Titre: Title:	Comparaison de critères de branchement pour le problème de recouvrement
Auteur: Author:	Richard Thibault
Date:	1989
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Thibault, R. (1989). Comparaison de critères de branchement pour le problème de recouvrement [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/58286/

UNIVERSITÉ DE MONTRÉAL

COMPARAISON DE CRITÈRES DE BRANCHEMENT POUR LE PROBLÈME DE RECOUVREMENT

par

Richard THIBAULT
DÉPARTEMENT DE MATHÉMATIQUES APPLIQUÉES
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU GRADE DE MAITRE ES SCIENCE APPLIQUEES (M.Sc.A.)
(MATHÉMATIQUES APPLIQUÉES)
Septembre 1989

ISBN 0-315-58200-6

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE

Ce mémoire intitulé:

COMPARAISON DE CRITÈRES DE BRANCHEMENT POUR LE PROBLÈME DE RECOUVREMENT

présenté par: Richard Thibault

en vue de l'obtention du grade de: MAITRE ES SCIENCES APPLIQUEES (M.Sc.A)

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

Mme. Godelieve Vanderstraeten, Ph.D., président

M. Jacques Desrosiers, Ph.D.

M. François Soumis, Ph.D.

M. Martin Desrochers, Ph.D.

À Michelle,

Sommaire

Ce mémoire porte sur l'étude d'une méthode de résolution du problème de recouvrement. Un problème de recouvrement est un programme en nombres entiers dont l'objectif est de minimiser une fonction de coût linéaire, et dont la matrice des contraintes est de type binaire, c'est-à-dire composée de 0 ou 1. Chacune des contraintes est de la forme ≥ 1. Les variables de décision de ce problème sont binaires, et notons que celui-ci fait partie de la classe de problèmes NP-complets. On l'utilise pour modéliser un grand nombre de problèmes de tournées et d'horaires. On résout habituellement ce problème à l'aide de méthodes d'énumération implicite.

Dans ce mémoire, nous étudions l'efficacité d'une procédure de branchement, basée sur l'élimination progressive d'une classe de sous-matrices indésirables appelées sous-matrices cycliques, contenues dans la matrice des contraintes. Un groupe d'entre elles produisent un écart entre les bornes inférieure et supérieure.

Nous avons tout d'abord construit une procédure d'énumération implicite utilisant des bornes obtenues à l'aide d'heuristiques duales (méthode de sous-gradient) et primales efficaces.

Puis, nous avons fait appel à plusieurs propriétés, portant principalement sur les matrices balancées, afin de construire nos différents critères de branchement et choisir le mode de sélection des sous-problèmes.

Enfin, ces différents critères sont expérimentés sur une série de problèmes générés aléatoirement, et les résultats de calcul sont présentés. À partir de ces résultats, de nouveaux critères seront formés et comparés.

Abstract

This thesis addresses the solution method of the set covering problem. The set covering problem seeks to minimize a linear cost function, The constraint matrix is composed of 0,1 elements, and each of the constraints is of the form ≥ 1. The decision variables of this problem are from binary type, and it is included in the NP-Complet class. The set covering problem is used in many routing and scheduling problems. This problem is usually solved using branch and bound methods.

In this thesis, we study the efficiency of a branching rule based on the progressive elimination of some classes of forbidden submatrices included in the constraint matrix. Some of these submatrices induce a gap between the lower bound and the upper bound.

We first developed a branch and bound method, which use efficient dual (subgradient method) and primal heuristics.

We then used various branching criteria and subproblem selection rules, principally based on balanced matrices properties.

Finally, these criteria were tested on randomly generated problems and the

computational results are presented. Some new criteria, based on the results, will be developed and compared.

Remerciements

J'aimerais d'abord témoigner toute ma gratitude à mes directeurs de recherche, François Soumis, Martin Desrochers et Brigitte Jaumard. M. Soumis, par ses encouragements et ses suggestions, m'a fait découvrir et aimer le monde de la recherche. Il est impossible d'évaluer toute l'aide que m'a apportée M. Desrochers dans ce projet, j'ai appris beaucoup en travaillant avec lui. Mme Jaumard m'a fait profiter de ses connaissances dans le domaine de la programmation en nombres entiers, et en informatique.

J'aimerais aussi remercier l'Ecole Polytechnique pour le soutien financier qu'elle m'a accordé, me permettant de consacrer tout mon temps à ce projet.

Je remercie également mon amie Michelle, ainsi que mes parents, pour leur appui et leurs encouragements tout au long de ce projet.

Enfin, pour le remarquable travail de dactylographie, je tiens à remercier Mme Anne-Marie Goulet.

Table des Matières

Figures

Tableaux

Chapitre 1

Introduction

Plusieurs chercheurs se sont penchés sur le problème de recouvrement ces derniers temps, celui-ci étant utilisé dans de nombreuses applications (fabrication de tournées de véhicules, d'horaires d'employés, de localisation de commodités). On résout habituellement ce problème par des méthodes d'énumération implicite (branch and bound). Les recherches furent principalement orientées vers des procédures efficaces permettant d'obtenir des bornes inférieures et des bornes supérieures, qui sont deux composantes importantes de la méthode de résolution.

Le choix du critère de branchement influence également l'efficacité de la méthode de résolution. Le branchement vise à diminuer l'écart entre les bornes inférieures et supérieures. Cet écart est créé par des sous-matrices de type cyclique, et nos critères de branchement sont basés sur l'élimination de ce type de sous-matrices. Des propriétés portant sur les matrices balancées sont utilisées afin de développer des conditions nécessaires sur la présence des sous-matrices cycliques, et pour construire diverses stratégies.

L'objectif principal de ce mémoire est d'évaluer les performances de cette approche, en utilisant des heuristiques efficaces et connues pour calculer les bornes inférieure et supérieure. L'expérimentation effectuée sur une série de problèmes de densité réaliste. Nous désirons également améliorer la structure de branchement à partir des résultats obtenus.

Au chapitre 2, nous allons tout d'abord décrire la structure du problème de recouvrement, et présenter des applications importantes dans lesquelles il intervient. Puis, on consacre le chapitre 3 à une revue de la littérature portant sur les composantes importantes des méthodes d'énumération implicite. Notre méthode de résolution, ainsi que les différentes procédures qui la composent, sont

ensuite présentées au chapitre 4. Une étude de propriétés reliées à la présence des sous-matrices cycliques, suivie du développement de différents critères sur ces propriétés, fait l'objet du chapitre 5. Enfin, les résultats et analyses de la résolution se retrouvent au chapitre 6.

Chapitre 2

Description et applications

La première partie de ce chapitre est consacrée à la présentation et la description du modèle mathématique du **problème de recouvrement** (PR). Dans la seconde partie, trois applications importantes, dans lesquelles la formulation PR est utilisée, seront exposées. L'objectif principal de ce chapitre, est de justifier l'intérêt d'une recherche sur le PR, considérant les nombreuses situations pratiques utilisant ce modèle.

2.1 Problème de recouvrement

2.1.1 Notations

Afin de faciliter la compréhension du modèle, on définit les paramètres suivants:

n	:Nombre de variables (colonnes).
m	:Nombre de contraintes (lignes).
$N = \{j : j = 1, \cdots, n\}$	:Ensemble des indices des variables.
$M = \{i : i = 1, \cdots, m\}$	:Ensemble des indices des lignes.
$x_j, j \in N$	:Variable binaire valant 1 si la colonne j est choisie et 0 sinon.
$c_j, j \in N$	:Coût associé à la variable j.
$a_{ij}, i \in M, j \in N$	:Constante binaire valant 1 si la colonne j recouvre la ligne i et 0 sinon.
$a^j, j \in N$	:Vecteur des éléments a_{ij} sur la colonne j.

$a_i, i \in M$:Vecteur des éléments a_{ij} sur la ligne i.

Énoncé

Nous disposons d'un ensemble de n colonnes. Chacune d'elles recouvre un ensemble de lignes (correspondant aux composantes non-nulles de a_j), et un coût d'une valeur c_j lui est associé. On désire trouver un sous-ensemble N^* de N, tel que chacune des lignes i de M sera recouverte par au moins une des colonnes j de N^* et ce, à coût minimum.

La formulation mathématique est présentée ci-dessous:

$$\text{MINIMISER} \quad Z_c = \sum_{j \in N} c_j x_j \tag{2.1}$$

Sous les contraintes:

$$\sum_{j \in N} a_{ij} x_j \geq 1 \quad , i \in M \tag{2.2}$$

$$x_j \geq 0 \quad , j \in N \tag{2.3}$$

$$x_j \text{ binaire} \quad , j \in N \tag{2.4}$$

Le PR est donc un problème **linéaire en nombres entiers** (PLE) et fait partie de la classe des problèmes **NP-complets.**

On présente souvent le PR comme étant une relaxation d'un problème bien connu de la même famille appelé **problème de partitionnement** (PP). La différence entre le PP et le PR réside au niveau des contraintes (2.2) qui sont remplacées par les contraintes suivantes dans le PP:

$$\sum_{j \in N} a_{ij} x_j = 1 \quad , i \in M \tag{2.5}$$

Le sur-recouvrement des lignes n'est pas accepté dans le PP, ce qui rend la recherche d'une solution entière réalisable non-triviale. C'est pourquoi on relaxe la contrainte (2.5) dans de nombreuses applications utilisant la formulation PP, de façon à obtenir un PR. Ce changement a généralement de nombreux avantages:

1. La formulation PR permet d'obtenir plus facilement une solution réalisable, i.e. une borne supérieure sur la valeur de la solution optimale.

2. La valeur de la solution optimale du PR est une borne inférieure sur celle du PP.

3. Dans plusieurs problèmes pratiques utilisant le modèle PP, on désire que chaque ligne (tâche à exécuter) soit recouverte. Le sous-recouvrement n'est pas accepté, mais le sur-recouvrement n'est généralement pas problématique.

4. A partir de la solution du PR, il est parfois possible (en pratique) d'annuler les éléments (a_{ij}) non-nuls d'une colonne a^j qui provoquent le sur-recouvrement d'une ligne sans toutefois détériorer l'objectif. On peut donc obtenir une solution valide pour le PP qui a la même valeur que la solution du PR ou dans le pire cas, une solution légèrement plus coûteuse.

2.2 Applications

2.2.1 Fabrication de tournées de véhicules

De nombreuses variantes du problème de fabrication de tournées de véhicules (PFTV) ont été étudiées. Elles se distinguent par les contraintes considérées. Nous présentons la variante du PFTV dans lequelle des limites de capacité sont imposées [2, Balinski et Quandt].

Description

Une compagnie, possédant une flotte de véhicules localisée à **un dépot**, doit désservir des clients géographiquement dispersés dans une région spécifique. Tous les véhicules ont la même **capacité**, et chaque client a une **demande** (commande) qui doit être satisfaite. On appelle **tournée**, un chemin débutant et se terminant au dépot, en visitant certains clients (figure 2.1). Le **coût d'une tournée** est une fonction de plusieurs facteurs, comme la charge des demandes de ses clients ou la distance parcourue entre le départ et le retour au dépot. Un **coût fixe** associé à l'utilisation d'un véhicule peut également être ajouté.

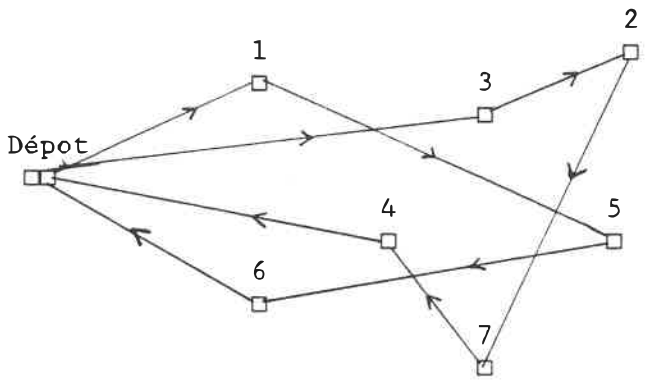

Figure 2.1: Tournées de véhicule réalisable pour un problème de 7 clients.

L'objectif de ce problème est de trouver un ensemble de tournées de véhicules, de coût total minimum, pouvant servir tous les clients en respectant les contraintes de capacité.

Formulation PR

Soit R, l'ensemble de toutes les tournées réalisables du PFTV. La valeur Ω_r correspond au coût de la tournée $r \in R$. Une constante binaire δ_{ri} est associé aux clients de chaque tournée, c'est-à-dire que δ_{ri} vaut 1 si la tournée r visite le client i et 0 sinon. Enfin, une variable booléenne x_r est utilisée pour identifier les tournées r choisies ($x_r = 1$) ou celles éliminées ($x_r = 0$).

Tous les clients $i (i \in N)$ doivent être servis par le sous-ensemble de tournées sélectionnées, et la formulation PR est comme suit:

$$\text{MINIMISER} \quad \sum_{r \in R} \Omega_r x_r \tag{2.6}$$

Sous les contraintes:

$$\sum_{r \in R} \delta_{ri} x_r \geq 1 \quad , i \in N \tag{2.7}$$

$$x_r \in \{0,1\} \quad , r \in R \tag{2.8}$$

Dans la plupart des problèmes de fabrication de tournées de véhicule, l'ensemble R a une cardinalité trop grande pour être traité explicitement, c'est pourquoi on ne considère souvent qu'un sous-ensemble K de tournées intéressantes. On obtient généralement K à l'aide d'un générateur de tournées, comme par exemple une procédure de **génération de colonnes**.

