiste à explorer l'arbre de B&B selon le critère "profondeur d'abord".

Tout ce qui a été vu jusqu'à présent dans ce chapitre s'applique donc au premier noeud (noeud 0) de l'arbre de B&B, puisque la relaxation continue de (PM) est en fait la relaxation continue du premier sous-problème.

3.2.2 Règle de division d'un sous-problème

Si la solution de la relaxation continue du sous-problème est non-entière (fractionnaire), alors on doit le décomposer. La règle de division utilisée est celle suggérée par Ryan et Foster [17] pour la résolution d'un problème de partitionnement. Cette règle de division a été reprise par Barnhart *et al.* [1] et Vanderbeck et Wolsey [18] dans leurs propositions d'un algorithme général de génération de colonnes pour la programmation en nombres entiers, appliquée au cas particulier du problème de partitionnement. Elle a d'ailleurs été utilisée avec succès dans de nombreuses applications.

La proposition qui suit, tirée des travaux que nous venons de citer, résume bien la règle de division d'un sous-problème:

Proposition 3.2.1 *Si la solution z de la relaxation du sous-problème est fractionnaire, alors il existe deux secteurs i_1 et i_2 tels que*

$$0 < \sum_{k\,:\,a_{i_1 k} = a_{i_2 k} = 1} z_k < 1 \tag{3.2}$$

et on peut définir les contraintes de division suivantes:

$$\sum_{k\,:\,a_{i_1 k} = a_{i_2 k} = 1} z_k = 1 \tag{3.3}$$

$$\sum_{k\,:\,a_{i_1 k} = a_{i_2 k} = 1} z_k = 0. \tag{3.4}$$

La contrainte (3.3) doit s'appliquer sur une branche et la contrainte (3.4) doit s'appliquer sur l'autre, chacune déterminant un nouveau sous-problème.

On peut tenir compte implicitement de la contrainte (3.3) en éliminant toutes les colonnes k de la matrice du sous-problème (problème maître) pour lesquelles

$a_{i_1 k} + a_{i_2 k} = 1$ et en ajoutant la contrainte

$$a_{i_1} = a_{i_2} \tag{3.5}$$

dans les problèmes auxiliaires.

On peut tenir compte implicitement de la contrainte (3.4) en éliminant toutes les colonnes k de la matrice du sous-problème pour lesquelles $a_{i_1 k} = a_{i_2 k} = 1$ et en ajoutant la contrainte

$$a_{i_1} + a_{i_2} \leq 1 \tag{3.6}$$

dans les problèmes auxiliaires.

En d'autres mots, la contrainte (3.3) oblige les deux secteurs i_1 et i_2 à être affectés ensemble, alors que l'autre (3.4) oblige ces secteurs à être affectés séparément.

La proposition suivante découle de la proposition 3.2.1:

Proposition 3.2.2 *Si la solution z de la relaxation du sous-problème est fractionnaire, alors il existe deux secteurs i_1 et i_2 tels qu'il est toujours possible d'identifier une colonne k_1 pour laquelle $0 < z_{k_1} < 1$ avec*

$$a_{i_1 k_1} = a_{i_2 k_1} = 1 \tag{3.7}$$

et une colonne k_2 pour laquelle $0 < z_{k_2} < 1$ avec

$$a_{i_1 k_2} = 1 \quad et \quad a_{i_2 k_2} = 0 \tag{3.8}$$

ou

$$a_{i_1 k_2} = 0 \quad et \quad a_{i_2 k_2} = 1. \tag{3.9}$$

Preuve: D'après l'expression (3.2) de la proposition 3.2.1, il est évident que la colonne k_1 existe. De plus, il est nécessaire que la colonne k_2 existe: en effet, si

elle n'existait pas, il faudrait alors, pour que la contrainte de partition (2.2) soit satisfaite, que

$$\sum_{k\,:\,a_{i_1 k} = a_{i_2 k} = 1} z_k = 1,$$

contredisant ainsi l'expression (3.2). □

Cette proposition peut être utilisée pour obtenir les secteurs i_1 et i_2. On pourra, par exemple, trouver une colonne fractionnaire k_1 et vérifier qu'à deux secteurs i_1 et i_2 satisfaisant l'équation (3.7), on trouve bien une colonne k_2 fractionnaire pour laquelle l'équation (3.8) ou (3.9) est satisfaite.

La règle de division épargne la structure du problème maître. Dans ces conditions, chaque sous-problème a la même forme que (PM) et ne se distingue d'un autre qu'au niveau des colonnes interdites. Il s'agit à la fois de modifier les colonnes présentes dans la matrice des contraintes du sous-problème (les colonnes explicites) et les colonnes à être générées par les problèmes auxiliaires, encore implicites. Ainsi, la transition d'un sous-problème à un autre se fait dans une relative continuité au fil de l'arbre de B&B, à tout le moins en ce qui concerne le problème maître.

3.2.3 Borne inférieure (relaxation) d'un sous-problème

La borne inférieure sur la valeur de la solution du sous-problème est obtenue en résolvant la relaxation continue du sous-problème. Comme chaque sous-problème a la même forme que (PM), l'algorithme de génération de colonnes présenté à la section 3.1.1 peut être utilisé tel quel (sans l'étape d'initialisation) pour résoudre la relaxation continue du sous-problème, à condition que la résolution des problèmes auxiliaires, à l'Étape 3, se fasse en tenant compte des contraintes additionnelles (3.5) et (3.6) découlant de la règle de division. La résolution des problèmes auxiliaires modifiés par l'ajout des contraintes est décrite à la section suivante.

3.2.4 Résolution des problèmes auxiliaires modifiés

Nous avons vu que la structure des problèmes auxiliaires à un noeud donné de l'arbre de B&B est affectée par l'application successive, jusqu'à ce noeud, de la règle de division. Soient respectivement E_p et I_p les ensembles de paires de secteurs $\{i_1, i_2\}$ considérés dans toutes les contraintes (3.5) et (3.6) accumulées jusqu'au noeud p de l'arbre de B&B. Le problème auxiliaire (pour le site j) au noeud p de l'arbre de B&B s'écrit comme suit:

$$\min \quad f_j - u_{m+j} + \sum_{i=1}^{m} (w_{ij}v_i - u_i)a_i \tag{3.10}$$

$$\text{s.c.} \quad \sum_{i=1}^{m} v_i a_i \leq V_j \tag{3.11}$$

$$\sum_{i=1}^{m} r_i a_i \leq R_j \tag{3.12}$$

$$a_{i_1} = a_{i_2} \qquad \forall \{i_1, i_2\} \in E_p \tag{3.13}$$

$$a_{i_1} + a_{i_2} \leq 1 \qquad \forall \{i_1, i_2\} \in I_p \tag{3.14}$$

$$a_i \in \{0, 1\} \qquad i = 1, 2, \ldots, m \tag{3.15}$$

$(\mathbf{PA\_}p)_j$

Les contraintes (3.13) peuvent être éliminées en les intégrant implicitement dans (3.10), (3.11) et (3.12): il s'agit en fait de regrouper les variables égales. On peut ensuite poser à zéro les variables qui ont un coefficient positif dans la fonction-objectif résultante et éliminer les contraintes (3.14) pour lesquelles ce changement rend la contrainte toujours satisfaite. Après avoir opéré un changement de variables (variables réindicées), on obtient le problème auxiliaire (pour le site j) au noeud p suivant, où les contraintes d'inégalité (3.16) s'ajoutent à une structure de problème de "sac-à-dos" à deux contraintes:

$$\min \quad f_j - u_{m'+j} + \sum_{i=1}^{m'} (w_{ij}v_i - u_i)a_i$$

$$(\mathbf{PA\_}p)'_j \qquad \text{s.c.} \quad \sum_{i=1}^{m'} v_i a_i \leq V_j$$

$$\sum_{i=1}^{m'} r_i a_i \leq R_j$$

$$a_{i_1} + a_{i_2} \leq 1 \qquad \forall \{i_1, i_2\} \in E'_p \qquad (3.16)$$

$$a_i \in \{0,1\} \qquad i = 1, 2, \ldots, m'$$

avec m', le nombre de variables restantes, et E'_p, le sous-ensemble obtenu de E_p après les changements dans les contraintes (3.14) expliqués ci-dessus.

En dualisant les contraintes (3.16), on obtient une relaxation lagrangienne qui peut être résolue en ayant recours à un algorithme de sous-gradient. Une solution de $(\mathrm{PA\_}p)'_j$ est trouvée avec un algorithme de séparation et évaluation progressive dont la règle de division est basée sur les contraintes (3.16) qui demeurent non-satisfaites.

Une relaxation lagrangienne pour $(\mathbf{PA\_}p)'_j$

La relaxation lagrangienne que nous proposons consiste à dualiser les contraintes (3.16), qui sont au nombre de $|E'_p|$, en les intégrant à la fonction-objectif:

$$\max_{\lambda_l \geq 0} L(\lambda) \;=\; \min_{a_i} k_j + \sum_{i=1}^{m'} d_{ij}a_i + \sum_{l=1}^{|E'_p|} \lambda_l (a_{i_1} + a_{i_2} - 1)_l$$

$$(\mathbf{RLaux\_}p)'_j \qquad \text{s.c.} \quad \sum_{i=1}^{m'} v_i a_i \leq V_j$$

$$\sum_{i=1}^{m'} r_i a_i \leq R_j$$

$$a_i \in \{0,1\} \qquad i = 1, 2, \ldots, m'$$

où nous avons noté $L(\lambda)$ la fonction duale, λ_l les multiplicateurs de Lagrange, k_j le terme constant $(f_j - u_{m'+j})$ et d_{ij} les coefficients $(w_{ij}v_i - u_i)$ des variables a_i. Il

faut bien sûr se rappeler que la fonction duale $L(\lambda)$ donne une borne inférieure sur la valeur de $(\text{PA\_}p)'_j$.

Un algorithme de sous-gradient permet d'obtenir la solution de cette relaxation. Pour un λ donné, $L(\lambda)$ est un problème de "sac-à-dos" bidimensionnel que nous savons résoudre. Un sous-gradient de $L(\lambda)$ est le vecteur de dimension $|E'_p|$ dont la composante l correspond à l'expression $(a_{i_1} + a_{i_2} - 1)_l$. L'algorithme se termine si le sous-gradient est nul (on a alors $a_{i_1} + a_{i_2} = 1$ pour tout l, c'est-à-dire que toutes les contraintes sont satisfaites) ou si le nombre maximal d'itérations a été atteint. Si toutes les contraintes sont satisfaites après l'application de l'algorithme, alors on a la solution de notre problème auxiliaire $(\text{PA\_}p)'_j$. Dans le cas contraire, on doit trouver la solution de $(\text{PA\_}p)'_j$ par un algorithme de séparation et évaluation progressive.

Algorithme de séparation et évaluation progressive pour $(\text{PA\_}p)'_j$

L'algorithme de séparation et évaluation progressive (B&B) pour ce problème prend en compte les contraintes d'inégalité qui sont violées après la résolution de la relaxation lagrangienne (cette résolution constitue d'ailleurs le calcul de la borne inférieure au noeud 0 de l'arbre de B&B). Dans ce qui suit, on notera λ^* le vecteur des multiplicateurs de Lagrange après l'application de l'algorithme de sous-gradient.

On considère, à chaque noeud de l'arbre de B&B, un sous-problème qui correspond à $(\text{PA\_}p)'_j$ avec des contraintes additionnelles résultant de l'application successive de la règle de division. Ces contraintes se ramènent en fait à une liste de variables dont la valeur est fixée (à 0 ou 1).

La **borne inférieure** sur la valeur d'un sous-problème est obtenue en résolvant $L(\lambda^*)$, qui est un problème de "sac-à-dos" bidimensionnel, avec les contraintes additionnelles résultant de la division. Il est facile d'en tenir compte, puisque, comme

il est expliqué ci-dessous, ces contraintes se ramènent à une liste de variables dont la valeur est fixée à 0 ou 1.

S'il reste des contraintes violées après le calcul de la relaxation du sous-problème, alors on doit le décomposer en appliquant la règle de **division**. On choisit donc une contrainte violée, c'est-à-dire une contrainte pour laquelle $(a_{i_1}+a_{i_2}-1)=1$, et on utilise les contraintes de division suivantes, qui se ramènent en fait à des variables fixées:

$$a_{i_1}=1,\ a_{i_2}=0 \tag{3.17}$$

pour un sous-problème (sur une branche) et

$$a_{i_1}=0,\ a_{i_2}\ \text{libre} \tag{3.18}$$

pour l'autre sous-problème (sur l'autre branche). Le fait de poser des variables à 1 ou 0 ne produit évidemment pas de difficulté supplémentaire pour la résolution du sous-problème. La structure de problème de "sac-à-dos" bidimensionnel de $L(\lambda^*)$ est préservée.

La règle de **branchement** est déterminée par un parcours de l'arbre selon le critère "profondeur d'abord".

Remarque: Le rôle du calcul de la relaxation lagrangienne au premier noeud de l'algorithme de séparation et évaluation progressive est de minimiser le nombre de noeuds dans l'arbre de recherche en obtenant une bonne borne. Il est clair qu'il est question ici d'un compromis entre, d'une part, le temps de calcul dans l'algorithme de sous-gradient et, d'autre part, le temps de calcul dans l'algorithme de séparation et évaluation. Dans ce compromis interviendra certainement toutes sortes de considérations, à commencer bien sûr par le nombre de contraintes d'inégalité et la qualité de la convergence de l'algorithme de sous-gradient (cet algorithme exige en

effet l'ajustement précis de nombreux paramètres). Il est donc possible que la relaxation lagrangienne ne soit pas toujours utile et que le temps de calcul que l'on doive passer dans le sous-gradient ne soit pas justifié. En fait, c'est ce qu'ont montré dans notre cas les résultats préliminaires (nous donnerons des justifications à ce sujet au chapitre 4). Si bien que nous avons laissé tomber l'évaluation de la relaxation lagrangienne au premier noeud de l'algorithme de séparation et évaluation, tel que décrit ci-dessus. La relaxation consiste alors à simplement ignorer les contraintes d'inégalité. Pour le reste, les explications qui sont données ci-dessus restent valables, à condition de poser $\lambda^* = \mathbf{0}$ (cf. "Algorithme de séparation et évaluation progressive pour $(\mathrm{PA\_}p)'_j$").

3.3 Complément: relaxation lagrangienne et génération de colonnes

Nous nous intéressons dans cette section à l'équivalence (par dualité) entre la résolution de la relaxation lagrangienne de (M1) par une méthode de plans coupants et la résolution par génération de colonnes de la relaxation continue de (PM). Il s'agit d'un principe général bien connu que nous voulons simplement illustrer ici à partir de notre problème. On consultera par exemple le livre de Minoux [15] pour un exposé de ce principe. Nous tenons à remercier Olivier du Merle pour sa collaboration dans l'écriture de cette section.

Considérons la relaxation lagrangienne de (M1) où les contraintes de partitionnement (1.2) sont dualisées. On obtient le problème (P) suivant:

$$\max_{\lambda} \min_{x_{ij}, y_j} \sum_{j=1}^{n} f_j y_j + \sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij} v_i x_{ij} - \sum_{i=1}^{m} \lambda_i \left(\sum_{j=1}^{n} x_{ij} - 1 \right)$$

$$\text{(P)} \qquad \begin{aligned} \text{s.c.} \quad & \sum_{i=1}^{m} v_i x_{ij} \leq V_j && j = 1, 2, \ldots, n \\ & \sum_{i=1}^{m} r_i x_{ij} \leq R_j && j = 1, 2, \ldots, n \\ & x_{ij} \leq y_j && i = 1, 2, \ldots, m;\ j = 1, 2, \ldots, n \\ & x_{ij} \in \{0, 1\} && i = 1, 2, \ldots, m;\ j = 1, 2, \ldots, n \\ & y_j \in \{0, 1\} && j = 1, 2, \ldots, n \end{aligned}$$

où $\lambda \in \mathbb{R}^m$ représente le vecteur composé des multiplicateurs λ_i.

Après réarrangement des termes, la fonction-objectif de ce problème peut s'écrire

$$\max_{\lambda} \sum_{i=1}^{m} \lambda_i + \sum_{j=1}^{n} \left(\min_{x_{ij}, y_j} f_j y_j + \sum_{i=1}^{m} \left(w_{ij} v_i - \lambda_i \right) x_{ij} \right)$$

et on peut le formuler comme ci-dessous:

$$\text{(P)} \qquad \max_{\lambda} \; G(\lambda) = \sum_{i=1}^{m} \lambda_i + \sum_{j=1}^{n} g_j(\lambda)$$

où $G(\lambda)$ représente la fonction duale et $g_j(\lambda)$ représente le problème de minimisation associé au site j suivant:

$$
\begin{aligned}
g_j(\lambda) \;=\; \min \;\; & f_j y + \sum_{i=1}^{m} (w_{ij} v_i - \lambda_i)\, x_i \\
\text{s.c.} \;\; & \sum_{i=1}^{m} v_i x_i \le V_j \\
& \sum_{i=1}^{m} r_i x_i \le R_j \\
& x_i \le y \qquad\qquad i = 1, 2, \ldots, m; \\
& x_i \in \{0, 1\} \qquad i = 1, 2, \ldots, m; \\
& y \in \{0, 1\}
\end{aligned}
$$

Nous devons faire quelques remarques à propos de $g_j(\lambda)$. D'abord, il s'agit d'une fonction concave par morceaux. Ensuite, comme la solution nulle est réalisable, on a forcément $g_j(\lambda) \le 0$. Finalement, on note que $g_j(\lambda)$ est, à un terme constant près et avec $y = 1$, de la même forme que le problème auxiliaire $(\mathrm{PA})_j$.

Nous voulons illustrer le fait que la résolution de (P) par une méthode de plans coupants revient, par dualité, à résoudre la relaxation continue de (PM) par génération de colonnes (une coupe équivalant à une colonne). Introduisons des variables z_j dans (P):

$$
\begin{aligned}
\max_{\lambda} \;\; & G(\lambda) \;=\; \sum_{i=1}^{m} \lambda_i + \sum_{j=1}^{n} z_j \\
\textbf{(P)} \qquad \text{s.c.} \;\; & z_j \le g_j(\lambda) \qquad\qquad j = 1, 2, \ldots, n.
\end{aligned}
$$

Plaçons nous maintenant dans le cadre d'une méthode de génération de plans coupants, à une itération k quelconque. La concavité de $g_j(\lambda)$ nous permet d'écrire

$$
g_j(\lambda) \;\le\; g_j(\lambda^k) \;+\; \nabla g_j(\lambda^k)\,(\lambda - \lambda^k) \qquad j = 1, 2, \ldots, n
$$

où λ^k est le vecteur des multiplicateurs à l'itération k, et on peut alors écrire

$$
z_j \;\le\; g_j(\lambda) \;\le\; \gamma_j^k \;-\omega_j^{kT} \lambda \qquad\qquad j = 1, 2, \ldots, n
$$

avec

$$\gamma_j^k \in \mathbb{R} \quad = g_j(\lambda^k) - \nabla g_j(\lambda^k)\lambda^k \qquad j = 1, 2, \ldots, n \tag{3.19}$$

$$\omega_j^k \in \mathbb{R}^m \quad = -\nabla g_j(\lambda^k)^T \qquad j = 1, 2, \ldots, n. \tag{3.20}$$

ce qui nous donne, à l'itération k, les coupes

$$z_j + {\omega_j^k}^T \lambda \leq \gamma_j^k \qquad j = 1, 2, \ldots, n.$$

La méthode consiste donc à générer, à chaque itération, n coupes pour (P), chacune d'elles étant associée à un site j. À l'itération k, l'approximation linéaire de (P) s'écrit

$$\begin{aligned} \max_{\lambda} \quad & G(\lambda) = \lambda^T \mathbb{1} + \sum_{j=1}^{n} z_j \\ (\mathbf{P}^k) \qquad & \text{s.c.} \qquad z_j \mathbb{1} + \Omega_j^T \lambda \leq \gamma_j \qquad j = 1, 2, \ldots, n \end{aligned}$$

où Ω_j est la matrice m par k composée des vecteurs-colonnes $\omega_j^{k'}$ $(k' = 1, 2, \ldots, k)$, et γ_j est le vecteur-colonne de dimension k composé des $\gamma_j^{k'}$ $(k' = 1, 2, \ldots, k)$:

$$\Omega_j = (\omega_j^1, \omega_j^2, \ldots, \omega_j^k)$$

et

$$\gamma_j = (\gamma_j^1, \gamma_j^2, \ldots, \gamma_j^k)^T.$$

Le problème dual de (P^k), considérant que $g_j(\lambda) \leq 0$, donc que $z_j \leq 0$, s'écrit:

$$
\begin{aligned}
& \min \quad \gamma_1^T x_1 + \gamma_2^T x_2 + \ldots + \gamma_n^T x_n \\
& \text{s.c.} \quad \Omega_1 x_1 + \Omega_2 x_2 + \ldots + \Omega_n x_n = \mathbb{1} \\
& \qquad x_1^T \mathbb{1} \leq 1 \\
(\mathbf{D}^k) \qquad & \qquad x_2^T \mathbb{1} \leq 1 \\
& \qquad \ldots \\
& \qquad x_n^T \mathbb{1} \leq 1 \\
& x_j \geq \mathbf{0} \qquad j = 1, 2, \ldots, n
\end{aligned}
$$

où x_j est le vecteur-colonne de dimension k composé des variables duales $x_j^{k'}$ ($k' = 1, 2, \ldots, k$):

$$x_j = (x_j^1, x_j^2, \ldots, x_j^k)^T.$$

On remarquera que les variables λ_i dans (P^k) sont les variables duales associées à la contrainte de partitionnement de (D^k).

Ce problème est la relaxation continue de notre problème maître (PM) après k itérations (restreint), c'est-à-dire après l'insertion de k colonnes pour chaque site ($k*n$ colonnes au total). Chaque colonne de ce problème est constituée des affectations de secteurs à un site (vecteur ω_j^k). et son coût (γ_j^k) représente le coût total de l'affectation, incluant le coût d'ouverture du site. Les équations (3.20) définissent un vecteur-colonne dont les composantes sont les valeurs x_i^* solutions de $g_j(\lambda^k)$. Les équations (3.19) définissent le coût de la colonne dont l'affectation est donné par ω_j^k. Soient x^* et y^* les vecteurs solutions du problème $g_j(\lambda^k)$. On a en effet

$$\omega_j^k = -\nabla g_j(\lambda^k)^T = x^*$$

et

$$\gamma_j^k = g_j(\lambda^k) - \nabla g_j(\lambda^k)\lambda^k$$

$$= f_j y^* + \sum_{i=1}^{m} w_{ij} v_i x_i^*$$

qui est bien le coût de la colonne. Le problème $g_j(\lambda^k)$ est le problème auxiliaire associé au site j et les variables λ_i^k correspondent aux variables duales u_i dans notre notation usuelle.