2.2.2 Fabrication d'horaires de conducteurs d'autobus

Le problème de fabrication d'horaires des conducteurs d'autobus urbains (PFHCAU) représente la dernière phase dans la planification d'un réseau de transports en commun. La société de transport municipal doit tout d'abord construire un ensemble de **tournées** d'autobus pour répondre aux besoins des usagers, et y affecter les conducteurs par la suite.

Étant donné le nombre de municipalités à travers le monde dotées d'un réseau de transports en commun, et l'importance des coûts investis dans ce service, le PFHCAU constitue une application très importante utilisant la formulation PR.

Description

Afin de bien illustrer le PFHCAU, nous présentons celui auquel est confrontée la société de transport de la communauté urbaine de Montréal [7, Desrochers et Soumis].

Chaque tournée d'autobus débute et se termine au garage, en passant par des **points de relève**, qui sont les endroits où un conducteur peut être remplacé par un autre. L'heure à laquelle un autobus arrive à un point de relève est appelé **heure de relève**. On nomme **tâche**, un intervalle de temps compris entre deux points de relève consécutifs sur la même tournée. La **journée de travail** d'un conducteur consiste en une ou plusieurs **pièces de travail** contenant chacune une ou plusieurs tâches consécutives sur un même autobus (figure 1.2). Chaque journée de travail doit respecter les conditions de faisabilité et être payée selon le traitement salarial défini dans la **convention collective.**

L'objectif de la société de transport est de fabriquer un horaire de travail **légal** de coût (salaires versés) minimum, qui recouvre toutes les tâches.

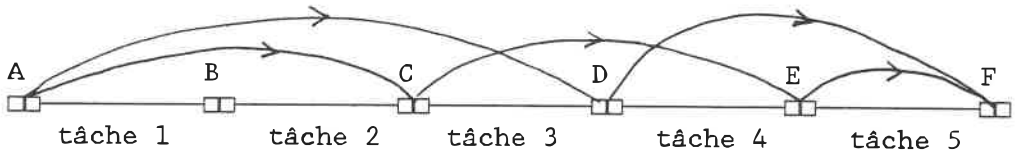

Figure 2.2: Tournée constituée de 5 tâches délimitées par le départ du garage (A), des points de relève (B-E) et le retour au garage (F). Les pièces de travail réalisables sont représentées par les arcs.

Formulation PR

Afin de formuler le PFHCAU par un PR, on doit tout d'abord faire appel à un **générateur** de journées de travail. Il s'agit donc, dans un premier temps, de générer l'ensemble des journées de travail légales et de choisir par la suite un sous-ensemble contenant les meilleurs candidats qui recouvre toutes les tâches.

La formulation du PR est alors:

$$\text{MINIMISER} \quad \sum_{j \in N} c_j x_j \tag{2.9}$$

Sous les contraintes:

$$\sum_{j \in N} a_{ij} x_j \geq 1 \quad , i \in M \tag{2.10}$$

$$x_j \in \{0,1\} \quad , j \in N \tag{2.11}$$

où,

M	:Ensemble des tâches.
N	:Ensemble des journées de travail.
$a_{ij}, i \in M, j \in N$	:Constante binaire égale à 1 si la journée de travail j contient la tâche i , 0 sinon.
$x_j, j \in N$	:Variable binaire qui vaut 1 si la journée de travail j est choisie, 0 sinon.

Notons que des contraintes peuvent être éliminées du PR. En effet, si un point de relève n'est utilisé dans aucune pièce de travail légale (ex: heure B de la figure 2.2), alors les deux tâches qu'il divise peuvent être jointes l'une à l'autre pour en former une seule (élimination d'une contrainte).

Chapitre 3

Revue de la littérature

Le problème de recouvrement est généralement résolu par des méthodes d'**énumération implicite**, mieux connue sous le nom anglais "**branch and bound**". La première partie de ce chapitre est consacrée à la description générale des méthodes d'énumération implicite, puis la seconde partie porte sur une synthèse de quelques articles pertinents concernant les caractéristiques importantes de ces méthodes, soient les **bornes inférieures**, les **bornes supérieures** et les critères de **branchement**.

3.1 Relaxation linéaire

La relaxation linéaire du problème de recouvrement est utilisée dans les méthodes d'énumération implicite. Rappelons tout d'abord la formulation du PR, telle que définie au chapitre 2:

$$ZP = \min \sum_{j \in N} c_j x_j \tag{3.1}$$

sous les contraintes

$$\sum_{j \in N} a_{ij} x_j \geq 1 \quad , i \in M \tag{3.2}$$

$$x_j \in \{0,1\} \quad , j \in N \tag{3.3}$$

La **relaxation linéaire** de ce problème consiste à remplacer la

contrainte (3.3) par:

$$0 \leq x_j \leq 1 \tag{3.4}$$

L'énoncé du **PROBLÈME DE RELAXATION LINÉAIRE** (PL) est alors le suivant:

$$ZPL = \min \sum_{j \in N} c_j x_j$$

s.l.c.

$$\sum_{j \in N} a_{ij} x_j \geq 1 \quad , i \in M$$

$$0 \leq x_j \leq 1 \quad , j \in N$$

et le **PROBLÈME DUAL** (PD) associé au PL est donné par:

$$ZD = \max \sum_{i \in M} u_i \tag{3.5}$$

s.l.c.

$$\sum_{i \in M} a_{ij} u_i \leq c_j \quad , j \in N \tag{3.6}$$

$$u_i \geq 0 \quad , i \in M \tag{3.7}$$

Les variables u_i sont appelées variables duales (ou **multiplicateurs de Lagrange**).

Les propriétés suivantes présentent des relations entre les valeurs optimales des

problèmes PR, PRL et PD, qui sont exploitées dans la méthode d'énumération implicite.

Propriété 3.1 $ZPL \leq ZP$

Preuve — Le PL correspond au PR, dont la contrainte 3.3 a été relaxée. Puisque c'est un problème de minimisation, alors $ZPL \leq ZP$.

Propriété 3.2 $ZD = ZPL$

(Propriété de la dualité forte [voir 16, Nemhauser et Wolsey])

Propriété 3.3 $ZD \leq ZP$

Preuve — Puisque $ZD = ZPL$ (propriété 3.2) et que $ZPL \leq ZP$ (propriété 3.1), alors $ZD = ZPL \leq ZP$.

3.2 Énumération implicite

La méthode d'énumération implicite est divisée en trois parties importantes :

1. La structure de branchement.
2. Le calcul de bornes inférieures.
3. Le calcul de bornes supérieures.

A partir du PR original, le **branchement** crée de nouveaux sous-problèmes (noeuds) en partitionnant l'ensemble de ses solutions entières. Ceci élimine généralement une partie des solutions continues du problème courant (PL). Les sous-problèmes sont obtenus en ajoutant des contraintes supplémentaires, ou en fixant des variables à 0 ou 1. Pour chaque sous-problème, le calcul d'une borne inférieure **locale** sur sa solution optimale entière, est effectué.

De plus, une borne supérieure **locale** est calculée, sur la solution optimale d'un sous-problème. Puisque toute solution entière d'un sous-problème est une solution entière réalisable du PR original, toute borne supérieure locale est aussi une borne supérieure **globale**.

Le principe de cette méthode est la diminution de l'écart entre les bornes supérieure et inférieure (figure 3.1), d'autres détails concernant cette méthode seront présentés au chapitre 4.

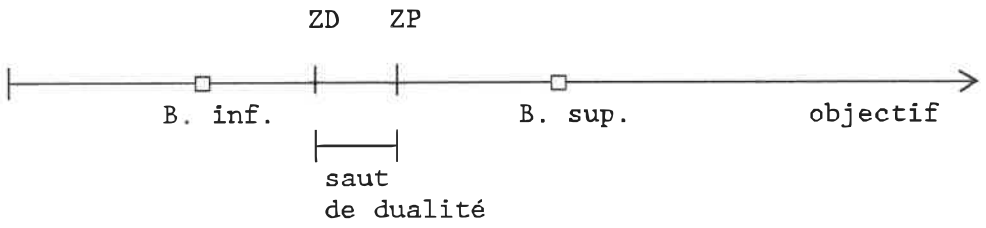

Figure 3.1: Bornes inférieures et supérieure d'un PR.

3.3 Notations

Les notations suivantes seront utilisées dans le présent chapitre, ainsi que dans les chapitres subséquents:

$A_{(0,1)}$	:Matrice des contraintes, dont les éléments a_{ij} sont 0 ou 1.
PR	:Sous-problème courant.
M	:Ensemble des indices des lignes du sous-problème courant.
N	:Ensemble des indices des colonnes du sous-problème courant.
PD	:Problème dual associé à la relaxation continue du sous-problème courant.
$U = (u_i)$	:Vecteur des variables duales (ou multiplicateurs de Lagrange).
$e = [1, 1, \cdots, 1]^T$	:Vecteur unitaire représentant le membre de droite de PR.
$I_j = \{i \in M : a_{ij} = 1\}$	:Ensemble des indices des lignes qui contiennent un élément non-nul dans la colonne j.
$J_i = \{j \in N : a_{ij} = 1\}$	:Ensemble des indices des colonnes qui contiennent un élément non-nul dans la ligne i.
$\bar{c}_j = c_j - \sum_{i \in I_j} a_{ij} u_i$	:coût réduit associé à la colonne j.

BI :Borne inférieure (**locale**) d'un sous-problème.

BS :Borne supérieure (**globale**) du problème original.

3.4 Saut de dualité

Le saut de dualité d'un problème de recouvrement est défini comme étant la différence entre ZP et ZD (figure 3.1). La propriété suivante permet d'identifier le cas où celui-ci est nul:

Propriété 3.4

Soit U, la solution otpimale du PD et SOL, l'ensemble défini par $\{j \in N : \bar{c}_j = 0\}$. Le saut de dualité est nul si et seulement si il existe un sous-ensemble

REA inclus dans SOL tel que:

$$\sum_{j \in REA} a_{ij} \begin{cases} = 1 \quad \text{si} \quad u_i > 0. \\ \\ \geq 1 \quad \text{si} \quad u_i = 0. \end{cases} , i \in M$$

Preuve

a. ($\Rightarrow$)

Si le saut de dualité est nul, alors il existe un ensemble $REA \subset N$ (ensemble des variables égales à 1 dans une solution primale réalisable) tel que:

$$(ZP) \quad = \quad (ZD)$$

$$\sum_{j \in REA} c_j = \sum_{i \in M} u_i \tag{3.8}$$

Puisque REA est réalisable, il recouvre toutes les lignes:

$$\sum_{j \in REA} a_{ij} \geq 1, i \in M \tag{3.9}$$

Afin de simplifier les notations, on définit :

$$d_i = \sum_{j \in REA} a_{ij}, i \in M \tag{3.10}$$

Le vecteur U étant une solution réalisable du problème dual, les coûts réduits sont non-négatifs:

$$c_j - \sum_{i \in M} a_{ij} u_i \geq 0, j \in N \tag{3.11}$$

En sommant l'équation (3.11) pour tout $j \in REA$, on obtient la relation suivante:

$$\sum_{j \in REA} c_j \geq \sum_{i \in M} d_i u_i \tag{3.12}$$

Puisque $u_i \geq 0 (i \in M)$, les équations (3.9) et (3.10) nous permettent d'établir une nouvelle relation, en sommant sur tous les multiplicateurs:

$$\sum_{i \in M} d_i u_i \geq \sum_{i \in M} u_i \tag{3.13}$$

Pour que l'équation (3.8) soit vraie, il faut que la relation d'inégalité devienne une relation d'égalité dans les équations (3.12) et (3.13):

$$\sum_{j \in REA} c_j = \sum_{i \in M} d_i u_i \tag{3.14}$$

et

$$\sum_{i \in M} d_i u_i = \sum_{i \in M} u_i \tag{3.15}$$

Pour que l'équation (3.14) soit vraie, la somme des coûts réduits sur les $j \in REA$ doit être nulle (en utilisant l'équation 3.10):

$$\sum_{j \in REA} (c_j - \sum_{i \in M} a_{ij} u_i) = 0 \tag{3.16}$$

Donc, l'équation (3.11) implique que si (3.16) est vraie, alors les coûts réduits de $j \in REA$ sont nuls:

$$c_j - \sum_{i \in M} a_{ij} u_i = 0, j \in REA \tag{3.17}$$

Puisque que $d_i \geq 1 (i \in M)$, pour que l'équation (3.15) soit vraie, $d_i (i \in M)$ doit respecter la condition qui suit:

$$u_i > 0 \Rightarrow d_i = 1 \quad , \quad i \in M \tag{3.18}$$

S'il n'existe pas de saut de dualité, il faut donc qu'il existe un sous-ensemble REA de N, tel que:

$$\sum_{j \in REA} a_{ij} \begin{cases} = 1 \quad \text{si} \quad u_i > 0 \\ \\ \geq 1 \quad \text{si} \quad u_i = 0 \end{cases} , i \in M \quad \text{(équations 3.9, 3.10 et 3.18)}$$

et

$$\bar{c}_j = 0, j \in REA \quad \text{(équation 3.17)}$$

b. ($\Leftarrow$)

S'il existe un sous-ensemble REA de N, tel que

$$\sum_{j \in REA} a_{ij} \begin{cases} = 1 \quad \text{si} \quad u_i > 0 \\ \\ \geq 1 \quad \text{si} \quad u_i = 0 \end{cases} , i \in M \tag{3.19}$$

et

$$\bar{c}_j = 0, j \in REA \tag{3.20}$$

alors,

$$
\begin{aligned}
ZP &\leq \sum_{j \in REA} c_j \quad \text{car } (x_j = 1 \text{ ,si } j \in REA \text{ est une sol. réal. du PR}) \\
&= \sum_{j \in REA} \sum_{i \in M} a_{ij} u_i \quad \text{(coût réduits nuls, équation 3.20)} \\
&= \sum_{i \in M} u_i \qquad \text{(équation 3.19)} \\
&= ZD
\end{aligned}
$$

Donc $ZP = ZD$, car $ZP \geq ZD$ par la propriété 3.3, et le saut de dualité est nul.