Nous venons ainsi d'illustrer l'équivalence duale entre la résolution de la relaxation continue de (PM) par génération de colonnes et la résolution de la relaxation lagrangienne de (M1) par une méthode de plans coupants. Ce point de vue sur la décomposition peut avoir comme avantage d'apporter un éclairage différent sur la technique de génération de colonnes, pouvant éventuellement donner lieu à de nouveaux développements. Par exemple, nous l'avons utilisé pour obtenir, dans notre algorithme de génération de colonnes, une borne inférieure sur la valeur optimale de la relaxation continue de (PM). Cette borne inférieure peut, entres autres choses, être utilisée pour définir un critère d'arrêt basé sur la dualité.

3.3.1 Application: critère d'arrêt et borne inférieure

Rappelons que l'algorithme de génération de colonnes est utilisé pour résoudre la relaxation continue de (PM), que nous noterons à nouveau PLM. Le critère d'arrêt de cet algorithme est généralement basé sur la valeur du coût réduit. Nous définissons ici un critère d'arrêt basé sur le saut de dualité, c'est-à-dire sur la différence entre une borne supérieure (sur la valeur optimale de PLM), fournie par la valeur courante du PLM restreint, et une borne inférieure, fournie par la valeur de $G(\lambda^k)$, calculée à chaque itération. Lorsque l'écart entre ces deux valeurs devient assez petit, on arrête la génération de colonnes et la solution courante du PLM restreint est optimale (il s'agit de la valeur optimale de PLM). On peut également utiliser cette borne inférieure pour sonder (par dominance) un noeud dans notre méthode de séparation

et évaluation progressive pour la résolution de (PM): à un noeud donné, on cesse la génération de colonnes et on sonde aussitôt que la valeur de la borne inférieure calculée dépasse la valeur de la meilleure solution connue dans l'arbre de B&B.

Il est clair que la valeur courante du PLM restreint est une borne supérieure sur la valeur optimale de PLM, qui est la valeur du problème (P), ou bien $G(\lambda^*)$. Une borne inférieure, à l'itération k, est donc $G(\lambda^k)$. On a en effet

$$G(\lambda^k) \leq G(\lambda^*) = v(P) \leq v(P^k)$$

où λ^* est la solution optimale du problème (P), $v(P)$ est la valeur optimale du problème (P) et $v(P^k) = v(D^k)$ est la valeur courante du PLM restreint.

La borne inférieure peut être aisément calculée avec les coûts réduits des colonnes fournies par les n problèmes auxiliaires et les variables duales, conformément à l'équation

$$G(\lambda^k) = \sum_{i=1}^{m} \lambda_i^k + \sum_{j=1}^{n} g_j(\lambda^k).$$

Des résultats préliminaires ont montré que l'application de ce critère d'arrêt pouvait donner lieu à une modeste économie d'itérations. Nous en verrons plus à ce sujet au chapitre 4.

CHAPITRE 4
Résultats

On trouve dans ce chapitre deux types de résultats: des résultats généraux et des résultats d'expériences qui ont été effectuées à titre de comparaison. Les résultats généraux ont été produits à partir d'une implantation (dite "version de base") de la méthode de résolution décrite dans le chapitre précédent et appliquée au modèle de génération de colonnes. Dans le cas des comparaisons (sauf pour la section 4.3), la version de base est confrontée à une autre version, différente de la première sur un point de détail, précisé à la section concernée. On commence par énoncer les caractéristiques informatiques de la version de base. Nous décrivons ensuite les problèmes-tests utilisés pour produire nos résultats.

4.1 Implantation informatique

La méthode de résolution décrite au chapitre précédent a été codée en C et compilée en mode optimisé avec le compilateur GNU gcc (version 2.7). Les résultats ont été produits sur une station de travail SUN SPARC 10 (135.5 Mips; 27.3 Mflops; 192 MB RAM) utilisant le système d'exploitation SUN OS (version 5.5).

Voici les caractéristiques principales de la version de base:

- L'algorithme de génération de colonnes utilisé pour résoudre la relaxation continue de (PM) et des sous-problèmes de l'arbre de B&B, décrit à la section 3.1.1, a été implanté tel quel:

- À l'étape d'Initialisation, l'ensemble de colonnes de départ est fourni par une méthode gloutonne qui consiste à affecter les secteurs aux sites, en commençant par les sites de capacité horaire la plus faible, dans l'ordre croissant du coût de transport ($c_i v_i$) de la neige;
- À l'Étape 1, le PLM restreint courant est résolu en ayant recours à la librairie CPLEX (version 2.1);
- À l'Étape 3, les n problèmes auxiliaires sont résolus exactement et les problèmes auxiliaires modifiés sont résolus sans faire de relaxation lagrangienne (cf. remarque à la section 3.2.3 et justifications à la section 4.4.3);
- À l'Étape 4, toutes les colonnes de coût réduit le plus négatif (parmi les n possibles) sont insérées dans la base du PLM restreint (cf. section 4.5);
- Le critère d'arrêt (Étape 3) de la génération de colonnes est basé sur l'absence de colonnes de coût réduit négatif (cf. section 4.6);
- Le nombre de colonnes dans la matrice du problème (base de données CPLEX) est limité à 500;
- Lorsque cette limite est atteinte, on élimine la moitié des colonnes hors-base (en commençant par les colonnes de coût réduit le plus positif).

- Le parcours de l'arbre de B&B dans l'algorithme de séparation et évaluation progressive résulte du critère d'exploration "profondeur d'abord" pur. Pour ce faire, on utilise la récursivité.

- La règle de division (cf. section 3.2.2) est appliquée de la même manière à chaque noeud: sur la branche de gauche, on considère les contraintes de type (3.5), alors que sur la branche de droite, on considère les contraintes de type (3.6). On commence toujours l'exploration par la branche de gauche.

- Pour trouver les secteurs i_1 et i_2 utilisés dans la règle de division (cf. proposition 3.2.2), on commence par l'examen de la première colonne fractionnaire

de la base et l'on poursuit, s'il y a lieu, selon l'ordre croissant des indices de colonnes.

4.2 Problèmes-tests

Définir un problème-test revient à donner les caractéristiques des m secteurs (v_i, volume annuel de neige à enlever et r_i, taux horaire d'enlèvement de la neige) et des n sites potentiels (R_j et V_j, capacités de stockage horaire et annuelle) ainsi que les différents coûts (f_j, b_j et c_{ij}) reliés au problème, conformément aux définitions exposées au premier chapitre.

Pour produire les problèmes-tests, nous nous sommes inspiré des données dont nous disposons sur la situation à Montréal (cf. Campbell et Langevin [2]), en supposant que celle-ci corresponde dans l'ensemble assez bien aux différents cas de déneigement urbain que l'on rencontre en pratique. Il est à noter que nous ne disposons pas des données relatives aux coûts (les f_j, b_j et c_{ij}). Pour le reste, en voici un résumé:

- Montréal possède 60 secteurs et utilise 20 sites;
- Le volume annuel moyen de neige à enlever dans un secteur est de 130000 m^3;
- Le taux moyen d'enlèvement de la neige dans un secteur est de 600 $m^3/heure$;
- Sur les 20 sites, 13 ont une capacité annuelle de stockage illimitée;
- Les autres sites disposent en moyenne d'une capacité annuelle de stockage de 879000 m^3;
- La capacité horaire moyenne de stockage pour un site est de 3095 $m^3/heure$.

Remarque 1: Sur les 13 sites de capacité annuelle illimitée, 3 sont des sites sur le fleuve (largement utilisés) qui devront éventuellement être fermés pour se conformer à une nouvelle réglementation environnementale interdisant les déversements de neiges usées dans les cours d'eau. Il est difficile de présumer de la situation future, mais il est vraisemblable que le nombre de sites augmente, et que la capacité annuelle moyenne sur l'ensemble des sites diminue. Il est difficile de se prononcer sur le cas de la capacité horaire moyenne.

Remarque 2: La capacité annuelle moyenne de stockage des autres sites tient compte de la très grande capacité (3000000 de m^3) d'une ancienne carrière (carrière Miron) utilisée comme site de déversement.

Nous avons généré les problèmes-tests en nous basant sur ces caractéristiques, avec cependant quelques modifications dont nous croyons qu'elles permettront, en tenant compte des remarques ci-dessus, de refléter plus exactement la situation qui prévaudra dans le futur (ou dans d'autres situations urbaines) et qui illustrent aussi notre volonté de rendre le problème un peu plus difficile (plus intéressant). En ce qui concerne les secteurs, nous n'avons rien changé par rapport à la situation de Montréal. Les caractéristiques des sites sont celles qui diffèrent le plus de la réalité montréalaise actuelle: en plus de limiter à 20% le nombre de sites de capacité annuelle illimitée, nous avons abaissé la capacité annuelle moyenne pour les autres sites. Par ailleurs, nous avons volontairement opté pour un nombre de sites potentiels restreint. Voici donc les caractéristiques (sauf pour les coûts) des secteurs et des sites pour les problèmes-tests générés:

- Le nombre de secteurs varie de 50 à 80;
- Le nombre de sites potentiels varie de 15 à 25;
- Le volume annuel de neige à enlever (en m^3) dans un secteur est généré

aléatoirement dans l'intervalle [91000. 169000];

- Le taux d'enlèvement de la neige (en m^3/h) dans un secteur est généré aléatoirement dans l'intervalle [480. 720];
- La capacité annuelle de stockage (en m^3) dans un site où elle n'est pas illimitée est générée aléatoirement dans l'intervalle [200000. 900000];
- La capacité horaire de stockage (en m^3/h) dans un site est générée aléatoirement dans l'intervalle [1000. 7000].

Reste alors la question des coûts. Nous n'avons pu les obtenir et il ne nous a pas été possible d'en faire une estimation rigoureuse. Dans ces conditions, nous supposons (c'est une hypothèse vraisemblable) que le coût de transport unitaire (c_{ij}) de la neige entre le secteur et le site est proportionnel à la distance qui les sépare. On peut donc utiliser cette distance comme coût. Ensuite, on pose le coût variable d'utilisation du site j (b_j) à 0 dans tous les problèmes-tests. Nous allons éclaircir plus bas le cas du coût fixe annuel d'utilisation (f_j).

Soit d_{ij} la distance (en km) entre le (centre du) secteur i et le site j. On suppose une relation de proportionnalité entre le coût unitaire de transport c_{ij} et d_{ij}. On a donc $d_{ij} = K c_{ij}$, où K est une constante ($m^3\ km/\$$). En prenant la distance d_{ij} plutôt que c_{ij}, le coût "de transport" calculé, pour un site j, est de

$$\sum_{i=1}^{m} v_i d_{ij} x_{ij} = \sum_{i=1}^{m} v_i K c_{ij} x_{ij} = K \sum_{i=1}^{m} v_i c_{ij} x_{ij},$$

ce qui ne cause évidemment pas de problème (les affectations, si l'hypothèse de la proportionnalité est correcte, restent les mêmes). On génère donc aléatoirement les sites et les secteurs sur une région rectangulaire de dimensions 30 km par 20 km, ce qui nous permet d'obtenir les distances d_{ij} utilisées comme coûts.

En toute rigueur, il faudrait alors considérer, dans nos calculs, un coût fixe annuel d'utilisation f'_j du site j tel que $f'_j = K f_j$, c'est-à-dire un coût d'utilisation proportionnel au coût d'utilisation réel f_j. Le problème est que nous ne connaissons pas ce dernier, bien qu'il soit probable que le coût d'utilisation d'un site (en excluant les coûts environnementaux) ne représente qu'une fraction du coût total de transport de la neige à ce site. Il est clair que la valeur de ce coût est critique, puisqu'il règle l'aspect "localisation" du problème, permettant une plus ou moins grande discrimination dans le choix des sites. L'ordre de grandeur de ce coût a par conséquent un effet décisif sur le temps de calcul et sur la performance globale de l'algorithme. L'idéal serait donc de faire une analyse de sensibilité pour étudier la performance de l'algorithme selon l'ordre de grandeur de ce coût. Il ne nous a pas été possible, faute de temps, de le faire. Des tests préliminaires ont cependant permis d'observer une augmentation rapide du temps de calcul lorsque l'ordre de grandeur du coût d'utilisation des sites dépasse 10e5. Pour des problèmes-tests où le nombre de sites est faible par rapport au nombre de secteurs, on note un temps de calcul considérable et une détérioration des conditions de résolution. Compte tenu du nombre de sites potentiels assez faible que nous nous sommes fixé, nous avons préféré ne pas aller au-delà d'un ordre de grandeur de 10e5. Nous reviendrons sur les implications de ce choix plus loin dans ce chapitre et au chapitre 5, où l'on présente quelques résultats sur l'effet de l'augmentation de l'ordre de grandeur du coût d'utilisation.

Nous avons également décidé de poser ce coût fixe d'utilisation proportionnel à la capacité horaire du site (autre hypothèse vraisemblable). Nous avons en effet opté pour la formule suivante, fournissant des coûts d'ouverture de l'ordre de 10e5:

$$f_j\ (m^3\ km)\ =\ 0,01 * [R_j/600\ m^3/h] * 130000\ m^3 * 15\ km. \qquad (4.1)$$

Voici notre interprétation de cette formule qui peut sembler exotique. Nous

considérons $[R_j/600\ m^3/h] * 130000\ m^3$ comme le nombre moyen de m^3 de neige pouvant être acheminés au site j. On suppose en effet que le volume de neige moyen est de 130000 m^3 et que le taux moyen d'enlèvement de cette neige est de 600 m^3/h. On multiplie ce nombre par une distance qu'on pourra considérer comme étant moyenne, soit 15 km. Ensuite, on divise le tout par 100 pour obtenir un coût f_j qui se situe approximativement dans l'intervalle [10e4, 10e5]: on considère donc que le coût d'ouverture du site est, *grosso modo* et *a priori*, 100 fois moindre que le coût moyen total de transport de la neige à ce site. Il est entendu que plus le rapport entre le coût d'ouverture et le coût total de transport est grand, plus le problème est difficile, comme il sera montré au chapitre 5.

4.3 Comparaison expérimentale des valeurs des relaxations continues de (M1) et (PM)

Cette section a pour but de compléter le résultat théorique énoncé à la section 2.3 concernant les valeurs des relaxations continues des modèles (M1) et (PM). La proposition 2.3.2 affirme que la valeur de la solution continue de (PM) est théoriquement plus grande ou égale à la valeur de la solution continue de (M1). Les résultats que nous montrons ici complètent cette proposition dans la mesure où ils nous permettent d'exclure la possibilité de l'équivalence des valeurs. De plus, ils nous donnent une idée de l'écart réel existant entre celles-ci.

Pour ce faire, nous comparons, pour 5 classes de problèmes-tests (chacune comprenant 20 problèmes), le saut d'intégralité moyen obtenu pour les deux modèles. Le saut d'intégralité (en %) est le rapport de la différence des valeurs de la solution entière et de la solution continue sur la valeur de la solution entière, le tout multiplié par 100. Pour (M1), la solution continue a été obtenue en utilisant le logiciel CPLEX

interactif. Rappelons que la solution entière de (M1) est égale à la solution entière de (PM). On trouve les résultats au tableau 4.1 ci-dessous.

Tableau 4.1 Saut d'intégralité pour (M1) et (PM)

		saut	
		M1	PM
n	m	(%)	(%)
15	60	4,3	0,09
20	50	3,4	0,07
	65	4,2	0,07
	70	5,2	0,09
25	70	4,9	0,06

On peut remarquer que

(i) la borne inférieure que constitue la valeur de la solution de la relaxation continue de (PM) est meilleure que la borne obtenue à partir de la relaxation continue de (M1);

(ii) l'écart entre les sauts d'intégralité est très appréciable;

(iii) le saut d'intégralité pour (PM) est très faible.

4.4 Résultats généraux

Nous avons testé notre méthode de résolution sur 11 classes de problèmes-tests (chacune comprenant 20 problèmes). On peut regrouper ces classes en 3 catégories, selon le nombre n de sites potentiels disponibles (15, 20 et 25). Pour chacune de ces catégories, on a fait varier (plus ou moins, selon le cas) le nombre m de secteurs. Nous présentons au tableau 4.2 de la section 4.4.1 les résultats principaux obtenus

pour chaque classe. Il s'agit de résultats portant sur la résolution complète (en nombres entiers) des problèmes. La section 4.4.2 est consacrée aux caractéristiques de la relaxation continue des problèmes, que l'on trouve résumées au tableau 4.3. Nous faisons quelques remarques sur les résultats obtenus à la section 4.4.3.

4.4.1 Résultats principaux

Le tableau 4.2 ci-dessous présente les résultats obtenus pour la résolution complète des problèmes-tests.

Tableau 4.2 Résultats principaux

		temps de calcul									
		total	CPLEX	aux.				it. ut.			
n	m	(s)	(%)	(%)	noeuds	prof.	it.	(%)	col.	pivots	piv/it
15	50	42,5	23	77	6,6	1,3	113	50	812	3274	32
	60	226	14	86	12,1	2,7	214	43	1350	7110	36
20	50	9,2	41	59	5,0	1,5	85	51	841	2386	30
	55	16,2	38	62	5,9	1,5	99	49	961	2851	34
	60	22,8	30	70	3,9	1,2	80	49	901	2959	38
	65	124	23	77	7,9	2,0	138	47	1257	5085	41
	70	181	19	81	11,8	2,6	196	45	1563	7299	44
	75	222	18	82	8,8	2,2	168	45	1575	6565	44
25	70	38,5	29	71	8,1	2,1	112	51	1230	4307	45
	80	81	29	71	9,5	2,0	167	51	1582	6331	49
	90	290	20	80	7,2	2,6	166	47	1732	7648	51

Les classes de problèmes sont définies par les paramètres n et m. Toutes les autres valeurs de ce tableau sont des moyennes pour les 20 problèmes de chaque classe. La section "temps de calcul" donne le temps de calcul total (en secondes) pour résoudre le problème et la fraction du temps de calcul qui revient à CPLEX (Étape 1 de l'algorithme de génération de colonnes) et à la résolution des problèmes auxiliaires (Étape 3). L'autre section du tableau donne le nombre total de noeuds

explorés dans l'arbre de B&B, la profondeur maximale de l'arbre (une profondeur de 0 indiquant un problème résolu au premier noeud), le nombre total d'itérations de l'algorithme de génération de colonnes, la fraction de ces itérations qui ont fait baisser la valeur de la fonction-objectif (on appelle ces itérations "itérations utiles"), le nombre total de colonnes générées, le nombre total de pivots CPLEX (Étape 1) et le rapport du nombre de pivots sur le nombre d'itérations.

On note que, pour un n donné,

(i) le temps de calcul augmente avec m;

(ii) le temps de calcul subit un changement d'ordre de grandeur (saut marqué) à partir d'un certain nombre m (qu'on pourra par exemple situer à 65 pour le cas où $n = 20$);

(iii) la fraction du temps de calcul qui est utilisée pour la résolution des problèmes auxiliaires augmente avec m;

(iv) la fraction du temps de calcul qui est utilisée par CPLEX diminue avec l'augmentation de m, bien que celui-ci augmente en valeur absolue;

(v) la fraction des itérations qui sont utiles diminue avec l'augmentation de m;

(vi) le nombre de pivots CPLEX par itération augmente avec m;

(vii) les échantillons utilisés ne permettent pas de conclure de façon définitive à une croissance stricte du nombre de noeuds, d'itérations, de colonnes générées et de pivots avec m, bien qu'il semble exister une tendance à l'augmentation.

On remarque également que, pour un m donné, le temps de calcul et la fraction utilisée pour la résolution des problèmes auxiliaires diminue lorsque n augmente.

4.4.2 Caractéristiques de la relaxation continue

Pour les mêmes problèmes-tests que ci-dessus (regroupés de la même manière), on a résumé au tableau 4.3 certaines caractéristiques de leur relaxation continue. Ces résultats, sauf pour la sixième colonne, sont des moyennes.

Tableau 4.3 Caractéristiques de la relaxation continue

		contribution de la relaxation				
		temps total	col.	it.		saut
n	m	(%)	(%)	(%)	nb. ent.	(%)
15	50	39	58	44	12/20	0,07
	60	21	56	39	8/20	0,09
20	50	58	72	53	8/20	0,07
	55	53	70	53	10/20	0,07
	60	59	74	55	11/20	0,06
	65	42	67	49	7/20	0,07
	70	28	57	38	8/20	0,09
	75	38	63	43	8/20	0,06
25	70	47	68	47	9/20	0,06
	80	43	60	41	11/20	0,05
	90	45	73	55	7/20	0,05
moyenne		43	65	47	9/20	0,07

La section "contribution de la relaxation" donne la fraction du temps de calcul total qui est consacrée à la résolution de la relaxation continue, la fraction du nombre total de colonnes générées qui l'ont été pendant la résolution de la relaxation continue et la fraction du nombre total d'itérations de génération de colonnes qui ont été produites pendant la résolution de la relaxation continue. On ne tient compte dans cette section du tableau que des problèmes qui ont un saut d'intégralité non nul, c'est-à-dire pour lesquels la solution entière ne coïncide pas avec la solution continue. On complète le tableau en fournissant le nombre de problèmes pour lesquels la solution continue est entière et le saut d'intégralité, tel que défini à la section 4.3, ce dernier tenant compte de tous les problèmes-tests. On trouve en dernière ligne

la moyenne globale (sur l'ensemble des classes) de ces valeurs.

On observe que

(i) la part consacrée à la résolution de la relaxation continue compte en moyenne pour près de la moitié des itérations, plus des deux-tiers des colonnes générées et plus des deux-cinquièmes du temps de calcul;

(ii) le nombre de problèmes pour lesquels la solution continue est entière représente en moyenne près de la moitié des cas;

(iii) le saut d'intégralité est très faible;

(iv) pour un n donné, le saut d'intégralité n'augmente pas avec le nombre de secteurs m.

4.4.3 Remarques générales sur la résolution des problèmes-tests

Nous voudrions ici faire des remarques d'ordre général sur les résultats que nous avons obtenus ainsi que sur la méthode de résolution, telle qu'implantée dans la version de base. Nous discuterons entre autres choses de la difficulté des problèmes, de la variabilité dans les résultats et de dégénérescence. Nous nous demanderons dans quelle mesure la méthode de résolution utilisée est validée par les résultats.