3.5 Bornes inférieures

Beaucoup d'efforts ont été concentrés sur cette partie de la méthode d'énumération implicite. Une borne inférieure est très souvent obtenue à l'aide d'un **algorithme dual**, résolvant le problème **dual** ou la relaxation linéaire du problème de recouvrement. Nous présenterons dans cette section, un échantillon d'algorithmes duaux, parmi les plus connus.

3.5.1 La programmation linéaire

Un algorithme de programmation linéaire permet d'obtenir une excellente borne inférieure, minimisant alors le nombre de noeuds créés (par rapport aux méthodes heuristiques). C'est, par contre, une méthode coûteuse en temps de calcul, car tout comme avec le PP, la solution du PR est souvent dégénérée. Afin de se prémunir contre cette situation, des procédures supplémentaires très coûteuses (Ex: per-

turbation des données, procédure lexicographique) [voir 10, Gondran et Minoux], doivent être ajoutées.

3.5.2 Méthodes heuristiques

Les méthodes heuristiques produisent une solution duale réalisable rapidement. La borne inférieure, obtenue par un algorithme de ce type, n'est alors qu'un **estimé** de ZD, donc moins bonne que celle obtenue par la programmation linéaire. Parmi les heuristiques les plus souvent utilisées, on trouve:

Algorithme glouton

L'**algorithme glouton** est une heuristique simple, produisant rapidement une borne inférieure, **raisonnablement** bonne. Il trouve une solution duale réalisable U, en parcourant les lignes $i \in I$ une à une et en affectant $\min_{j \in J_i} \bar{c}_j$ à u_i :

ÉTAPE 1 (INITIALISATION)

- $i = 0$.
- $\bar{c}_j = c_j, j \in N$.

ÉTAPE 2 (AJUSTEMENT DES MULTIPLICATEURS)

- $i = i + 1$.
- $u_i = \min_{j \in J_i}\{\bar{c}_j\}$.
- $\bar{c}_j = \bar{c}_j - u_i, j \in J_i$ (Mise à jour des coûts réduits).

ÉTAPE 3 (TEST D'ARRÊT)

- Si $i = m$, alors FIN.
- Aller à l'étape 2.

Méthodes de sous-gradients

On pourrait qualifier les méthodes de **sous-gradients**, de **compromis** entre la programmation linéaire et les heuristiques duales usuelles. Elles produisent une **excellente** borne inférieure, en temps raisonnable. On observe en pratique que $(t_{heu} < t_{s.grad.} < t_{pr.lin.})$.

Soit le lagrangien $L(X,U)$ défini ci-dessous:

$$L(X,U) = CX + Ue - UAX = \sum_{j \in N} x_j(c_j - \sum_{i \in M} a_{ij}u_i) + \sum_{i \in M} u_i \qquad (3.21)$$

Le principe des méthodes de sous-gradient [11, Held, Wolfe et Crowder] consiste à trouver un bon estimé de la valeur optimale du problème:

$$\max_{U \in F} \min_{X \in G} L(X,U) \qquad (3.22)$$

où, $F = \{U : 0 \leq u_i \leq u_i'\}$, $u_i' = \min_{j \in J_i} c_j$

et $G = \{X : 0 \leq x_j \leq 1\}$

À l'itération k de l'algorithme, la fonction $L(X(U^k), U^k) = \min_{X \in G} L(X, U^k)$, $(U_k \in G)$ est calculée. Cette valeur constitue une **borne inférieure** sur la valeur optimale de la fonction objective du problème de recouvrement.

Il existe de nombreux algorithmes de sous-gradient, celui présenté ci-dessous est le plus souvent rencontré:

ÉTAPE 1 (AJUSTEMENT DES VARIABLES x_j)

soit, $U^k \in G$, alors le vecteur $X(U^k)$ qui minimise $L(X, U^k)$ est

$$x_j^k = \begin{cases} 1 \quad \text{si} \quad \bar{c}_j < 0, \\ 0 \quad \text{si} \quad \bar{c}_j > 0, \qquad j \in N \\ \in [0,1] \quad \text{si} \quad \bar{c}_j = 0, \end{cases}$$

ÉTAPE 2 (CALCUL DE SOUS-GRADIENT)

$$d^k = e - AX^k$$

Le vecteur d^k est appelé **sous-gradient.**

ÉTAPE 3 (CALCUL DU PAS)

$$t^k = \frac{\beta_k (Z_u - L(X^k, U^k))}{||d^k||^2}$$

où, Z_u est une borne supérieure sur ZP,

$||v||$ est la norme du vecteur v,

et $0 < \beta_k \leq 2$ est un coefficient de relaxation.

ÉTAPE 4 (MODIFICATION DES MULTIPLICATEURS u_i)

$$U^{k+1} = P_F(U^k + t_k d_k),$$

où P_F est une projection du vecteur $H^k = (U^k + t_k d_k)$ sur le domaine réalisable F, c'est-à-dire que:

$$u_i^{k+1} = \begin{cases} u_i' & \text{si } h_i^k > u_i', \\ h_i^k & \text{si } 0 \leq h_i^k \leq u_i', \quad h_i^k \text{ est la i}^{\text{ième}} \\ & \qquad\qquad\qquad\qquad \text{composante de } H^k. \\ 0 & \text{si } h_i^k < 0, \end{cases}$$

$$k = k + 1.$$

Les étapes 1-4 sont répétées en accord avec un critère d'arrêt pré-établi, et le vecteur de départ U^0 provient généralement d'une heuristique duale.

L'algorithme du sous-gradient trouve, en pratique, une borne inférieure très près (différence $< 1\%$) de l'optimum du problème dual, en peu d'itérations.

Algorithme 3-OPT

L'algorithme **3-OPT** est une heuristique particulière utilisée par Fisher et Kedia [9]. Il débute à partir d'une solution duale réalisable (obtenue à l'aide d'un algorithme glouton) et tente d'augmenter la valeur $\sum_{i \in I} u_i$ en modifiant adéquatement le vecteur U. Le principe de base est la recherche d'un triplet de lignes i_1, i_2 et i_3 pour lequel le changement suivant:

$$u_{i_1} = u_{i_1} - \delta_{\max}$$

$$u_{i_2} = u_{i_2} + \delta_{\max}, \quad \text{où} \quad \delta_{\max} > 0$$

$$u_{i_3} = u_{i_3} + \delta_{\max}$$

est possible, $\delta_{\max}$ étant la valeur **maximum** permettant de conserver la faisabilité duale. La borne inférieure augmente alors de $\delta_{\max}$. Pour ce faire, les contraintes duales sont partitionnées en deux sous-ensembles:

$$J_a = \{j \in J : \sum_{i \in I_j} u_i = c_j\}$$

$$J^i = \{j \in J : \sum_{i \in I_j} u_i < c_j\}$$

L'algorithme 3-OPT est composé de deux étapes principales:

ÉTAPE 1

Sélection d'un triplet i_1, i_2, i_3 **valide**, c'est-à-dire vérifiant les conditions suivantes:

(a) $u_{i_1} > 0$. (nécessaire pour augmenter la borne inférieure)

(b) $a_{i_2 j} + a_{i_3 j} - a_{i_1 j} \leq 0, \quad j \in J_a$. (maintien de la faisabilité duale)

ÉTAPE 2

Les multiplicateurs u_{i_1}, u_{i_2} et u_{i_3} sont modifiés selon la plus grande valeur δ qui maintient la faisabilité duale:

$$J^{i^*} = \{j \in J^i : a_{i_2 j} + a_{i_3 j} - a_{i_1 j} = 1\}$$

$$\delta = \min\{\min_{j \in J^{i^*}} \left[c_j - \sum_{i \in I_j} u_i \right], u_{i_1}\}$$

Les étapes 1,2 sont alors répétées jusqu'à ce qu'aucun autre ajustement du vecteur U ne soit possible. L'arrêt se produit à l'étape 2, après avoir examiné toutes les possibilités sans trouver de triplet respectant les contraintes (a) et (b).

L'algorithme 3-OPT, selon les résultats obtenus [9, Fisher and Kedia] sur des problèmes peu denses ($\leq 5\%$) et générés aléatoirement, trouve une borne inférieure égale ou très près de l'optimum continu. De plus, sur les problèmes ayant un saut de dualité nul, la méthode d'énumération implicite (contenant le 3-OPT) trouve la solution optimale en très peu de temps.

Algorithme MAM

L'algorithme MAM [5, Chan, Bean et Yano], conçu spécialement pour résoudre le problème de partitionnement (PP), peut être adapté au problème de recouvrement (PR) après quelques modifications **mineures**. Il démarre avec une solution duale réalisable (obtenue à l'aide d'un algorithme glouton), puis tente de modifier les multiplicateurs afin que l'ensemble $SOL = \{j \in J : \bar{c}_j = 0\}$ forme un recouvrement du PR. Le principe général de MAM est d'établir une relation de complémentarité entre les solutions duales et primales.

Soient les notations suivantes:

- t: Numéro de l'itération courante.
- T: Nombre maximum d'itérations permis.
- $SOL = \{j \in N : \bar{c}_j = 0\}$.
- $VIOLE = \{j \in N : \bar{c}_j < 0\}$.
- $LB = \sum_{i \in I} u_i$.
- $n_i = |SOL \cap J_i|$.
- $S_0 = \{i \in M : n_i = 0\}$.
- $S_1 = \{i \in M : n_i = 1\}$.
- $S_2 = \{i \in M : n_i \geq 2 \text{ et } u_i > 0\}$.[1]

Cette heuristique duale tente d'obtenir un ensemble, défini SOL, pour lequel $S_0 = S_2 = \emptyset$, car ceci implique que $LB = ZP$. En effet, dans ce cas $\sum_{i \in M} u_i = \sum_{j \in SOL} c_j$ (voir propriété 3.4) , et l'étude du noeud courant est terminée puisque les bornes inférieure et supérieure sont égales. L'algorithme (incluant les modifications mineures) est divisé en cinq étapes:

ÉTAPE 1 (INITIALISATION)

- Si $S_0 = S_2 = \emptyset$, alors FIN .
- $t = 1$.

ÉTAPE 2 (AUGMENTATION DES MULTIPLICATEURS)

[1] Modificaton apportée à la notation afin d'adapter l'algorithme MAM au PR.

- Si $S_0 \neq \emptyset$, alors $UP = S_0$ et MODE $= 1$.

  Si $S_0 = \emptyset$, alors $UP = \bigcup_{j \in SOL'}\{I_j \cap S_1\}$ et MODE $= 2$,

  où $SOL' = \{j \in SOL : I_j \cap S_1 \neq \emptyset, I_j \cap S_2 \neq \emptyset\}$.

- Si $UP = \emptyset$, alors FIN.

  sinon, $u_i = u_i + \min_{j \in J_i \backslash SOL}\{\bar{c}_j / |I_j \cap UP|\}, i \in UP$.

ÉTAPE 3 (DIMINUTION DES MULTIPLICATEURS DES CONTRAINTES DUALES VIOLÉES)

- Si MODE $= 2$, alors faire:

  (a) $u_i = \max\{0, (u_i + (n_i / \sum_{i \in I_j \cap S_2} n_i)\bar{c}_j)\}$[2],

  $i \in \{I_j \cap S_2\}$, $j \in \{SOL' \cap VIOLE\}$.

  (b) $u_i = \max\{0, (u_i c_j / (c_j - \min\{\bar{c}_j, 0\}))\}^2$, $j \in VIOLE$.

ÉTAPE 4 (COUVRIR CHAQUE CONTRAINTE PRIMALE $\Rightarrow n_i \geq 1, i \in M$)

- Si MODE $= 2$, alors tant que $S_0 \neq \emptyset$ faire

  (a) $p = \min_{i \in S_o}\{i\}$.

  (b) $u_p = u_p + \min_{j \in J_p}\{\bar{c}_j\}$.

  (c) $S_o = S_o \backslash I_q$, où $q = \arg\min_{j \in J_p}\{\bar{c}_j\}$.

ÉTAPE 5 (CRITÈRE D'ARRÊT)

- Si $S_0 = S_2 = \emptyset$, alors FIN.

[2]Lignes où des modifications ont été apportées à l'algorithme MAM original, afin de l'adapter au PR

- $t = t + 1$.
- Si $t = T$, alors FIN .
- Aller à l'étape 2 .

3.6 Bornes supérieures

Plusieurs heuristiques **primales** efficaces existent pour trouver une borne supérieure sur un problème de recouvrement. Ce sont généralement des procédures simples et qui donnent des résultats très satisfaisants. Nous désirons, dans cette section, rappeler l'idée générale de quelques-unes des bonnes heuristiques connues actuellement.

3.6.1 Algorithme glouton

Un algorithme glouton classique [6, Chvátal] consiste à fixer à 1 la variable x_j qui minimise de coût $c_j/|I_j \cap$ lignes non-recouvertes$|$, et cette étape est répétée jusqu'à ce que toutes les lignes soit recouvertes. Celui-ci se décompose en trois étapes:

ÉTAPE 1 (INITIALISATION)

- $LIBRE = M$.
- $SOL = \emptyset$.