La méthode de résolution utilisée nous permet de résoudre une majorité de problèmes-tests dans un temps "raisonnable". Il est évidemment difficile de définir ce qu'est un temps raisonnable, compte tenu du fait qu'il n'existe pas de résultats avec lesquels il serait possible de faire des comparaisons (le seul résultat utilisable concerne le temps de calcul obtenu par Campbell et Langevin [2] pour la résolution sur CPLEX du problème d'affectation (sans localisation) pour Montréal, soit 4.5

heures, tel que mentionné à la section 1.2.3). Mais il ne nous semble (pour le moment) pas déraisonnable d'avoir à attendre en moyenne, au plus 5 minutes pour obtenir la solution d'un problème-test (c'est le cas de la classe $[n = 25;\ m = 90]$, qui a récolté la plus grande valeur moyenne). En fait, le problème-test dont le temps de résolution est le plus grand a été résolu en environ 30 minutes. Il faut cependant garder à l'esprit que la grandeur du coût d'utilisation des sites a un impact significatif sur la difficulté des problèmes (et sur les performances de l'algorithme). Nous reviendrons sur ce sujet au chapitre 5.

Reportons-nous au tableau 4.2. Comme nous l'avons déjà fait remarquer (cf. remarque *(ii)* de la section 4.4.1), on constate que le temps de calcul subit, pour un n donné, un changement d'ordre de grandeur à partir d'un certain m. Ce seuil peut être situé à $m = 65$ pour le cas où $n = 20$, par exemple. On pourrait donc admettre, *a priori*, que ce seuil puisse, pour un n donné, séparer les problèmes "faciles" et les problèmes "difficiles". Pour être plus précis, il serait cependant peut-être utile de distinguer plusieurs types ou niveaux de difficulté. On pourra, par exemple, parler de la difficulté reliée à la résolution de la relaxation continue des sous-problèmes (problèmes de partition de la forme de (PM)), et de la difficulté reliée à la résolution des problèmes auxiliaires. Pour ce faire, on prendra respectivement comme critère le nombre de pivots CPLEX par itération de génération de colonnes et la fraction du temps de calcul total qui est consacrée aux problèmes auxiliaires. Ainsi, les remarques *(iii)* et *(vi)* de la section 4.4.1 permettent de conclure, comme il fallait s'y attendre, à une augmentation de la difficulté des problèmes auxiliaires et de la relaxation continue des sous-problèmes en fonction de l'augmentation de m (pour un n donné). Globalement, ceci se traduit donc par une augmentation (pour un n donné) du temps de calcul par itération lorsque m augmente. Le tableau 4.4 ci-dessous nous permet d'observer comment le temps de calcul par itération évolue.

Pour chaque classe de problèmes-tests, on montre en colonne (1) le temps de calcul total moyen et, en colonne (2), le nombre total moyen d'itérations (tels qu'ils apparaissent au tableau 4.2). La dernière colonne fournit le rapport des deux premières colonnes, tel qu'indiqué. Cette colonne donne une idée de l'évolution du temps de calcul par itération. On constate que ce temps augmente, pour un n donné, en fonction de m. Également, notons que le temps de calcul par itération est, pour un m donné, plus grand lorsque n est petit.

Tableau 4.4 Variation du temps de calcul par itération

n	m	(1) temps total (s)	(2) nombre total d'itérations	(1)/(2) (s)
15	50	42,5	113	0,38
	60	226	214	1,06
20	50	9,2	85	0,11
	55	16,2	99	0,16
	60	22,8	80	0,28
	65	124	138	0,90
	70	181	196	0,92
	75	222	168	1,32
25	70	38,5	112	0,34
	80	81	167	0,48
	90	290	166	1,75

Les résultats généraux montrent que la part du temps de calcul qui est consacrée aux problèmes auxiliaires est considérable. Il ne faut pas oublier qu'à chaque itération de génération de colonnes, n problèmes auxiliaires sont résolus exactement. Il est donc évident que la performance de l'algorithme de résolution du problème de "sac-à-dos" bidimensionnel est critique. Pour se faire une idée plus exacte de la performance de notre algorithme, on consultera le tableau 4.5 ci-dessous.

Pour chaque classe de problèmes-tests, on montre en colonne (2) le nombre

moyen de problèmes de "sac-à-dos" bidimensionnels (apparaissant dans la résolution des problèmes auxiliaires, modifiés ou non) résolus par problème-test et, en colonne (1), le temps moyen utilisé pour ce faire. La dernière colonne fournit simplement le rapport des deux colonnes, tel qu'indiqué. Cette colonne nous donne une idée du temps que l'algorithme prend (en moyenne) pour résoudre un problème de "sac-à-dos", nous permettant du même coup d'apprécier la variation de la difficulté des problèmes auxiliaires.

Tableau 4.5 Performance du "sac-à-dos" bidimensionnel

n	m	(1) temps *biknap* (s)	(2) nombre de *biknap*	(1)/(2) (ms)
15	50	36,6	1928	19,0
	60	211	3910	54,0
20	50	5,0	1817	2,8
	55	10,9	2114	5,2
	60	17,3	1682	10,3
	65	114	3271	34,8
	70	165	4507	36,6
	75	206	3764	54,7
25	70	29,4	2971	9,9
	80	65,7	4609	14,2
	90	269	4538	59,3

À ce sujet, on peut répéter ici ce que nous avons déjà exprimé plus haut: la dernière colonne montre que, pour un n donné, la difficulté des problèmes auxiliaires augmente en fonction de m. Ceci est évident si on se rappelle que la taille des problèmes dépend précisément du nombre de secteurs m. On constate par ailleurs que, pour un m donné, la difficulté des problèmes auxiliaires augmente lorsque n (le nombre de sites potentiels) diminue. Autrement dit, la difficulté de l'affectation se répercute au niveau des problèmes auxiliaires. On constate que les problèmes les plus difficiles se trouvent dans les classes $[n = 15;\ m = 60]$, $[n = 20;\ m = 75]$ et

[$n = 25$: $m = 90$]. Il s'agit bien sûr de ceux qui ont les temps de calcul les plus élevés dans le tableau 4.2. On remarque pour finir qu'il n'y a pas nécessairement de corrélation entre la difficulté des problèmes auxiliaires et la difficulté de la relaxation continue des sous-problèmes, telle que décrite par le nombre de pivots CPLEX par itération (cf. dernière colonne du tableau 4.2).

Nous voudrions dès maintenant justifier la décision que nous avons prise au sujet de la résolution des problèmes auxiliaires modifiés. Rappelons que nous avons décidé, après avoir obtenus des résultats préliminaires, qu'il ne valait pas la peine de résoudre une relaxation lagrangienne avant d'appliquer l'algorithme de séparation et évaluation (cf. remarque à la section 3.2.4). Le tableau 4.6 ci-dessous nous permet de comprendre pourquoi.

Tableau 4.6 Performance du problème auxiliaire modifié

		(1)	(2)	(3)	(4)
n	m	p. aux.	p. aux. mod. (%) de (1)	b&b n.r. (%) de (2)	nds/b&b
15	50	1695	43	15	3,07
	60	3210	47	13	3,04
20	50	1700	36	11	3,04
	55	1980	36	9	3,01
	60	1600	34	12	3,05
	65	2760	37	9	3,13
	70	3920	49	11	3,07
	75	3360	46	11	3,02
25	70	2800	41	8	3,07
	80	4175	48	9	3,05
	90	4150	34	9	3,02
moyenne		-	41	11	3,05

Pour chaque classe de problèmes-tests, la colonne (1) donne le nombre moyen de problèmes auxiliaires résolus par problème-test, la colonne (2) donne la fraction moyenne de ces problèmes auxiliaires qui sont modifiés par des contraintes

d'inégalité, la colonne (3) donne la fraction moyenne des problèmes auxiliaires modifiés pour lesquels l'algorithme de séparation et évaluation (*branch-and-bound: b&b*) doit se poursuivre après le premier noeud (non réalisable après le premier noeud), et, finalement, la colonne (4) donne le nombre moyen de noeuds (incluant le premier) dans cet algorithme de séparation et évaluation, lorsque celui-ci se poursuit. La dernière ligne du tableau fournit la moyenne globale (sur l'ensemble des classes) des ces valeurs.

On constate que la fraction des problèmes auxiliaires modifiés qui se poursuivent après le premier noeud est relativement faible (11% en moyenne). Autrement dit, la majorité de ces problèmes sont résolus au premier noeud, malgré l'absence de la relaxation lagrangienne. De plus, on note que, lorsqu'il y a poursuite de l'algorithme de séparation et évaluation progressive, le nombre de noeuds dans l'arbre de recherche reste petit, dépassant rarement 3 noeuds (3,05 noeuds en moyenne). Ceci s'explique certainement en partie par le fait que le nombre de contraintes d'inégalité dans les problèmes auxiliaires reste faible, compte tenu de la faible profondeur des arbres de recherche (nous parlons ici de l'arbre de recherche principal, utilisé pour résoudre (PM)). Ces résultats confirment donc que, pour les problèmes qui nous occupent ici, il n'est pas nécessaire d'investir du temps de calcul dans un algorithme de sous-gradient. Il pourrait bien sûr en aller autrement pour d'autres types de problèmes, plus difficiles.

Pour compléter les résultats que nous avons présentés jusqu'ici, nous montrons au tableau 4.7 ci-dessous le nombre moyen de sites utilisés dans la solution optimale (colonne (1)) et le rapport entre cette valeur et le nombre de sites potentiels disponibles, pour toutes les classes de problèmes-tests.

On voit que le nombre relatif de sites utilisés est élevé et que celui-ci, comme il fallait s'y attendre, augmente avec l'augmentation de m (pour un n donné). De

plus, on observe que, pour un m donné, le nombre relatif de sites utilisés diminue lorsque le nombre de sites potentiels n augmente.

Tableau 4.7 Nombre de sites utilisés

n	m	(1) nombre de sites utilisés	(1)/n (%)
15	50	14,0	93
	60	14,5	97
20	50	16,7	84
	55	17,4	87
	60	18,2	91
	65	18,3	92
	70	18,8	94
	75	19,3	97
25	70	22,0	88
	80	23,2	93
	90	23,7	95

On pourrait légitimement se demander si le pouvoir de discrimination de l'algorithme dans la sélection des secteurs n'est pas trop faible, limitant ainsi l'aspect "localisation" du problème. Un coût d'ouverture plus grand aurait en théorie pour effet de réduire le nombre de sites utilisés. Cependant, il ne faut pas oublier que le nombre de sites potentiels est assez restreint. Il est par conséquent possible qu'une réduction significative soit difficile à obtenir, compte tenu des contraintes de capacités, lorsque le nombre de sites est faible par rapport au nombre de secteurs (par exemple pour la classe $[n = 15; m = 60]$). Nous verrons au chapitre 5 quelques résultats permettant d'illustrer l'effet de l'augmentation du coût d'ouverture des sites sur le nombre de sites utilisés. D'autres résultats montrent l'effet de l'augmentation du nombre de sites potentiels sur le temps de calcul.

Notre analyse ne serait pas complète si nous passions sous silence le fait que, à l'intérieur de chaque classe de problèmes, les résultats ont démontré une variabilité

non négligeable. Pour illustrer ceci, on présente au tableau 4.8 ci-dessous les valeurs minimales et maximales observées à l'intérieur de chaque classe de problèmes-tests pour le temps total de calcul, le nombre total de noeuds dans l'arbre de recherche, le nombre total d'itérations de génération de colonnes, le nombre total de colonnes générées et le nombre total de pivots CPLEX.

Tableau 4.8 Étendue des résultats

		MIN					MAX				
n	m	temps total (s)	noeuds	it.	col.	pivots	temps total (s)	noeuds	it.	col.	pivots
15	50	5,2	1	48	507	1591	327	33	382	1885	8837
	60	8,9	1	50	568	2019	1541	71	963	4574	32346
20	50	3,2	1	38	572	1196	25,9	23	244	1637	5013
	55	3,9	1	40	608	1504	78	29	323	2083	7233
	60	7,6	1	49	693	1858	104	19	202	1528	6409
	65	12,2	1	54	807	2616	1172	35	459	3007	16189
	70	15,5	1	53	809	2707	757	55	732	4022	21518
	75	16,3	1	65	899	3218	1685	45	466	3290	16566
25	70	12,5	1	55	835	2813	189	49	323	2319	6735
	80	13,5	1	63	987	3259	524	43	569	3922	19591
	90	35,3	1	63	1019	3576	1710	35	491	3864	22106

Les valeurs moyennes, comme chacun le sait, sont sensibles aux valeurs extrêmes. Il faut donc garder à l'esprit que tous les résultats que nous avons présentés dans les tableaux précédents n'échappent pas à cette règle.

Nous allons maintenant poursuivre avec des remarques sur la dégénérescence dans l'algorithme de génération de colonnes. Nous avons cru possible de quantifier cette dégénérescence en calculant la fraction des itérations de génération de colonnes qui permettent d'abaisser la fonction-objectif (nous les avons appellées "itérations utiles"). Nous avons constaté au tableau 4.2 qu'environ 50% des itérations étaient utiles. Ce qui veut dire que la moitié des itérations ne permettent pas d'abaisser la

fonction-objectif. En fait, nous avons remarqué la présence de nombreux plateaux (dans la valeur de la fonction-objectif) en cours de résolution. Ces plateaux s'avèrent d'ailleurs très fréquents en fin de génération de colonnes, à chaque noeud de l'arbre de recherche (c'est d'ailleurs une des raisons qui ont motivé l'étude d'un critère d'arrêt basé sur la dualité). Il existe également un assez large plateau en début de génération de colonnes au premier noeud, qui correspond, au moins en partie, à l'équivalent de la phase 1 du simplexe. Par ailleurs, nous avons observé que du cyclage était possible: nous avons rencontré quelques cas de cyclage dans des problèmes-tests (celui-ci est d'ailleurs plus fréquent pour les classes de problèmes ayant un temps de calcul plus long). Il semble que les problèmes soient assez sensibles au cyclage: nous avons observé que la modification de divers paramètres dans la méthode de résolution pouvait causer du cyclage dans des problèmes qui, autrement, se comportaient correctement. Précisons que le cyclage avait été confirmé, dès le début du projet, par la simulation avec CPLEX de la méthode de génération de colonnes sur un problème simple. Il est probable que la méthode de stabilisation de la génération de colonnes, proposée par Du Merle *et al.* [5], permette d'obtenir une convergence plus rapide et de pallier la dégénérescence en améliorant les conditions de résolution. Cette méthode combine des stratégies de perturbation et de pénalités exactes en norme l_1 pour assurer la stabilisation et l'accélération de l'algorithme de génération de colonnes. Elle a été utilisée avec succès dans quelques applications.

Nous avons déjà dit que le saut d'intégralité moyen de nos problèmes-tests est très faible. Reportons-nous au tableau 4.3. On observe un saut d'intégralité moyen de 0,07% sur l'ensemble des classes de problèmes-tests. Ce faible saut s'explique en partie par le nombre considérable de problèmes ayant un saut d'intégralité nul. Mais, même en tenant compte de ces problèmes, on obtient toujours un faible saut. Le tableau 4.9 ci-dessous montre, pour toutes les classes, quelques caractéristiques des problèmes qui ont un saut d'intégralité non-nul (qui branchent). On y trouve

le nombre de problèmes-tests (parmi les 20) qui ont branché, le saut d'intégralité moyen pour ces problèmes, ainsi que le nombre total moyen de noeuds dans l'arbre de recherche et sa profondeur moyenne, toujours pour ces mêmes problèmes.

Tableau 4.9 Quelques caractéristiques des problèmes qui branchent

n	m	nombre de problèmes	saut (%)	nombre de noeuds	prof.
15	50	8/20	0,2	15,0	3,3
	60	12/20	0,2	19,5	4,5
20	50	12/20	0,1	7,7	2,5
	55	10/20	0,1	10,8	3,0
	60	9/20	0,1	7,4	2,7
	65	13/20	0,1	11,6	3,1
	70	12/20	0,2	19,0	4,3
	75	12/20	0,1	14,0	3,7
25	70	11/20	0,1	13,9	3,8
	80	9/20	0,1	19,9	4,4
	90	13/20	0,1	10,5	4,0

On constate que le saut d'intégralité moyen reste faible, se situant autour de 0,1%. Il faudrait se demander si le nombre de noeuds, quoique raisonnable (il tourne autour de 13 noeuds en moyenne), ne serait pas trop grand compte tenu de ce faible saut d'intégralité. Ces résultats tendent peut-être à montrer que des améliorations sont possibles en ce qui concerne la règle de division et le critère de branchement. Par exemple, le choix des secteurs dans la règle de division (cf. section 3.2.2) aurait avantage à être étudié de plus près. Il peut en effet exister plusieurs possibilités pour le choix de la paire de secteurs i_1 et i_2 satisfaisant la proposition 3.2.1 (ou 3.2.2) que nous y avons énoncée. En fait Ryan et Foster [17] restent muets sur ce point. Chaque problème ayant ses caractéristiques propres, il n'existe probablement pas de règle générale pour ce faire. Rappelons que ce choix, dans notre version de base, a été fait plus ou moins au hasard (cf. section 4.1). Il est également possible d'envisager un critère de branchement, à un noeud, qui consisterait en une méthode

de profondeur d'abord avec évaluation des deux noeuds fils et choix du meilleur de ceux-ci, c'est-à-dire de celui qui apporte la meilleure borne (la plus plus petite valeur de la relaxation).

Il serait finalement utile de s'interroger sur la possibilité d'utiliser une heuristique pour obtenir une bonne borne supérieure de départ pour l'algorithme de séparation et évaluation progressive. Par exemple, il apparaît qu'une méthode de recherche locale à voisinage variable a permis d'obtenir de bons résultats dans un certain nombre de problèmes d'optimisation combinatoire. Voir à ce sujet le travail de Mladenović et Hansen [16].

4.5 Effet du nombre de colonnes insérées

Nous allons voir dans cette section que le fait d'insérer, comme nous le faisons dans la version de base, toutes les colonnes de coût réduit négatif (parmi les n possibles) dans la base du PLM restreint (cf. Étape 4 de l'algorithme de génération de colonnes) est, sans aucun doute, la meilleure option. Pour ce faire, on compare ici cette version de base à une autre version dans laquelle on n'insère que la colonne de coût réduit le plus négatif seulement. De manière à établir une comparaison qui soit valable, nous n'avons retenu que les 11 problèmes-tests de la classe $[n = 20; m = 60]$ dont la solution est optimale dès le premier noeud (saut d'intégralité nul). Ainsi, les effets du branchement n'interviennent pas dans les résultats de la comparaison, que l'on trouve résumés au tableau 4.10 ci-dessous.

La première colonne donne le numéro du problème-test. Pour chaque problème, on montre dans ce tableau deux lignes de résultats. La première correspond à la version où une seule colonne (la meilleure parmi celles de coût réduit négatif) est

Tableau 4.10 Comparaison 1/20 colonnes insérées

		temps de calcul						
		total	CPLEX	aux.		it. ut		
		(s)	(s)	(s)	it.	(%)	col.	pivots
1	1	50.4	10,3	39,6	417	57	416	3024
	20	14,0	3,9	9,8	60	45	760	2287
2	1	52,6	11,8	40,1	447	59	446	2946
	20	11,6	4,0	7,5	50	60	724	2240
3	1	44,1	13,2	30,2	482	63	481	3453
	20	7,6	3,3	4,1	54	54	746	2291
4	1	45,7	9,5	35,6	403	51	402	2743
	20	12,6	3,6	8,8	69	49	800	2502
5	1	78	18,0	58,7	542	69	541	4839
	20	9,6	3,6	5,8	52	50	743	2189
6	1	46,0	15,7	29,4	501	65	500	3365
	20	9,7	5,0	4,3	73	48	902	2638
7	1	89	17,0	71	525	62	524	4368
	20	16,6	4,8	11,6	59	61	911	3049
8	1	108	13,5	94	446	42	445	3334
	20	22,5	3,7	18,6	58	55	701	2194
9	1	45,6	9,5	35,3	423	55	422	2630
	20	9,1	3,4	5,6	49	55	708	2117
10	1	395	16,4	378	501	62	500	4680
	20	72	4,3	68	62	56	790	2646
11	1	171	13,0	157	464	48	463	3257
	20	23,7	3,1	20,5	54	44	693	1858

introduite, alors que la deuxième correspond à la version de base, c'est-à-dire qu'on y insère toutes les colonnes (parmi les 20 possibles) de coût réduit négatif. On retrouve d'ailleurs en deuxième colonne les chiffres "1" et "20" pour identifier ces deux lignes. La section "temps de calcul" donne le temps de calcul total, le temps de calcul total consacré à CPLEX, et le temps de calcul total consacré aux problèmes auxiliaires. Les autres résultats sont le nombre total d'itérations de génération de colonnes, la fraction de ces itérations qui sont utiles, le nombre total de colonnes générées et le nombre total de pivots CPLEX.

Le fait d'insérer toutes les colonnes de coût réduit négatif (parmi les 20 possibles) favorise une convergence nettement plus rapide, comme le nombre d'itérations

l'indique. Ceci se traduit par un temps de calcul total plus petit. On note cependant que la fraction d'itérations utiles n'est pas nécessairement meilleure. En fait, on observe généralement une diminution de cette fraction.

Il va sans dire que cette illustration pourrait être généralisée à toutes les classes de problèmes-tests.

4.6 Effet du critère d'arrêt

Nous allons observer ici l'effet de l'application du critère d'arrêt basé sur le calcul de la borne inférieure (cf. section 3.3.1). Pour ce faire, on compare la version de base à une version dans laquelle ce critère d'arrêt est utilisé. Pour vérifier l'effet de la dominance, il est certain que nous devons utiliser des problèmes-tests qui branchent. Cependant, il est possible dans ce cas que l'effet de la règle de division intervienne dans les résultats, puisqu'il n'est pas exclu que des secteurs différents soient choisis à la fin de la génération de colonnes, dans la règle de division (c'est ce que nous avons observé). Il faut donc prendre soin de comparer les résultats pour des problèmes qui ont progressé de la même manière. C'est ce que nous avons fait. Sur les 20 problèmes-tests de la classe $[n = 20; m = 50]$, nous n'avons donc conservé que ceux qui ont progressé de la même façon dans les deux versions et sur lesquels un effet (positif ou négatif) a été effectivement observé. Ceci nous donne 12 problèmes, dont les résultats sont comparés au tableau 4.11 ci-dessous.

La première colonne donne le numéro du problème-test. Pour chaque problème, on montre dans ce tableau deux lignes de résultats. La première correspond à la version de base et la deuxième à la version modifiée (nouveau critère d'arrêt avec dominance). Ces deux lignes sont identifiées en deuxième colonne par les lettres

Tableau 4.11 Comparaison des critères d'arrêt

		temps de calcul (s)	it.	col.	pivots
1	B	25.8	139	1134	4152
	N	26.5	129	1126	3908
2	B	10.8	108	1037	2648
	N	12.1	106	1037	2644
3	B	19.6	111	925	3202
	N	17.9	91	921	3361
4	B	9.7	123	978	2937
	N	8.7	110	951	2716
5	B	9.0	91	1011	2602
	N	8.6	89	1000	2712
6	B	5.6	54	653	1681
	N	5.8	53	652	1680
7	B	5.8	69	786	2496
	N	5.9	71	815	2251
8	B	6.4	63	638	1597
	N	6.5	59	632	1589
9	B	8.9	151	1156	3077
	N	10.0	155	1251	3733
10	B	7.8	87	847	2697
	N	7.2	71	825	2615
11	B	8.4	60	693	1894
	N	7.7	57	687	1879
12	B	3.9	43	572	1196
	N	3.6	41	570	1194

"B" et "N", pour "Base" et "Nouvelle" si l'on veut. Les résultats sont: le temps de calcul total, le nombre total d'itérations de génération de colonnes, le nombre total de colonnes générées et le nombre total de pivots CPLEX.