ÉTAPE 2 (CHOIX DE LA VARIABLE)

– $SOL = SOL \cup \{k\}$, où $\overbrace{c_k/|I_k \cap LIBRE|}^{\text{fonction de coût}} = \min_{j \in J \setminus SOL}(c_i/|I_k \cap LIBRE|)$

ÉTAPE 3

– $LIBRE = LIBRE \setminus I_k$.

– si $LIBRE = \emptyset$, alors FIN .

sinon, aller à l'étape **2** .

Lorsque $LIBRE = \emptyset$, SOL constitue alors une solution réalisable ($x_j = 1, j \in SOL$) et $\sum_{j \in SOL} c_j$, une borne supérieure sur Z_{opt}, l'objectif de la solution optimale.

L'écart maximal entre Z_{heu} et Z_{opt} est donné par la relation suivante:

$$\frac{Z_{heu}}{Z_{opt}} \leq \sum_{j=1}^{d} \frac{1}{j}, \qquad \text{où} \quad d = \max_{j \in J} |I_j| \tag{3.23}$$

3.6.2 Algorithmes de Balas et Ho

L'algorithme PRIMAL1 [1, Balas et Ho] utilise l'algorithme glouton précédent, mais avec une modification de la fonction de coût. Il est divisé en deux étapes, soient le **recouvrement** (1) et la **relaxation** (2):

ÉTAPE 1 (RECOUVREMENT)

– Appliquer l'algorithme glouton, en utilisant une des cinq fonctions de coût suivantes :

(i) c_j ;

(ii) c_j/k_j ;

(iii) $c_j/\log_2 k_j$; , où $k_j = |I_j \cap LIBRE|$

(iv) $c_j/(k_j \log_2 k_j)$;

(v) $c_j/(k_j \ln k_j)$.

* Remarque : si $k_j = 1$, alors on remplace $\log_2 k_j$ (ou $\ln k_j$) par 1.

ÉTAPE 2 (RELAXATION)

- Retrancher de SOL, les colonnes contenant un élément non-nul pour au moins une ligne **surrecouverte.**

Ces 2 étapes sont répétées 3 fois, en utilisant une fonction de coût différente à chaque fois, lors du recouvrement (étape 1).

Sur les problèmes testés[3], les résultats obtenus avec PRIMAL1 furent significativement meilleurs qu'avec une seule application de l'algorithme glouton.

Un second algorithme, nommé **PRIMAL5** [1, Balas et Ho], utilise le vecteur U obtenu à partir d'un algorithme de sous-gradient. L'ensemble $SOL^4 = \{j \in J : \bar{c}_j = 0\}$ correspond alors à un ensemble-solution réalisable, et $\sum_{j \in SOL} c_j$ à une borne supérieure. Cet algorithme donne de très bons résultats sur les problèmes testés[3], et trouve une borne supérieure en moyenne de 4.37 % plus basse que celle obtenue avec PRIMAL1.

[3]Problèmes de $100 \leq m \leq 200$ lignes par $100 \leq n \leq 1000$ colonnes, générés aléatoirement dont la densité est de 2 % [1, Balas et Ho].

[4]Après projection de U sur le domaine réalisable ($\bar{c}_j \geq 0, j \in J$) et saturation des contraintes ($\min_{j \in J_i} \bar{c}_j = 0, i \in I$).

3.7 Critères de branchement

Chaque sous-problème est créé par branchement à partir de son prédécesseur, appelé **père**. Le choix du critère de branchement affectera évidement les sous-problèmes générés. Dans cette section, trois critères sont présentés, et l'on indique comment cela modifie le problème **courant**.

3.7.1 Sur une variable

Le branchement sur une variable est la méthode la plus classique utilisée dans une méthode d'énumération implicite pour résoudre un PR. Lors du branchement sur x_k , deux branches sont créées à partir du sous-problème courant (figure 3.2). Dans l'une d'elle, on fixe x_k à 0, ce qui correspond à éliminer une colonne au sous-problème courant. Dans la seconde branche, on fixe x_k à 1, ce qui élimine la colonne k et les lignes $i \in I_k$ du sous- problème courant car les contraintes associées sont maintenant saturées. De plus, un coût fixe c_k s'ajoute à l'objectif du nouveau sous-problème créé dans cette seconde branche.

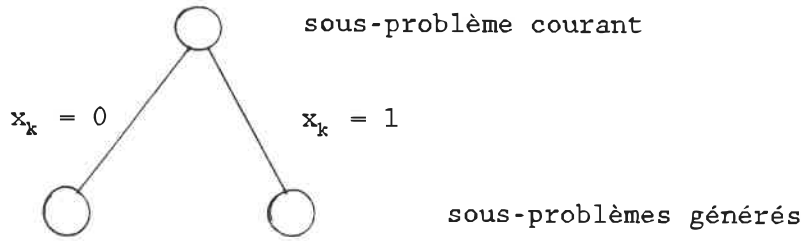

Figure 3.2: Branchement sur une variable, où les noeuds correspondent aux sous-problèmes.

3.7.2 Sur une contrainte

Ce type de branchement, exploité par Fisher et Kedia [9], est effectué à partir d'une contrainte ℓ, formant plusieurs nouveaux sous-problèmes dans lesquels une variable $x_j, j \in J_\ell$ (figure 3.3) est fixée à 1.

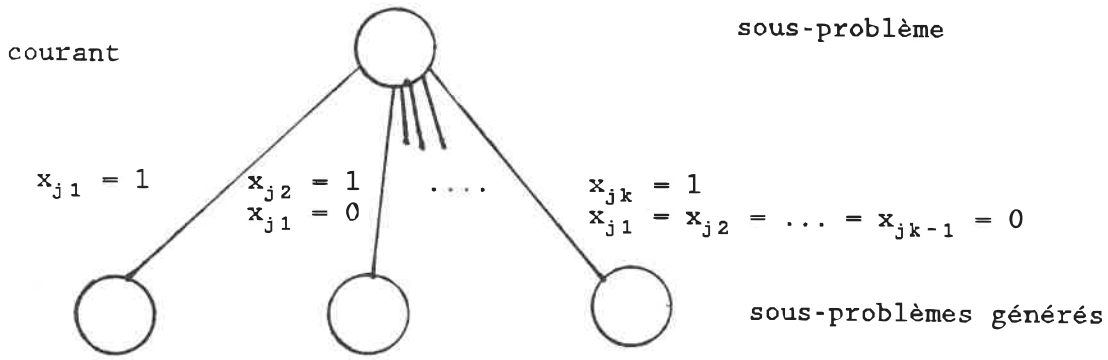

Figure 3.3: Branchement sur les variables de la contrainte ℓ.

$k = |J_\ell|$ nouveaux sous-problèmes, mutuellement exclusifs, sont donc créés.

3.7.3 Addition d'une contrainte

Le dernier critère présenté a été utilisé par Ryan et Foster [17] dans la résolution du problème de partitionnement (PP). Faisant d'abord appel à la programmation linéaire pour trouver une borne inférieure, la solution obtenue permet d'identifier les **sous-matrices cycliques** (section 5.1) qui produisent une solution fractionnaire. Le but visé par le branchement est d'obtenir une solution entière, donc de **briser** (éliminer les cycles) des sous-matrices cycliques. Pour chaque couple de rangées ayant au moins un colonne commune et dont la variable associée est non-nulle dans la solution continue, on calcule:

$$\theta_{kl} = \sum_{j \in J(k,l)} x_j, \quad \text{où} \quad J(k,l) = J_k \cap J_l,$$

où X est la solution continue obtenue par programmation linéaire.

Soit l'ensemble T^2 est défini comme suit:

$$T^2 = \{(k,l) : 0 < \theta_{kl} < 1\}$$

Le branchement consiste alors à produire deux branches, suite à l'addition de deux contraintes mutuellement exclusives (figure 3.4), avec $(k,l) \in T^2$:

$$\sum_{j \in J(k,l)} x_j \leq 0, \tag{3.24}$$

$$\sum_{j \in J(k,l)} x_j \geq 1, \tag{3.25}$$

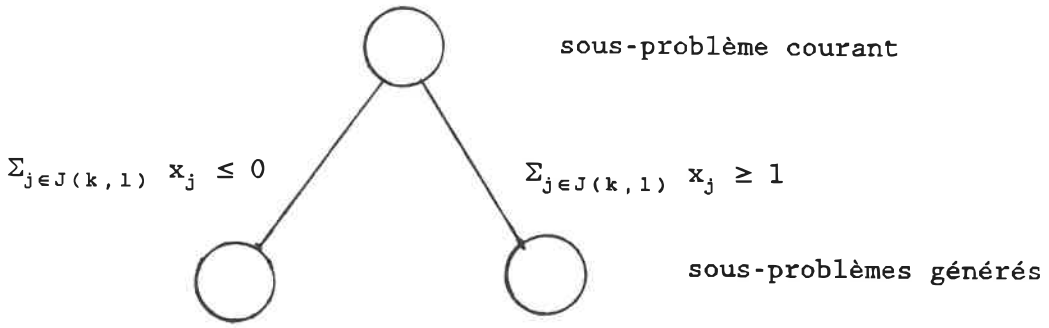

Figure 3.4: Branchement sur contraintes disjonctives.

L'addition de la contrainte (3.24) modifie le sous-problème en éliminant toutes les colonnes $j \in J(k,l)$, tandis que l'addition de la contrainte (3.25) élimine les lignes k et l (les contraintes deviennent redondantes).

Notre mémoire est basé sur l'étude de ce même critère de branchement. L'objectif visé est également l'élimination de **sous-matrices cycliques**, mais la sélection des lignes i_1 et i_2 diffère cependant de celle utilisée par Ryan et Foster [17], puisque nous exploitons d'autres conditions nécessaires, dans lesquelles on utilise les solutions obtenues par des **méthodes heuristiques** plutôt que par la **programmation linéaire**. Nous disposons donc d'un outil moins puissant (nous ne connaissons pas la solution X du PL), au point de vue information, que ne l'est la programmation linéaire.

Remarque:

La recherche d'une facette de la forme $\sum_{j \in H} x_j \geq 1$ présenterait une autre utilisation de ce critère de branchement, et pourrait certainement faire l'objet de recherches éventuelles, vu l'intérêt croissant pour les méthodes de coupes **fortes**.

Chapitre 4

Méthode de résolution

Ce chapitre est consacré à la description de la méthode de résolution. L'organigramme général et ses composantes principales seront d'abord présentés dans la section 1, puis une description de chacune des composantes sera donnée dans les sections subséquentes.

4.1 Organigramme

Nous devons tout d'abord distinguer le noeud 0 des autres noeuds, puisqu'une quantité importante de travail est effectuée sur le problème initial avant le premier branchement. Deux objectifs importants sont alors visés, soient:

- Réduction de la taille du problème.
- Minimisation de l'écart entre les bornes inférieure et supérieure.

La qualité du traitement initial aura une influence importante sur le temps de résolution et le nombre de noeuds générés. Si le temps moyen de traitement d'un noeud est inférieur à 10 % de celui du traitement initial, il n'en demeure pas moins que ce premier effort peut être très avantageux, s'il permet une diminution substentielle du nombre de noeuds générés, ce fait étant d'ailleurs observé en pratique.

Le travail effectué au noeud 0 est composé principalement des procédures suivantes :

1. PRÉ-TRAITEMENT (ensemble de tests de réduction).

2. ALGORITHME PRIMAL (fournit une première borne supérieure).

3. ALGORITHME DUAL GLOUTON (fournit une solution duale réalisable).

4. ALGORITHME DE SOUS-GRADIENT (réoptimisation à partir de la solution duale réalisable, produisant une borne inférieure)

5. CALCUL DE PÉNALITÉ (procédure visant à réduire le nombre de colonnes).

6. ALGORITHME PRIMAL SATURATION-RELAXATION (produit une borne supérieure en utilisant la solution duale obtenue par l'algorithme de sous-gradient).

Les procédures principales appliquées au noeud courant (excluant le branchement) sont les suivantes:

1. ALGORITHME DE SOUS-GRADIENT (appliqué à partir de la solution duale du noeud père)

2. CALCUL DE PÉNALITÉ (réduction du système)

3. ALGORITHME PRIMAL SATURATION-RELAXATION

L'organigramme principal (figure 4.1) présente la procédure proposée et la séquence des prises de décision. Les fonctions des 4 tests sont les suivantes:

TEST 1. Si la différence entre les bornes inférieure et supérieure (ZP-ZD) est supérieure à δ (tolérance choisie), alors on continue le traitement du problème (1.1); sinon, la solution du problème est trouvée et on arrête l'exécution

(1.2). Selon cette seconde éventualité, le problème possèderait alors un saut de dualité "nul".

TEST 2. Si la différence entre les bornes inférieure et supérieure (ZP-ZD) est supérieure à δ, alors on continue le traitement du sous-problème; sinon, on ferme le noeud (2.2) et on sélectionne un nouveau sous-problème à traiter.

TEST 3. Si la différence entre les bornes inférieure et supérieure (ZP-ZD) est supérieure à δ, alors on sauvegarde la solution duale du sous-problème (on garde le noeud ouvert) et on sélectionne un nouveau sous-problème à traiter (3.1); sinon, on ferme le noeud (3.2) et on sélectionne un nouveau sous-problème à traiter.

TEST 4. S'il existe au moins un noeud ouvert, alors choisir le prochain candidat et brancher sur celui-ci (4.1); sinon, arrêter l'éxécution (4.2).

4.2 Description des composantes

4.2.1 Pré-traitement

Le pré-traitement constitue une étape très importante de la méthode d'énumération implicite, sa fonction étant de réduire la taille du problème (la matrice des contraintes), en éliminant des contraintes ou des variables sans toutefois en exclure la solution optimale. Trois critères d'élimination importants sont inclus dans le pré-traitement de notre algorithme, soient la **redondance de lignes, la redondance de colonnes** et **le recouvrement unique d'une ligne.**

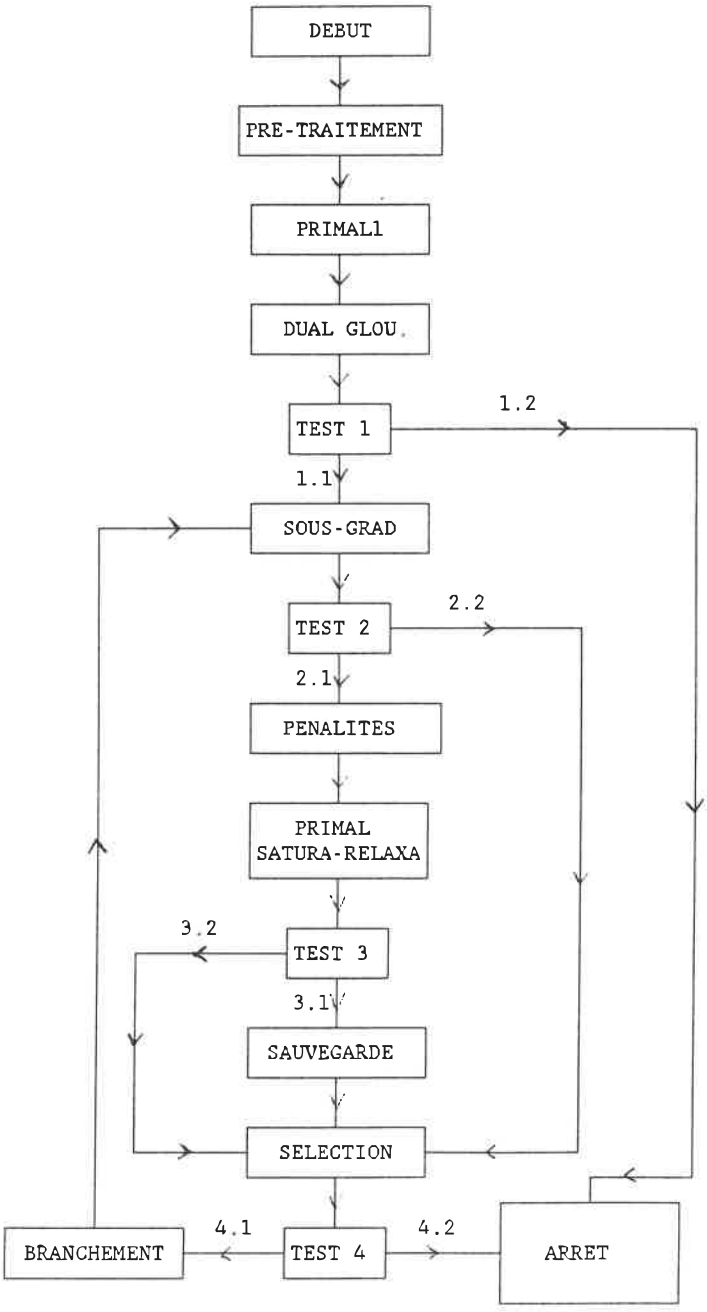

Figure 4.1: Organigramme du programme principal.

Redondance de lignes

Si J_{i_1} est inclus dans J_{i_2}, alors la ligne i_2 peut être éliminée de la matrice des contraintes (figure 4.2). En effet, la contrainte i_2 est redondante par rapport à la contrainte i_1.

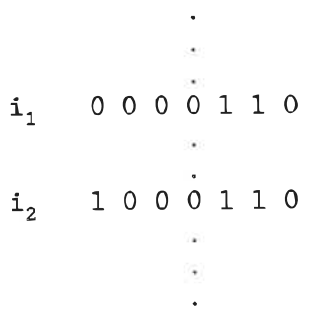

Figure 4.2: PR à 6 variables. Puisque $x_5 + x_6 \geq 1(i_1)$, alors $x_1 + x_5 + x_6 \geq 1$. La contrainte i_2 est donc une conséquence (ne coupe pas le domaine réalisable) de la contrainte i_1 et peut être éliminée.

Ce critère d'élimination est appliqué sur chaque couple de ligne (i_1, i_2) (où $i_1, i_2 \in M$ et $i_1 < i_2$), à l'aide de l'algorithme suivant:

ÉTAPE 1 (INITIALISATION)

- $LI_1 = J_{i_1}$. (liste des éléments non-nuls non-traités de la ligne i_1)
- $LI_2 = J_{i_2}$. (liste des éléments non-nuls non-traités de la ligne i_2)

- $REDON_1 = VRAI$. (indique si la ligne i_1 est redondante par rap. à i_2)
- $REDON_2 = VRAI$. (indique si la ligne i_2 est redondante par rap. à i_1)
- $DUO = VRAI$. (indique si on sélectionne dans les deux ensembles LI_1 et LI_2)

ÉTAPE 2 (SÉLECTION DU PROCHAIN ÉLÉMENT DE LA LIGNE i_1 À TRAITER)

- Si $LI_1 = \emptyset$, alors aller à l'étape 5.
- $j_1 = \min_{j \in LI_1} j$.
- $LI_1 = LI_1 \backslash \{j_1\}$.
- Si $DUO = VRAI$, alors $DUO = FAUX$ et aller à l'étape 3.
  sinon, aller à l'étape 4.

ÉTAPE 3 (SÉLECTION DU PROCHAIN ÉLEMENT DE LA LIGNE i_2 À TRAITER)

- SI $LI_2 = \emptyset$, alors aller à l'étape 6.
- $j_2 = \min_{j \in LI_2} j$.
- $LI_2 = LI_2 \backslash \{j_2\}$.

ÉTAPE 4 (TEST DE REDONDANCE)

- Si $j_1 = j_2$, alors $DUO = VRAI$ et aller à l'étape 2.
- Si $j_1 < j_2$, alors $REDON_2 = FAUX$ et aller à l'étape 2.
- Si $j_1 > j_2$, alors $REDON_1 = FAUX$ et aller à l'étape 3.

ÉTAPE 5 (VÉRIFICATION DE REDONDANCE DE LA LIGNE i_2)

- Si $REDON_2 = VRAI$, alors la ligne i_2 est redondante par rapport à i_1, nous pouvons donc l'**éliminer.**

ÉTAPE 6 (VÉRIFICATION DE REDONDANCE DE LA LIGNE i_1)

- Si $REDON_1 = VRAI$, alors la ligne i_1 est redondante par rapport à i_2, nous pouvons donc l'éliminer.

Redondance des colonnes

Si I_{j_2} est inclus dans I_{j_1} et $c_{j_1} \leq c_{j_2}$, alors la colonne j_2 peut être éliminée (figure 4.3).

Preuve:

Soient $SOL = \{j \in N : x_j = 1\}$, une solution réalisable du problème de recouvrement et $OBJ = \sum_{j \in SOL} c_j$, l'objectif de la solution.

Supposons que $j_2 \in SOL$, ceci implique que:

$$OBJ = c_{j_2} + \sum_{j \in SOL \setminus \{j_2\}} c_j.$$

$$\geq c_{j1} + \sum_{j \in SOL \setminus \{j_2\}} c_j \quad (\text{puisque} \quad c_{j_1} \leq c_{j_2})$$

$$= \sum_{j \in SOL'} c_j, \quad \text{où} \quad SOL' = SOL \setminus \{j_2\} \cup \{j_1\}.$$

Puisque j_1 recouvre toutes les lignes que recouvre j_2, alors SOL' est réalisable et coûte moins ou autant que SOL.

Donc, la colonne j_2 peut être éliminée car il existe au moins une solution optimale dont l'ensemble des variables non-nulles **ne contient pas** j_2.

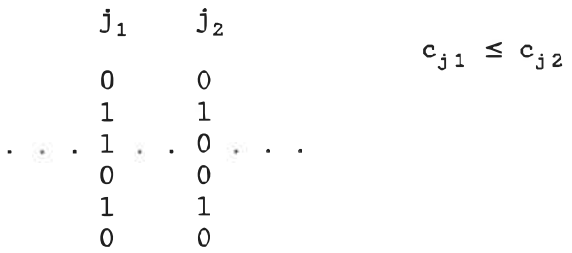

Figure 4.3: Matrice des contraintes $(m \times n)$, où $m = 6$. La colonne j_2 est redondante par rapport à la colonne j_1.

L'algorithme utilisé pour identifier les colonnes redondantes est **similaire** à celui utilisé pour la redondance des lignes. On doit cependant y ajouter une étape de comparaison du coût des colonnes.

Recouvrement unique

Si $J_{i_1} = j_1$, alors $x_{j_1} = 1$. Ceci entraîne l'élimination de la colonne j_1 ainsi que de toute ligne $i \in I_{j_1}$ (chapitre 3). La fixation de j_1 à 1 est triviale, puisque c'est

la seule colonne recouvrant la ligne i_1.

Notons que ce test d'élimination, bien que présent, n'intervient pas lors du pré-traitement (chapitre 6), car les problèmes ont été conçus de façon telle que chaque ligne contienne au moins 2 éléments non-nuls ($|J_i| > 1, i \in M$).

4.2.2 Algorithme dual glouton

Cet algorithme, de complexité en pire cas $O(|M||N|)$, consiste à affecter à la variable u_i, la valeur du plus petit **coût réduit** $\bar{c}_j$, où $\{j \in N : a_{ij} = 1\}$. Celui-ci est composé des 3 étapes suivantes:

ÉTAPE 1 (INITIALISATION)

- $\bar{c}_j = c_j, j \in N$. (initialisation des coûts réduits)
- $i = 1$. (initialisation du numéro de ligne)

ÉTAPE 2 (AJUSTEMENT DU MUTLIPLICATEUR)

- $u_i = \min \bar{c}_j, j \in J_i$.
- $\bar{c}_j = \bar{c}_j - u_i, j \in J_i$. (ajustement des coûts réduits)

ÉTAPE 3 (TEST D'ARRÊT)

- si $i = |M|$, alors **TERMINÉ.**
- $i = i + 1$.

– aller à l'étape 2 .

On obtient donc une solution duale (U) réalisable, qui sera utilisée comme solution de départ dans l'algorithme de sous-gradient.

4.2.3 Algorithme de sous-gradient

Cet algorithme utilise comme solution duale de départ:

- La solution de l'algorithme dual glouton, lors de son premier appel (noeud 0).
- La solution du **père**, lors du traitement d'un sous-problème, celle-ci demeurant réalisable après le branchement.

Notons que notre algorithme se distingue de l'algorithme classique (section 3.5.2) par la direction de **modification** des multiplicateurs, appelée S^k, qu'il choisit (voir ÉTAPE 4 ci-après). En effet, lors du premier appel de l'algorithme (noeud 0), S^k correspond à une direction **pondérée** , tandis qu'aux appels subséquents, S^k correspond simplement au **sous-gradient** .

L'algorithme comporte six étapes:

ÉTAPE 1 (INITIALISATION)

– $k = 0$. (Initialisation du numéro d'itération) .

– $U^k = U^0$, où U^0 est la solution de départ .

- $\sup_i = \min_{j \in J_i} c_j, i \in M$. (Valeur maximum que prend u_i)
- $\bar{c}_j = c_j - \sum_{i \in I_j} u_i^k$. (Initialisation des coûts réduits)
- $W = BS$, où BS est la borne supérieure actuelle du PR.
- $BI = \sum_{i \in M} u_i^k$. (Borne inférieure de départ)
- $\delta = 1.5$. (Coefficient de relaxation)
- $SEQUENCE = 0$. (Nombre d'itérations consécutives **sans augmentation** de BI)
- $MAXSEQ = 0$. (Nombre de séquences **consécutives** sans augmentation de BI)

ÉTAPE 2 (AJUSTEMENT DES VARIABLES x_j)

$$x_j^k = \begin{cases} 1, & \text{si } \bar{c}_j \leq 0 \\ & \\ & , j \in N. \\ & \\ 0, & \text{sinon} \end{cases}$$

ÉTAPE 3 (CALCUL DU LAGRANGIEN)

- $LAG^k = CX^k + U^k e - U^k A X^k$.
- Si $LAG^k > BI$, alors $BI = LAG^k$ (mise à jour de la borne inférieure), $SEQUENCE = 0$ et $MAXSEQ = 0$;
  sinon, $SEQUENCE = SEQUENCE + 1$.
- Si $SEQUENCE = 10$, alors $MAXSEQ = MAXSEQ + 1$ et $SEQUENCE = 0$.

– Si $MAXSEQ = 3$ alors **TERMINÉ**. (critère d'arrêt)

ÉTAPE 4 (CALCUL DU SOUS-GRADIENT ET DE LA DIRECTION PONDÉRÉE)

– $d_k = e - AX_k$, direction de croissance de LAG.

– Si l'algorithme en est à son premier appel (noeud 0), alors la valeur de S^k est calculée en (1). Sinon, S^k est calculée en (2) :

– (1) $S^k = d^k + \beta_k S^{k-1}$, ([4, Camerini,Fratta et Maffioli], combinaison **convexe** du sous-gradient et la direction précédente)

où

$$\beta_k = \begin{cases} -(\alpha S^{k-1} \cdot d^k)/|S^{k-1}| & \text{,si } S^{k-1} \cdot d_k \text{ et } k \neq 0. \\ 0 & \text{sinon.} \end{cases}$$

et $\alpha = 1.5$

(2) $S^k = d^k$.