On observe que la version modifiée (ou nouvelle) entraîne en général (pas nécessairement) une économie d'itérations et de pivots CPLEX. On note d'ailleurs qu'une amélioration au niveau des itérations ne s'accompagne pas toujours de l'amélioration du nombre de pivots (et inversement). Cependant, on constate que la diminution du nombre total d'itérations ne se répercute pas, en général sur le temps de calcul. En effet, le calcul de la borne inférieure exige plus de temps dans

les problèmes auxiliaires, que la diminution des itérations ne suffit pas toujours à compenser. Il faut donc conclure, sur la base de ces essais, qu'à toutes fins pratiques les deux versions sont équivalentes.

Nous voulons par ailleurs faire remarquer que ceci ne disqualifie en rien le point de vue que nous avons adopté à la section 3.3 concernant la génération de colonnes et la dualité. Il n'est pas indifférent que d'autres enseignements, utiles ou non, puissent être tirés de ce point de vue.

CHAPITRE 5
Résultats supplémentaires

Les résultats présentés ici ont pour but de compléter ceux du chapitre précédent. Nous voulons illustrer l'effet de l'augmentation du coût d'utilisation des sites, de l'augmentation du nombre de sites potentiels et de l'augmentation de la taille des problèmes sur le temps de calcul et autres indicateurs. Notre but n'est pas d'être exhaustif, mais plutôt d'observer des effets spécifiques. Nous apportons aussi, à la lumière de certains de ces résultats, une discussion finale sur les problèmes-tests utilisés. Tous les résultats de ce chapitre ont été obtenus avec la version de base de l'algorithme. Les problèmes-tests ont été générés conformément à la méthode décrite au chapitre 4.

5.1 Effet de l'augmentation des coûts d'ouverture

Nous avons déjà mentionné ailleurs l'impact du coût d'ouverture des sites sur la difficulté du problème et donc sur les performances de l'algorithme. Il est intuitivement évident qu'un coût d'ouverture élevé demande une plus grande attention au niveau de la sélection des sites. Le problème consiste en effet à trouver un bon compromis (de coût minimum) entre le coût total des sites et le coût total de transport: l'ouverture d'un grand nombre de sites permet de réduire le coût de transport, mais cette réduction doit être confrontée à l'augmentation des coûts associés à l'utilisation des sites. L'algorithme ouvre donc autant de sites que nécessaire pour réduire le coût global. Il est clair que plus le coût d'ouverture (ou d'utilisation) des sites est grand, moins le nombre de sites ouverts sera élevé.

Notre but est d'illustrer et surtout de quantifier l'effet de ce coût d'utilisation des sites sur les performances de l'algorithmes et sur le nombre de sites ouverts. Pour ce faire, on a choisi de tester 3 classes de problèmes-tests ($[n = 15; m = 60]$, $[n = 20; m = 60]$ et $[n = 25; m = 60]$), chacune comprenant 10 problèmes. On a défini 3 ordres de grandeur pour le coût d'ouverture des sites et on a utilisé les mêmes problèmes pour tous les ordres de grandeur (seule la position du point décimal dans le coût d'utilisation est changée). La formule (4.1) permet de produire, rappelons-le, des coûts d'utilisation dont l'ordre de grandeur se situe dans l'intervalle [10e4, 10e5]. C'est l'ordre de grandeur qui a été utilisé jusqu'ici.

Le tableau 5.1 ci-dessous montre les résultats obtenus pour les 3 ordres de grandeur du coût d'utilisation des sites, pour les trois classes de problèmes.

Tableau 5.1 Caractéristiques selon l'ordre de grandeur du coût d'utilisation

ordre de grandeur	n	m	temps total (s)	noeuds	prof.	it.	pivots	saut (%)	sites
[10e3, 10e4]	15	60	74	5,2	1,4	115	4380	0,03	14,8
	20		18,9	2,6	0,7	66	2632	0,02	18,7
	25		7,4	1,2	0,1	51	2162	0,01	22,2
[10e4, 10e5]	15	60	181	8,8	2,8	165	5413	0,06	14,5
	20		30,1	8,0	1,8	119	3792	0,03	17,8
	25		12,2	11,6	2,8	114	3075	0,12	20,4
[10e5, 10e6]	15	60	3266	55	7,5	756	27299	0,18	13,3
	20		1961	100	8,9	1175	37717	0,29	14,4
	25		801	307	14,4	2682	64631	0,42	14,8

Toutes les valeurs de ce tableau sont des moyennes pour les 10 problèmes de chaque classe. On y donne le temps de calcul total, le nombre de noeuds dans l'arbre de recherche, sa profondeur, le nombre total d'itérations de génération de colonnes, le nombre total de pivots CPLEX, le saut d'intégralité et le nombre de sites utilisés.

On observe une augmentation de toutes les valeurs, sauf bien sûr pour le nombre de sites utilisés. Cette augmentation est particulièrement marquée entre les deux derniers ordres de grandeur du tableau. On voit que l'utilisation d'un coût d'ouverture 10 fois plus élevé que celui que nous avons choisi pour les problèmes-tests générés au chapitre 4 a un impact considérable sur les mesures de performance (et provoque une augmentation des cas de cyclage). Nous avons observé que le temps de calcul par itération devient assez grand pour la classe $[n = 15;\ m = 60]$, alors que la classe $[n = 25;\ m = 60]$ est surtout affectée par la profondeur accrue des arbres de recherche.

Comme on pouvait s'y attendre, la diminition du nombre desites utilisés est moins significative lorsque le nombre n de sites potentiels est faible par rapport au nombre de secteurs m.

On revient au coût d'ouverture habituel dans les problèmes-tests qui suivent.

5.2 Effet de l'augmentation du nombre de sites potentiels

Nous voulons voir ici comment se comporte l'algorithme lorsque nous augmentons le nombre n de sites disponibles. Pour ce faire, on utilise 12 classes de problèmes-tests, chacune comportant 10 problèmes (sauf pour les deux premières classes, où les 20 problèmes définis au chapitre 4 ont été utilisés), pour lesquels m est fixé à 60 alors que n varie de 15 à 500. Il est évidemment impossible qu'un si grand nombre de sites potentiels puissent être utilisés en pratique. Cette constatation n'enlève rien à la validité de l'expérience.

Le tableau 5.2 ci-dessous montre, pour chaque classe de problèmes, le temps de calcul total moyen (colonne (1)), le nombre moyen de sites utilisés dans la solution

optimale (colonne (2)) et, en troisième colonne, le nombre relatif de sites utilisés.

Tableau 5.2 Effet de l'augmentation du nombre de sites potentiels

n	m	(1) temps total (s)	(2) nombre de sites utilisés	(2)/n (%)
15		42,5	14,0	93
20		22,8	18,2	91
30		8,0	23,0	77
40		7,1	24,9	62
50		7,3	26,3	53
60		5,3	28,8	48
70	60	5,5	30,3	43
80		5,5	31,9	40
.		.	.	.
200		5,3	39,6	20
300		5,3	42,1	14
400		6,5	42,4	11
500		8,9	45,4	9

Lorsque le nombre de sites potentiels n augmente, le nombre de sites utilisés augmente, mais avec un taux de croissance de plus en plus petit. Il est probable que le nombre de sites utilisés se stabilise aux alentours de 45-50 sites, lorsque n est très grand. En effet, tant que l'ouverture de nouveaux sites permet d'abaisser le coût global, l'algorithme le fait. Lorsqu'il n'est plus possible de faire baisser suffisamment le coût de transport de manière à compenser le coût d'ouverture des sites supplémentaires, le nombre de sites utilisés se stabilise.

Par ailleurs, plus le nombre de sites ouverts est grand, plus l'affectation des secteurs est facile. On note par conséquent que le temps de calcul diminue lorsque n augmente, pour se stabiliser à une valeur de 5 à 6 secondes, avant de progresser lentement pour un nombre de sites potentiels très élevé. Cette lente progression pourrait peut-être s'expliquer par le fait que n problèmes auxiliaires sont résolus

à chaque itération. Lorsque n est très grand, la plupart des problèmes-tests sont résolus au premier noeud en un très faible nombre d'itérations; c'est le temps de calcul par itération qui alors devient plus élevé, en raison du grand nombre de problèmes auxiliaires (la part consacrée aux problèmes auxiliaires augmente).

En théorie, l'aspect combinatoire de la localisation devrait tendre à faire augmenter la difficulté du problème (donc le temps de calcul) lorsque le nombre de sites potentiels augmente. Il ne semble donc pas que cet effet soit déterminant ici.

5.3 Effet de l'augmentation de la taille des problèmes

Nous voulons illustrer ici l'effet de l'augmentation de la "taille" des problèmes sur le temps de calcul. Nous avons déjà observé au chapitre 4 l'effet de l'augmentation (pour un nombre de sites potentiels n donné) du nombre de secteurs m. Dans ce cas, plus m est grand, plus l'affectation des secteurs est difficile, et plus le temps de calcul est élevé. À la section 5.2, nous avons montré l'effet de l'augmentation (pour m donné) du nombre de sites potentiels n.

Notre but est ici de faire varier à la fois n et m tout en maintenant un rapport m/n constant (d'une valeur de 2,5). Pour 7 classes de problèmes, chacune comportant 10 problèmes-tests (sauf pour la classe $[n = 20;\ m = 50]$ dont les problèmes sont ceux du chapitre 4), on trouve au tableau 5.3 ci-dessous le temps de calcul total moyen (colonne (1)), le nombre total moyen d'itérations (colonne (2)) et le rapport de ces deux colonnes, tel qu'indiqué.

On remarque que le temps de calcul subit une nette progression (de type exponentiel) en fonction de n et m. Il semble en être de même pour les valeurs de la dernière colonne. Celle-ci indique d'ailleurs un temps de calcul par itération

Tableau 5.3 Effet de l'augmentation de la taille des problèmes

n	m	(1) temps total (s)	(2) nombre total d'itérations	(1)/(2) (s)
20	50	9,2	85	0,1
30	75	49,1	167	0,3
40	100	63	154	0,4
50	125	205	169	1,2
60	150	501	194	2,6
70	175	1356	382	3,5
80	200	3918	551	7,1

considérable pour la dernière classe testée. Nous avons d'autre part observé que plus les problèmes deviennent difficiles, plus ils ont de chance d'être affectés par le cyclage.

5.4 Retour sur les problèmes-tests

Un des aspects les plus problématiques (et les plus discutables) dans la définition des problèmes-tests est le coût d'utilisation des sites. Seule une estimation réaliste de ces coûts pourrait clore la question. À notre avis cependant, le coût d'ouverture d'un site représente probablement une fraction du coût de transport total de la neige à ce site (si l'on exclut les coût environnementaux).

Remarque: Nous avons déjà mentionné que, d'après la formule (4.1), le coût d'ouverture d'un site est *a priori* approximativement 100 fois moindre que le coût total de transport de la neige à ce site (cf. section 4.2). Après vérification, nous avons constaté que ce coût était plutôt, dans les problèmes que nous avons testés au chapitre 4, approximativement 10 fois moindre que le coût de transport à un site. En fait, la formule (4.1) surévalue, en général, la quantité de neige qui est acheminée

à un site.