ÉTAPE 5 (CALCUL DU PAS)

– $t^k = \dfrac{\delta[1.1W - LAG^k]}{|S^k|}$

où, δ = 1.5 (coefficient de relaxation),

et W = Borne supérieure (globale).

ÉTAPE 6 (MISE À JOUR DES MULTIPLICATEURS)

- $U^{k+1} = U^k + t^k S^k$.

$$u^{k+1} = \begin{cases} \sup_i & , \text{si} \quad u_i^{k+1} > \sup_i \\ u_i^{k+1} & , \text{si} \quad u_i^{k+1} \in [0, \sup_i], \quad i \in M \\ 0 & , \text{si} \quad u_i^{k+1} < 0 \end{cases}$$

- $k = k + 1$.

- aller à l'étape 2 .

Nous avons remarqué, sur les problèmes testés (chapitre 6), que l'algorithme de Camerini, Fratta et Maffioli [4] trouvait une meilleure borne inférieure de départ que l'algorithme classique. Par contre, les réoptimisations subséquentes nous ont semblées plus lentes par cette méthode qu'avec la méthode classique. Nous utilisons donc un algorithme de sous-gradient où S^k est une direction pondérée (1) lors de son application au noeud 0, et la direction classique (2) aux noeuds subséquents.

La qualité de la solution de départ est importante. En effet, nous avons observé sur les problèmes testés (chapitre 6) que si le nombre d'itérations requis au noeud 0 se situait en moyenne autour de 120 lors du premier appel, celui-ci dépassait rarement 40 lors des appels subséquents. Une bonne solution de départ permet donc de réduire considérablement le nombre d'itérations de l'algorithme de sous-gradient, et le temps de calcul par conséquent.

Une autre caractéristique importante de l'algorithme de sous-gradient décrit ci-haut, est le fait que la solution duale (U) obtenue ne respecte pas nécessairement les contraintes duales. Afin de pouvoir réduire le système, en éliminant des

variables par un calcul de **pénalités** (section 4.2.4), la solution duale utilisée se doit d'être **réalisable.** Une procédure de **PROJECTION** de U sur le domaine réalisable est alors utilisée. Le principe de cette procédure est de réduire tous les multiplicateurs, d'une contrainte duale **violée**, selon la même proportion (voir ÉTAPE 3 de l'algorithme ci-après):

ÉTAPE 1 (INITIALISATION)

- $i = 1$. (initialisation du numéro de ligne)
- $VUE = J_i$.
- $MIN = \bar{c}_k$, où $k = \min_{j \in J_i} j$.

ÉTAPE 2 (VÉRIFICATION DE LA FAISABILITÉ)

- si $VUE = \emptyset$, aller à l'étape 4 .
- $DELTA = \bar{c}_k$, où $k = \min_{j \in \{J_i \setminus VUE\}} \{j\}$.
- $MIN = \min\{DELTA, MIN\}$.
- $VUE = VUE \setminus \{k\}$.
- si $DELTA < 0$, aller à l'étape 3;
- retour à l'étape 2 .

ÉTAPE 3 (RÉDUCTION DES MULTIPLICATEURS)

- $u_l = \dfrac{u_l c_k}{(c_k - \bar{c}_k)}, l \in I_k$. (Réduction proportionnelle à la violation du coût réduit)
- $\bar{c}_j = \bar{c}_j + \dfrac{u_l(-\bar{c}_k)}{(c_k - \bar{c}_k)}, j \in J_l, l \in I_k$. (Mise à jour des coûts réduits)

- aller à l'étape 2.

ÉTAPE 4 (SATURATION DES LIGNES)

- Si $MIN \leq 0$, aller à l'étape 5 .
- $u_i = u_i + MIN$.
- $\bar{c}_j = \bar{c}_j - MIN, j \in J_i$.

ÉTAPE 5 (CRITÈRE D'ARRÊT)

- si $i = |M|$, alors **TERMINÉ.**
- $i = i + 1$.
- $VUE = J_i$.
- aller à l'étape 2 .

Notons que cette procédure de projection est efficace. En effet, sur les problèmes traités (chapitre 6), nous avons observé que si l'objectif dual diminuait, c'était généralement d'une valeur inférieure à 0.1 % .

4.2.4 Pénalités

Propriété 4.1

Soit U, une solution duale réalisable et $\bar{c}_j = c_j - \sum_{i \in I_j} u_i$, le coût réduit de la variable j ($j \in N$). Si $\bar{c}_j > (UB - LB)$, alors la variable x_j peut être fixée à 0.

Preuve:

Si x_j est fixée à 1, alors toute ligne $i \in I_j$ est éliminée et le vecteur U est toujours réalisable. La borne inférieure corrrespond donc à $\bar{c}_j + \sum_{i \in M} u_i$.

Un calcul de pénalités est effectué sur les variables ayant un coût réduit positif. Cette procédure entre en action dans le cas où (BS-BI) est inférieur à la valeur du plus grand coût ($\max_{j \in N} c_j$). L'intérêt de cette procédure est la possibilité de réduire considérablement la taille du sous-problème (matrice des contraintes) lorsque l'écart entre les bornes inférieure et supérieure est mince.

4.2.5 Algorithme primal saturation-relaxation

L'idée générale de l'algorithme **SATURATION-RELAXATION** est basée sur la diminution de l'écart entre les objectifs dual et primal [9, Fisher et Kedia]. Soit **SOL** défini comme l'ensemble des variables **en base** ($x_j = 1$). Toute nouvelle variable x_j, entrant en base, augmente cet écart de $\bar{c}_j + \sum_{i \in I_j \setminus LIBRE} u_i$ (où $LIBRE = \{i \in M : \sum_{j \in J_i \cap SOL} a_{ij} > 1\}$). IL s'agit de trouver un ensemble SOL recouvrant toutes les lignes, puis d'éliminer par la suite les indices des colonnes qui augmentent le coût de la solution, jusqu'à ce qu'il ne reste aucune ligne surrecouverte. Ce processus de saturation-relaxation, inspiré de Balas et Ho [1], est effectué 3 fois.

ÉTAPE 1 (INITIALISATION)

- $\bar{c}_j = c_j - \sum_{i \in I_j} u_i, j \in N$. (calcul des coûts réduits)

- $\bar{c}_j = \max\{\theta, \bar{c}_j\}, j \in N$ et $\theta << 1$.
- $SOL = \emptyset$, ensemble des indices j pour lesquelles $x_j = 1$.
- $LIBRE = M$, ensemble des contraintes non-saturées.
- $ITER = 1$. (numéro d'itération)

ÉTAPE 2 (CHOIX DE LA NOUVELLE VARIABLE QUI ENTRE EN BASE)

- $SOL = SOL \cup \{k\}$, où

$$\frac{\bar{c}_k}{|I_k \cap LIBRE|} = \min_{j \in \backslash SOL} \frac{\bar{c}_j}{|I_k \cap LIBRE|}$$

- $\bar{c}_j = \bar{c}_j + u_i, j \in J_i, i \in I_k \cap LIBRE$.

ÉTAPE 3 (MISE À JOUR DE L'ENSEMBLE DES LIGNES RECOUVERTES)

- $LIBRE = LIBRE \backslash I_k$.
- si $LIBRE \neq \emptyset$, alors aller à l'étape 2;
- $SOL = SOL \backslash \{k\}$, où $\{k \in SOL : \sum_{j \in J_i \cap SOL} > 1, i \in I_k\}$.
- si $ITER = 3$, alors **TERMINÉ**.
- $ITER = ITER + 1$.

ÉTAPE 4 (INITIALISATION DE LA PARTIE RELACHEMENT)

- $SURSAT = \{i \in M : \sum_{j \in J_i \cap SOL} > 1\}$.
- si $SURSAT = \emptyset$, alors **TERMINÉ**.
- $\bar{c}_j = c_j - \sum_{i \in I_j \backslash SURSAT} u_i, j \in SOL$.

ÉTAPE 5 (CHOIX DE LA VARIABLE DE BASE À ÉLIMINER)

- $SOL = SOL \backslash \{k\}$, où $\bar{c}_k = \max_{j \in SOL} \bar{c}_j$
- $SAT = \{i \in I_k : \sum_{j \in J_i \cap SOL} = 1\}$.

ÉTAPE 6 (MISE À JOUR DE L'ENSEMBLE DES LIGNES SUR-RECOUVERTES

- $SURSAT = SURSAT \backslash SAT$. (Ensemble des lignes sur-recouvertes)
- si $SURSAT = \emptyset$, alors aller à l'étape 2.
- $\bar{c}_j = \bar{c}_j - \sum_{i \in SAT} u_i, j \in SOL$.

Lorsque l'exécution est terminée, la valeur $BS = \sum_{j \in SOL} c_j$ constitue une borne **supérieure**, puisque SOL recouvre toutes les lignes. La première exécution (au noeud 0) de l'algorithme, précède l'appel à la méthode du sous-gradient, et une seule séquence des étapes 1–3 est alors effectuée, avec $u_i = 0, i \in M$, ceci correspond alors à une séquence de **saturation** de l'algorithme PRIMAL1 de Balas et Ho [1].

Sur un ensemble de 40 problèmes générés aléatoirement (chapitre 6), nous avons appliqué au noeud 0 les algorithmes PRIMAL1 et PRIMAL5 de Balas et Ho [1], ainsi que l'algorithme SATURATION-RELAXATION. Dans 95 % des cas, l'algorithme SATURATION-RELAXATION a trouvé une meilleure borne supérieure et celle-ci était en moyenne de 1.21 % inférieure au minimum du résultat des deux autres.

4.2.6 Sélection et branchement

La méthode de SELECTION du sous-problème à traiter, ainsi que la procédure de BRANCHEMENT utilisée, seront présentées en détail au chapitre 5. Cette partie a été exclue du chapitre 4 afin de bien identifier les composantes de la **méthode d'énumération implicite** qui sont analysées dans ce mémoire.

Chapitre 5

Le branchement

Ce chapitre porte sur la stratégie de branchement utilisée lors de la création de nouveaux sous-problèmes, durant l'exploration de l'arbre d'**énumération implicite**. Des notions relatives à la **structure** de la matrice des contraintes seront d'abord présentées. Puis, une description brève des **critères, stratégies** et **types** de branchement, expérimentés sur les problèmes tests (chapitre 6), sera donnée par la suite.

5.1 Matrices cycliques

Une matrice carrée $B_{(0,1)}$, ayant des éléments 0 ou 1, est **cyclique** si elle ne contient aucune ligne ou colonne identiques, et telle que la somme des éléments non-nuls sur chaque ligne et chaque colonne est égale à 2 (figure 5.1) .

Figure 5.1: Matrice cyclique de degré 3, où les lignes sont associées aux indices des noeuds du réseau.

Dans un réseau, dont les noeuds correspondent aux lignes et les arcs aux colonnes, ce type de matrice définit un ou plusieurs cycles (figure 5.1), et un cycle **impair** a pour effet de produire un saut de dualité (solution fractionnaire) lors de la résolution du problème $\{\min CX : BX \geq e, x_j \text{ binaire}\}$, pour certaines valeurs de C (exemple 5.1).

$$ZP = \min 5x_1 + 6x_2 + 7x_3 \tag{5.1}$$

Sous les contraintes:

$$x_1 + x_2 \geq 1 \tag{5.2}$$

$$x_1 + x_3 \geq 1 \tag{5.3}$$

$$x_2 + x_3 \geq 1 \tag{5.4}$$

$$x_j \in \{0,1\}, j = 1,2,3. \tag{5.5}$$

SOLUTION : $x_1^* = x_2^* = 1, x_3^* = 0; ZP = 11.$

SOLUTION du PL : $x_1^* = x_2^* = x_3^* = 0.5; ZPL = 18/2 = 9.$

Exemple 5.1 Problème de recouvrement dont la matrice des contraintes est cyclique et de degré impair.

5.2 Matrice balancée

Une matrice $A_{(0,1)}$ est **balancée** si elle ne contient aucune sous-matrice cyclique (SMC) de degré impair [3, Berge]. Un problème de recouvrement, dont la matrice

des contraintes fait partie de cette classe, ne possède pas de saut de dualité [3, Berge].

Toute sous-matrice cyclique de degré impair contenue dans une matrice non-balancée ne provoque pas nécessairement un saut de dualité. Les propriétés suivantes apportent plus de précision quant à leur degré d'influence :

Propriété 5.1

Soit U, la solution optimale du PD. Si $A_{(0,1)}$ ne contient aucune **sous-matrice cyclique** de degré impair dont **chacune des colonnes** a un coût réduit ($\bar{c}_j = c_j - \sum_{i \in I_j} u_i$) nul, alors $ZP = ZD$.

Preuve

Soit $B_{(0,1)}$ la sous-matrice de A composée des colonnes de A ayant un coût réduit nul.

Soient PR_1 et PR_2 les PR dont les matrices de contraintes sont respectivement A et B. Les PD associés sont PD_1 et PD_2.

Puisque l'élimination de colonnes correspond à la fixation à 0 des variables libres associées à celles-ci dans $PR1$, alors

$$ZP_2 \geq ZP_1 \tag{5.6}$$

Comme PD_1 contient toutes les contraintes duales saturées (dont le coût réduit

est nul) de PD_2 , alors

$$ZD_1 = ZD_2 \tag{5.7}$$

Si B est balancée, le saut de dualité de PR_2 est nul [voir 16, Nemhauser] :

$$ZP_2 = ZD_2 \tag{5.8}$$

à partir des relations (5.7) et (5.8), on a que:

$$ZD_1 = ZP_2 \tag{5.9}$$

et (5.6) dans (5.9) implique:

$$ZD_1 \geq ZP_1 \tag{5.10}$$

puisque $ZD_1 \leq ZP_1$ (propriété 3.1), on trouve finalement que:

$$ZP_1 = ZD_1 \tag{5.11}$$

Propriété 5.2

Soit un problème de recouvrement dont la matrice des contraintes $A_{(0,1)}$ est cyclique de degré k impair ne contenant aucune autre sous-matrice cyclique de degré inférieur à k. Le saut de dualité de ce problème est égal à la valeur du plus petit multiplicateur $u_i (i \in M)$, où le vecteur U est la solution optimale du problème dual.

Preuve

Pour toute matrice carrée cyclique de degré k, ne contenant aucune autre sous-matrice cyclique de degré inférieur, il existe une permutation des lignes et

contraintes, de telle sorte que la ligne l devienne la ligne 1 et que la matrice soit sous la forme suivante:

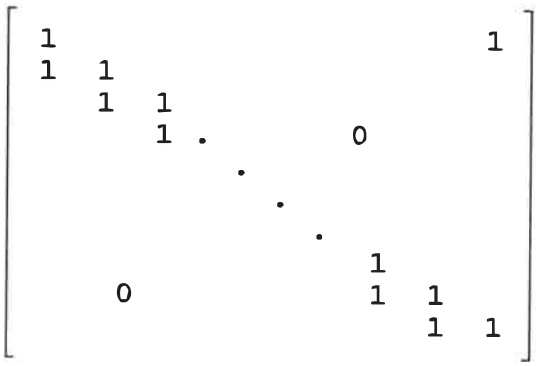

Figure 5.2: Matrice carrée cyclique de degré k, après la permutation.

Soit $i = 1$, la ligne associée au multiplicateur de valeur minimum (après permutation des lignes et des colonnes), alors il existe un ensemble H tel que en posant $x_j = 1$ si $j \in H$ et 0 sinon, on obtient:

$$\sum_{j \in J_i} x_j = \begin{cases} 2, & \text{si } i = 1 \\ 1, & \text{sinon} \end{cases} \tag{5.12}$$

où H = $\{ j \in N : j$ est impair $\}$.

S'il existe un saut de dualité, alors $\bar{c}_j$ est nul pour tout $j \in N$ (propriété 5.1). On peut donc exprimer le coût des variables $x_j, j \in H$, en fonction des multiplicateurs u_i si la matrice des contraintes est sous la forme présentée à la figure 5.3 :

- $c_1 = u_1 + u_2$
- $c_3 = u_3 + u_4$
- $c_5 = u_5 + u_6$
- $c_7 = u_7 + u_8$
- .
- .
- .
- $c_{n-2} = u_{n-2} + u_{n-1}$
- $c_n = u_1 + u_n$

L'ensemble H constitue une solution réalisable du PR, puisque:

$$\sum_{j \in H} a_i j = \begin{cases} 2 & ,\text{si} \quad i = 1, \\ 1 & ,\text{si} \quad i \neq 1, \quad i \in M \end{cases}$$

Puisque la valeur de l'objectif dual $ZD = \sum_{i \in M} u_i$ et que la valeur de l'objectif primal $ZP = \sum_{j \in H} c_j$, alors :

$$\begin{aligned} ZP &= \sum_{j \in H} c_j \\ &= c_1 + c_3 + c_5 + ... + c_{n-2} + c_n \\ &= u_1 + u_2 + u_3 + u_4 + ... + u_{n-2} + u_{n-1} + u_n + u_1 \\ &= \sum_{i \in M} u_i + u_1 \\ &= ZD + u_1. \end{aligned} \tag{5.13}$$

Matrice totalement balancée

Une matrice $A_{(0,1)}$ est **totalement balancée** si elle ne contient aucune sous-matrice cyclique [voir 16, Nemhauser et Wolsey]. Les matrices totalement balancées forment donc une **sous-classe** des matrices balancées.

5.3 Matrice gloutonne

Une matrice carrée (2×2) de la forme

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

est appelée **matrice gloutonne.**

Un problème de recouvrement, dont la matrice des contraintes ne contient aucune sous-matrice de ce type, ne possède pas de saut de dualité et peut être résolu à l'aide d'un algorithme glouton très simple [13, Hoffman, Kolen et Sakorovitch].

Propriété 5.3

Une matrice $A_{(0,1)}$ **totalement balancée,** dont les lignes et les colonnes sont placées en ordre doublement lexicographique[1] , ne contient aucune sous-matrice gloutonne [13, Hoffman, Kolen et Sakorovitch].

Notre branchement a pour but d'éliminer des sous-matrices cycliques qui créent un saut de dualité, afin de réduire plus rapidement l'écart entre les bornes inférieure et supérieure. Pour ce faire, nous éliminerons des sous-matrices gloutonnes, en exploitant les deux propriétés suivantes, qui les relient aux sous-matrices cycliques produisant un saut de dualité, énonçant ainsi des conditions nécessaires sur la présence de ces dernières :

Propriété 5.4

Toute sous-matrice cyclique de degré impair contient au moins une sous-matrice gloutonne.

[1] Ordre alphabétique sur les lignes et les colonnes.

Preuve

Si la matrice $B_{(0,1)}$ ne contient aucune sous-matrice gloutonne, elle est alors totalement balancée (propriété 5.3), donc **balancée**. De ce fait, B ne peut contenir de sous-matrice cyclique.

Propriété 5.5

Si un problème de recouvrement possède un saut de dualité, alors la matrice $A_{(0,1)}$ des contraintes contient au moins une sous-matrice gloutonne

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

telle que $\bar{c}_{j_1}$ et $\bar{c}_{j_2}$ sont nuls ($\bar{c}_j = c_j - \sum_{i \in I_j} u_i$, où U = solution optimale du PD).

Preuve

Directement prouvé par les propriétés 5.1 et 5.3.

5.4 Sous-matrice interdite

Une sous-matrice gloutonne

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$
$$j_1 \quad j_2$$

telle que $\bar{c}_{j_1}$ et $\bar{c}_{j_2}$ sont égaux à 0, est appelée sous-matrice interdite (SMI).

Puisque le vecteur U est obtenu par un algorithme de **sous-gradient**, il ne correspond pas à la solution optimale du PD, mais plutôt à un bon estimé de celle-ci. C'est donc une **approximation** de la propriété 5.5 qui sera utilisée, pour déterminer les sous-matrices gloutonnes qui sont **interdites**.

5.5 Le branchement

Notre procédure de branchement est orientée vers l'élimination progressive des sous-matrices cycliques susceptibles de créer un saut de dualité. Pour ce faire, nous utiliserons les conditions nécessaires énumérées précédement (section 5.4), c'est-à-dire que nous chercherons à éliminer les sous-matrices **interdites**.

5.5.1 Recherche d'une sous-matrice interdite

La procédure utilisée pour trouver une SMI est la suivante:

ÉTAPE 1 (INITIALISATION)

- $SOL = \{j \in J : \bar{c}_j = 0\}$.
- $k_1 = 0$.

ÉTAPE 2 (CHOIX DE LA PREMIÈRE COLONNE)

- $T_{k_1} = \{j \in SOL : j > k_1\}$.
- Si $T_{k_1} = \emptyset$, alors FIN.
- $k_1 = \min j, j \in T_{k_1}$.
- $k_2 = k_1$.
- $i_1 = 0$.

ÉTAPE 3 (CHOIX DE LA PREMIÈRE LIGNE)

- $T_{i_1} = \{i \in I_{k_1} : i > i_1\}$.
- si $T_{i_1} = \emptyset$, alors aller à l'étape 2.
- $i_1 = \min i, i \in T_{i_1}$.
- $i_2 = i_1$.

ÉTAPE 4 (CHOIX DE LA SECONDE LIGNE)

- $T_{i_2} = \{i \in I_{k1} : i > i_2\}$.
- si $T_{i_2} = \emptyset$, alors aller à l'étape 3.
- $i_2 = \min i, i \in T_{i_2}$.

ÉTAPE 5 (CHOIX DE LA SECONDE COLONNE)

- $T_{k_2} = \{j \in SOL : j > k_2\}$.
- si $T_{k_2} = \emptyset$, alors aller à l'étape 2.
- $k_2 = \min j, \{j \in SOL : j > k_2\}$.

– si $i_1 \in I_{k_2}$ et $i_2 \notin I_{k_2}$, alors ARRÊT. Sinon, recommencer l'étape 5.

Une SMI définie par les lignes i_1, i_2 et les colonnes k_1, k_2 est trouvée lors de l'arrêt à l'étape 5.

5.5.2 Stratégies de branchement

Plusieurs SMI peuvent être contenues dans la matrice des contraintes et le degré d'influence de chacune d'elles sur le saut de dualité est différent.

Trois stratégies distinctes, définies dans le but de sélectionner une SMI intéressante (possédant un haut degré d'influence), seront étudiées.

Soit T = { ensemble des SMI contenues dans A},

et S = $\{(l_1, l_2) : l_1 \in I, l_2 \in I$ et il existe une SMI dont les lignes sont l_1 et $l_2\}$.

STRATÉGIE 1

Choisir $SMI_k \in T$, telle que

$$\min\{u_{k_1}, u_{k_2}\} = \max_{(l_1,l_2)\in S}(\min\{u_{l_1}, u_{l_2}\})$$

Se basant sur la propriété 5.2, nous émettons l'hypothèse que l'effet maximum possible d'une sous-matrice cyclique de degré impair (contenue dans la matrice des contraintes) sur le saut de dualité, correspond à la valeur du plus petit multiplicateur de ses lignes. Puisque toute sous-matrice cyclique de degré impair contient au

moins une sous-matrice gloutonne (propriété 5.4), son effet maximum possible sur le saut de dualité n'est pas plus grand que la valeur du plus petit multiplicateur des 2 lignes de la sous-matrice gloutonne.

On cherche donc à éliminer une sous-matrice cyclique ayant beaucoup d'effet, pour ainsi réduire plus rapidement le saut de dualité.

STRATÉGIE 2

Soient les paramètres suivants:

- BASE, l'ensemble des variables de base qui définissent BS, la borne supérieure.
- SUR $= \{i \in M : \sum_{j \in BASE} x_j > 1\}$, ensemble des ligne surrecouvertes.
- $BSUR_i = \{j \in N : x_j = 1\}, i \in SUR.$
- i_{l_1}, première ligne de $SMI_l \in T$.
- i_{l_2}, deuxième ligne de $SMI_l \in T$.
- j_{l_1}, première colonne de $SMI_l \in T$.
- j_{l_2}, deuxième colonne de $SMI_l \in T$.

Cette stratégie consiste à choisir $SMI_k \in T$, respectant les conditions suivantes:

1. $i_{k_1} \in SUR.$

2. $j_{k_1} \in BSUR_{i_{k_1}}$ et $j_{k_2} \in BSUR_{i_{k_1}}$.

3. $u_{i_{k_1}} = \max\{u_{i_{l_1}}\}$, où $SMI_l \in T$ respecte les conditions (1) et (2) .

L'écart entre BI et BS est principalement causé par le surrecouvrement des lignes dont le multiplicateur est non-nul (propriété 3.1). On cherche donc à éliminer une SMI qui affecte cet écart, par les conditions (1) et (2). La condition (3) permet d'éliminer la SMI_k dont les colonnes j_{k_1} et j_{k_2} affecte le **plus** l'écart, en surrecouvrant la ligne i_{k_1}.

Remarque: Si il n'existe aucune SMI respectant ces conditions, la stratégie 1 est alors appliquée.

STRATÉGIE 3

Choisir $SMI_k \in T$, qui élimine le plus de sous-matrices gloutonnes lors du branchement (somme sur les deux branches).

Cette dernière stratégie vise le rapprochement rapide vers la forme d'un PR pouvant être résolu à l'aide d'un algorithme glouton (section 5.4).

5.5.3 Critère de branchement

Soit SMI_k, la sous-matrice sélectionnée (figure 5.4).

Le critère de branchement exploité lors la résolution des problèmes (chapitre 6) est le suivant:

$$\begin{array}{c} \\ k_1 \\ k_2 \end{array} \begin{array}{c} \begin{array}{cc} j_1 & j_2 \end{array} \\ \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \end{array}$$

Figure 5.3: Sous-matrice interdite, définie par les lignes k_1, k_2 et les colonnes j_1, j_2.

Addition d'une contrainte, sur les variables incluses dans l'intersection des lignes k_1 et k_2.

BRANCHE 1:

$$\sum_{j \in I_{k_1} \cap I_{k_2}} x_j \leq 0$$

Élimination des colonnes $j (x_j = 0) \in I_{k_1} \cap I_{k_2}$.

BRANCHE 2:

$$\sum_{j \in I_{k_1} \cap I_{k_2}} x_j \geq 1$$

Addition d'une coupe et élimination des contraintes I_{k_1} et I_{k_2} alors redondantes.

L'application de ce critère comporte l'**avantage** de ne générer aucune **nouvelle** sous-matrice gloutonne dans chacune des deux branches.

Remarque: Les sous-problèmes générés, en appliquant ce critère, possèdent toujours la structure d'un PR, et ceci est essentiel pour l'application des heuristiques.