Quoiqu'il en soit, nous sommes maintenant en mesure d'apprécier ce qui se serait passé si nous avions utilisé des coûts d'utilisation plus élevés (par exemple, multipliés par 10). L'impact le plus significatif est bien sûr un temps de calcul nettement plus grand, comme nous l'avons montré à la section 5.1. La profondeur des arbres de recherche et le nombre de noeuds seraient plus élevés. Le nombre de sites utilisés aurait été réduit. Par ailleurs, les cas de cyclage auraient été plus nombreux. L'algorithme exigerait donc certainement des ajustements de manière à le rendre plus performant (cf. section 4.4.3)

Toutes les expériences révèlent une variation importante du temps de calcul selon le rapport entre le nombre de sites potentiels n et le nombre de secteurs m. Plus ce rapport est grand, plus les problèmes sont faciles. Ceci nous amène à un autre aspect délicat dans la définition des problèmes-tests: le nombre de sites potentiels. Nous avons utilisé, pour certaines classes de problèmes-tests du chapitre 4, un nombre de sites potentiels assez faible compte tenu du nombre de secteurs. Un nombre de sites potentiels plus élevé nous aurait donc permis d'utiliser des coûts d'ouverture plus grands. Il est certain que la determination du nombre de sites potentiels, ainsi que leur localisation (autrement dit la détermination de la liste de sites potentiels nécessaire à notre problème), est en soi une problématique entièrement nouvelle. Il en est de même pour le découpage d'un centre urbain en secteurs.

CONCLUSION

Nous avons présenté au premier chapitre le Problème de Localisation-Affectation avec Contraintes de Capacités (PLACC) dans un contexte de déneigement urbain. Campbell et Langevin [3] se sont déjà intéressés à ce problème. Ils ont proposé un modèle mathématique pour le résoudre, que nous avons appelé (M1). Ils n'ont cependant pas développé de méthode de résolution exacte. Notre objectif était donc de résoudre PLACC de façon exacte. Pour ce faire, nous avons présenté un modèle de génération de colonnes (M2), équivalent à (M1). Nous avons proposé au chapitre 2 une décomposition de ce modèle pour la résolution par génération de colonnes, obtenant ainsi un problème maître (PM) et des problèmes auxiliaires $(PA)_j$.

La méthode de résolution combine la technique de génération de colonnes et un algorithme de séparation et évaluation. Elle a été décrite en détails au chapitre 3. Le problème auxiliaire est un problème de "sac-à-dos" bidimensionnel avec des contraintes additionnelles résultant de la règle de division. L'algorithme utilisé pour le résoudre est également décrit au même chapitre.

Cette méthode de résolution a été testée sur un ensemble de 11 classes de problèmes-tests. Nous avons présenté et analysé les résultats au chapitre 4. Des résultats supplémentaires, présentés au chapitre 5, ont permis d'observer certains effets particuliers. Il semble que la méthode de résolution utilisée soit validée par nos résultats. Bien sûr, des améliorations sont souhaitables, en particulier au niveau de la dégénérescence et de l'arbre d'exploration, et elles pourront certainement favoriser l'attaque de problèmes plus difficiles. Nous avons proposé à ce sujet quelques

pistes de réflexion.

Nous croyons que ce travail peut contribuer à une réduction des coûts dans le domaine du déneigement urbain. Par ailleurs, nous rappelons que le problème auquel nous nous sommes intéressés ici ne représente qu'un des nombreux problèmes que l'on rencontre dans la planification d'un système complet de déneigement. C'est un vaste terrain pour la recherche opérationnelle.

BIBLIOGRAPHIE

[1] BARNHART C., JOHNSON E.L., NEMHAUSER G.L., SAVELSBERGH M.W.P., VANCE P.H. (1994), Branch-and-price: column generation for solving huge integer programs, in *Mathematical Programming: State of the Art 1994*, Birge J.R., Murty K.G. (éds.), University of Michigan, USA

[2] CAMPBELL J.F., LANGEVIN A. (1995), The snow disposal assignment problem, *Journal of the Operational Research Society 46*, pages 919-929

[3] CAMPBELL J.F., LANGEVIN A. (1995), Operations management for urban snow removal and disposal, *Transportation Research 29A*, pages 359-370

[4] DESROSIERS J., SOUMIS F., DESROCHERS M. (1984), Routing with time windows by column generation, *Networks 14*, pages 545-565

[5] DU MERLE O., VILLENEUVE D., DESROSIERS J., HANSEN P. (1997), Stabilisation dans le cadre de la génération de colonnes, *Les Cahiers du GERAD G-97-08*, Montréal, Québec

[6] FRÉVILLE A., PLATEAU G. (1992), FPBK92: An implicit enumaration code for the solution of the 0-1 bidimensional knapsack problem based on surrogate duality, *Graphs and Optimization Colloquium*, Grimentz, Suisse

[7] FRÉVILLE A., PLATEAU G. (1993), An exact search for the solution of the surrogate dual of the 0-1 bidimensional knapsack problem, *European Journal of Operational Research 68*, pages 413-421

[8] GAVISH B., PIRKUL H. (1985), Efficient algorithms for solving multiconstraint zero-one knapsack problems to optimality, *Mathematical Programming 31*, pages 78-105

[9] GAVISH B., PIRKUL H. (1991), Algorithms for the multi-resource generalized assignment problem, *Management Science 37*, pages 695-713

[10] GILMORE P.C., GOMORY R.E. (1961), A linear programming approach to the cutting stock problem, *Operations Research 9*, pages 849-859

[11] HANSEN P., JAUMARD B., POGGI DE ARAGÃO M. (1991), Un algorithme primal de programmation linéaire généralisée pour les programmes mixtes, *Comptes-rendus de l'Académie des Sciences 313*, pages 557-560

[12] JAUMARD B., GOURDIN É. (1992), Techniques informatiques pour la recherche opérationnelle, *Les Cahiers du GERAD G-92-43*, Montréal, Québec

[13] LECLERC G. (1985), Least cost allocation of snow zones to elimination sites: formulation and post-optimal analysis, *Civil Engineering Systems 2*, pages 217-222

[14] MARTELLO S., TOTH P. (1990), *Knapsack problems: algorithms and computer implementations*, Wiley, New York

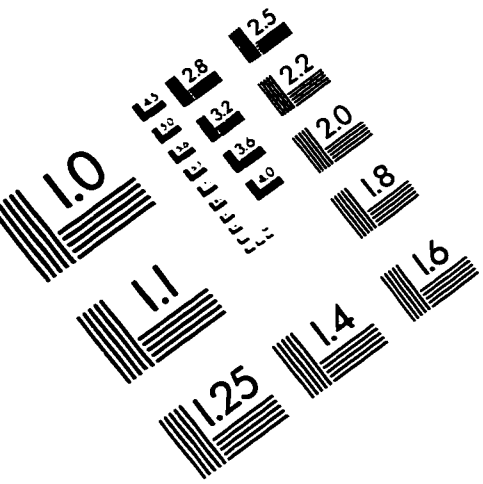
[15] MINOUX M. (1983), *Programmation mathématique, théorie et algorithmes, tome 2*, Dunod, Paris

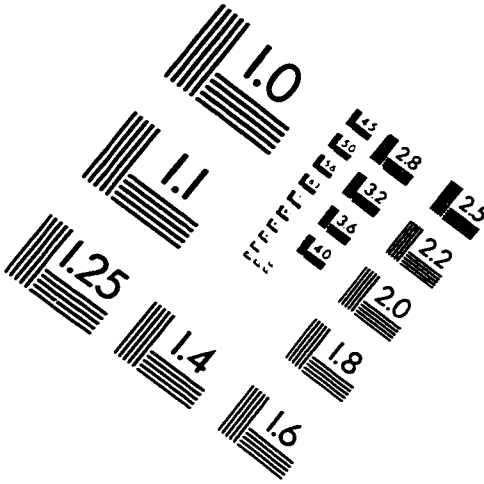
[16] MLADENOVIĆ N., HANSEN P. (1996), Variable neighbourhood search, *Les Cahiers du GERAD G-96-49*, Montréal, Québec

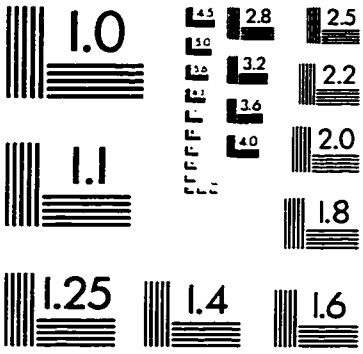
[17] RYAN D.M., FOSTER B.A. (1981), An integer programming approach to scheduling, in *Computer Scheduling of Public Transport, Urban Passenger Vehicle and Crew Scheduling*, éd. Wren, A., North-Holland, Amsterdam

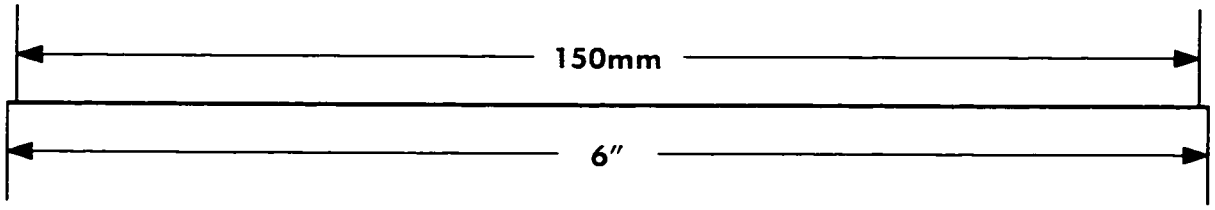
[18] VANDERBECK F., WOLSEY L.A. (1994), An exact algorithm for IP column generation, *Operations Research Letters 19*, pages 151-159

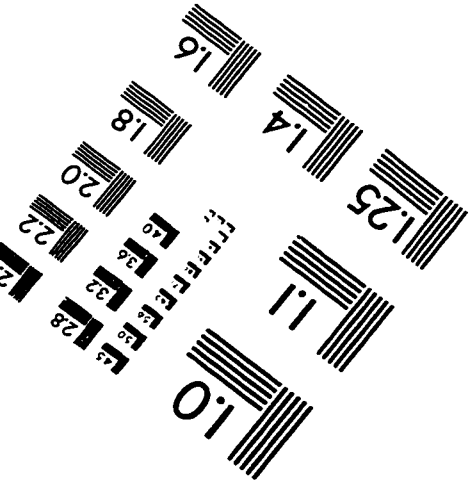
IMAGE EVALUATION
TEST TARGET (QA-3)

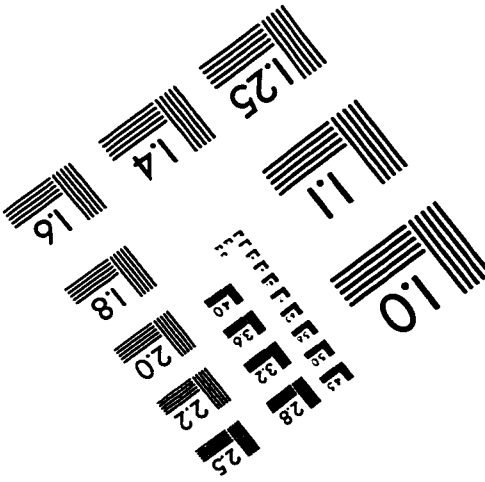












APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989