5.5.4 Types de branchement

Notre procédure d'énumération implicite sera executée selon deux types de branchement, soit:

1. **MEILLEUR D'ABORD** (MA):

   Consiste à traiter le noeud dont la borne inférieure est la plus basse (figure 5.4).

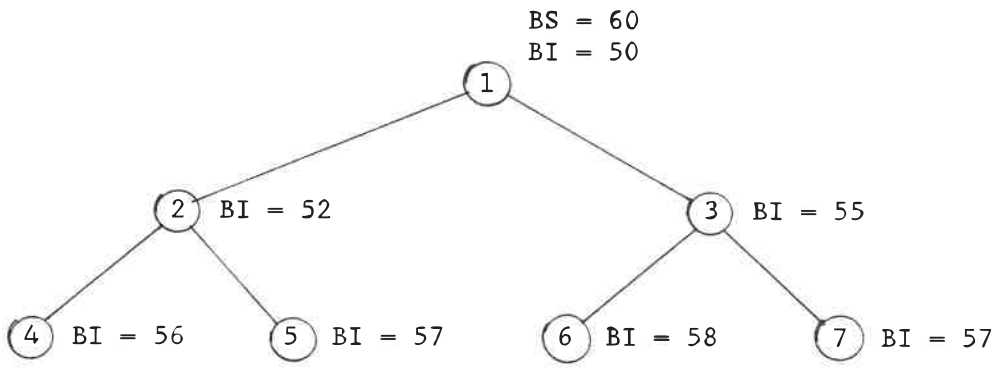

Figure 5.4: Arbre de branchement selon le type MA, où 4 est le prochain noeud à traiter.

2. **PROFONDEUR D'ABORD** (PA):

   Consiste à traiter le fils, du dernier noeud traité, dont la valeur de BI est la plus basse (figure 5.5).

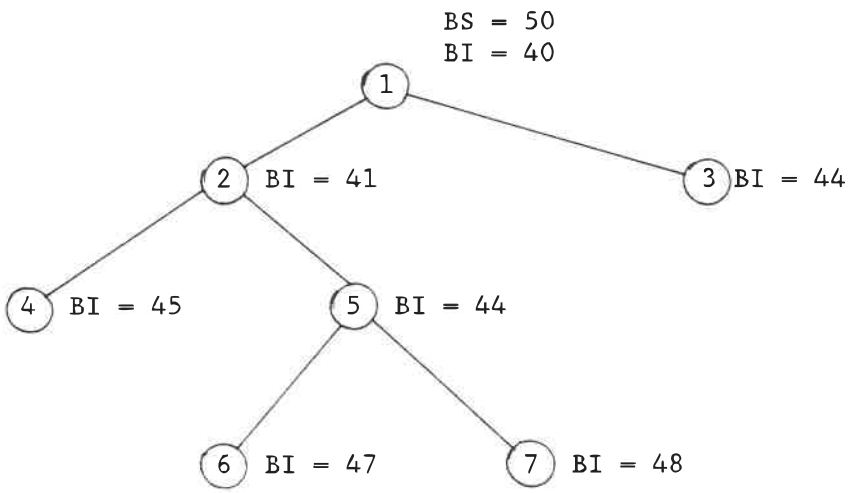

Figure 5.5: Arbre de branchement selon le type PA, où 6 est le prochain noeud à traiter.

Un **TAS** (HEAP) est utilisée pour sélectionner le prochain **noeud** à traiter. Soient BI_i, la borne inférieure calculée au noeud i et N_i, la **profondeur** (nombre de branches entre le noeud i et le noeud initial) de celui-ci. La valeur associée à chaque étiquette du TAS correspond à BI_i pour MA et à $(-N_i + 1/BI_i$, où $BI_i \geq 1$ pour PA.

Chapitre 6

Analyse des Résultats

Ce chapitre est consacré à l'analyse de notre méthode d'énumération implicite, appliquée sur une série de problèmes tests. Nous désirons d'une part évaluer l'efficacité générale de notre méthode, et d'autre part, analyser l'effet du changement dans la structure du branchement, sur le temps de calcul et le nombre de noeuds générés. Cette seconde partie constitue l'objet principal de ce chapitre.

6.1 Nature des problèmes

Notre **méthode d'énumération implicite** est expérimentée sur divers problèmes de structures différentes et de densité comparable à celle que possèdent les problèmes réels, donc réalistes. Pour ce faire, nous avons utilisé un générateur de problèmes **aléatoires** ayant les propriétés suivantes:

- Chaque colonne contient au moins un élément non-nul, c'est-à-dire que $\sum_{i \in I_j} a_{ij} \geq 1, j \in N$.
- Chaque ligne contient **au moins deux** éléments non-nuls, c'est-à-dire que $\sum_{j \in J_i} a_{ij} \geq 2, i \in M$. Cette caractéristique élimine la présence de fixation triviale d'une variable au noeud initial, diminuant les chances de traiter un problème trop facile.
- Le coût de chaque variable est entier et contenu dans l'intervalle [1,10000]. Lors de la résolution, ce coût est divisé par 100, et est considéré comme un réel.

Nous avons alors généré une série de 60 problèmes divisée en 4 classes différentes, soient:

1. Problèmes de 500 variables, 200 contraintes et de densité 2% .
2. Problèmes de 1000 variables, 200 contraintes et de densité 2% .
3. Problèmes de 500 variables, 200 contraintes et de densité 5% .
4. Problèmes de 1000 variables, 200 contraintes et de densité 5% .

Puisque le but principal de ce chapitre est de comparer plusieurs critères de branchement, chaque classe comporte **au moins dix** problèmes possédant un saut de dualité **non-nul**. Pour vérifier la présence d'un saut de dualité, nous avons résolu la relaxation linéaire de ces problèmes en utilisant le logiciel XMP. Afin de s'assurer de cette situation, nous avons utilisé la méthode suivante lors de la génération de chaque classe:

- Si on obtient 5 problèmes sans saut de dualité (avant d'atteindre le total de 15 probl.), on poursuit la génération en éliminant les problèmes sans saut de dualité.

Cette contrainte sur le nombre de problèmes possédant un saut de dualité non-nul permet de comparer plus efficacement les critères de branchement. En effet, la résolution de ces problèmes est plus difficile et exige la génération de plusieurs noeuds. Le minimum imposé de 10 problèmes semble statistiquement raisonnable, pour que les résultats soient représentatifs, et pour qu'on puisse établir une bonne analyse comparative.

6.2 Présentation des résultats

Les résultats ont été divisés suivant leur classe. Ainsi, les tableaux 6.1 à 6.4 se réfèrent respectivement aux classes 1 à 4, et contiennent les valeurs numériques des bornes **inférieure** et **supérieure** au noeud 0, ainsi que l'objectif des solutions **continue** et **entière** des problèmes.

On retrouve aux tableaux 6.5 à 6.8, les informations suivantes concernant les écarts intéressants (en %):

1. Ecart entre les bornes inférieure et supérieure au noeud 0, soit BS-BI(0).
2. Ecart entre l'objectif de la solution entière et la borne inférieure au noeud 0, soit ZP-BI(0).
3. Ecart entre la borne supérieure au noeud 0 et l'objectif de la solution entière, soit BS(0)-ZP.
4. Saut de dualité du problème, soit ZP-ZPL.

Au bas des tableaux 6.5 à 6.8, deux types de moyennes sont présentées, soient:

- **MOY 1:**

  Moyenne des valeurs sur **tous** les problèmes de la colonne de ce tableau.

- **MOY 2:**

  Moyenne des valeurs sur les problèmes de la colonne de ce tableau possédant un **saut de dualité non-nul.**

Problème	BI(0)	BS(0)	ZPL	ZP
1	92245	92669	92463	92669
2	114778	116969	115051	116800
3	102259	102873	102432	102432
4	78551	78551	78551	78551
5	92594	92023	92740	93259
6	90567	92718	90747	90950
7	101687	103022	101969	102454
8	99873	100327	100158	100325
9	95809	95809	95809	95809
10	96208	97415	96551	97415
11	101577	102352	102158	102158
12	100725	103469	101419	101880
13	89641	91770	89803	90689
14	97514	98219	97542	97542
15	97082	97903	97306	97516

Tableau 6.1: Valeur des bornes au noeud 0, ainsi que de ZPL et ZP, pour les problèmes (200×500) de densité 2%.

Problème	BI(0)	BS(0)	ZPL	ZP
16	49783	49783	49783	49783
17	43132	43364	43315	43315
18	42980	44412	43113	43512
19	49320	49320	49320	49320
20	48742	49468	48804	49018
21	48379	48379	48379	48379
22	45358	45358	45358	45358
23	54567	56104	54721	54837
24	56780	58683	56926	57179
25	46962	47909	47015	47306
26	39682	40569	39764	39921
27	57424	58052	57570	57766
28	46362	47484	46407	46926
29	41262	42191	41279	41468
30	50928	51323	50934	51128

Tableau 6.2: Valeur des bornes au noeud 0, ainsi que de ZPL et ZP, pour les problèmes (200×1000) de densité 2%.

Problème	BI(0)	BS(0)	ZPL	ZP
31	34147	39879	34560	37366
32	23004	24587	23507	24288
33	24302	24770	24385	24743
34	23608	23727	23727	23727
35	24771	24967	24967	24967
36	28236	30898	28421	29462
37	23783	26622	24136	25598
38	24263	26079	24536	25233
39	25951	29242	26322	28047
40	32573	34102	32755	33442
41	27057	31263	27312	28722
42	27422	29134	27691	28960
43	23564	24240	23734	24068
44	23050	24270	23246	23984
45	23027	25593	23286	24478

Tableau 6.3: Valeur des bornes au noeud 0, ainsi que de ZPL et ZP, pour les problèmes (200×500) de densité 5%.

Problème	BI(0)	BS(0)	ZPL	ZP
46	13092	13534	13203	13534
47	12586	13549	12742	13114
48	11276	11431	11421	11421
49	12919	13430	12954	13157
50	9806	10204	9856	10108
51	13331	13480	13399	13399
52	12431	12431	12431	12431
53	10406	10637	10541	10541
54	16076	16710	16152	16477
55	13764	14257	13951	14257
56	14030	14703	14129	14401
57	13329	13866	13416	13663
58	12708	13287	12766	13070
59	12646	13243	12794	12960
60	16257	17315	16458	16878

Tableau 6.4: Valeur des bornes au noeud 0, ainsi que de ZPL et ZP, pour les problèmes (200 $\times$ 1000) de densité 5%.

Problème	BS-BI(0) (%)	ZP-BI(0) (%)	BS(0)-ZP (%)	ZP-ZPL (%)
1	0.46	0.46	0	0.22
2	1.91	1.76	0.14	1.52
3	0.60	0.17	0.43	0
4	0	0	0	0
5	4.78	0.72	3.88	0.56
6	2.38	0.42	1.91	0.22
7	1.31	0.75	0.55	0.48
8	0.45	0.45	0	0.17
9	0	0	0	0
10	1.25	0.48	0.77	0.12
11	0.76	0.57	0.19	0
12	1.62	1.15	0.46	0.45
13	2.38	1.17	1.18	0.99
14	0.72	0.03	0.69	0
15	0.85	0.45	0.40	0.22
MOY 1	1.30	0.57	0.71	0.33
MOY 2	1.74	0.84	1.00	0.53

Tableau 6.5: Écarts entre les valeurs de BI (noeud 0), BS (noeud 0), ZPL et ZP pour les problèmes (200×500) de densité 2%.

Problème	BS-BI(0) (%)	ZP-BI(0) (%)	BS(0)-ZP (%)	ZP-ZPL (%)
16	0	0	0	0
17	0.54	0.42	0.11	0
18	3.33	1.19	2.08	0.87
19	0	0	0	0
20	1.49	0.57	0.91	0.44
21	0	0	0	0
22	0	0	0	0
23	2.82	0.49	2.26	0.21
24	3.35	0.70	2.56	0.44
25	2.02	0.73	1.26	0.62
26	2.23	0.60	1.60	0.39
27	1.09	0.60	0.49	0.34
28	2.42	1.22	1.18	1.12
29	2.25	0.50	1.71	0.46
30	0.78	0.39	0.38	0.38
MOY 1	1.49	0.49	0.97	0.35
MOY 2	2.18	0.70	1.44	0.53

Tableau 6.6: Écarts entre les valeurs de BI (noeud 0), BS (noeud 0), ZPL et ZP pour les problèmes (200×1000) de densité 2%.

Problème	BS-BI(0) (%)	ZP-BI(0) (%)	BS(0)-ZP (%)	ZP-ZPL (%)
31	16.79	9.43	6.30	8.12
32	6.88	5.58	1.22	3.32
33	1.93	1.81	0.11	1.47
34	0.50	0.50	0	0
35	0.79	0.79	0	0
36	9.43	4.34	4.65	3.66
37	11.94	7.63	3.85	6.06
38	7.48	4.00	3.24	2.84
39	12.68	8.08	4.09	6.55
40	4.69	2.67	1.94	2.10
41	15.54	6.15	8.13	5.16
42	6.24	5.61	0.60	4.58
43	2.87	2.14	0.71	1.41
44	5.29	4.05	1.18	3.17
45	11.14	6.30	4.36	5.12
MOY 1	7.61	4.61	2.69	3.57
MOY 2	8.68	5.22	3.10	4.12

Tableau 6.7: Écarts entre les valeurs de BI (noeud 0), BS (noeud 0), ZPL et ZP pour les problèmes (200 $\times$ 500) de densité 5%.

Problème	BS-BI(0) (%)	ZP-BI(0) (%)	BS(0)-ZP (%)	ZP-ZPL (%)
46	3.38	3.38	0	2.51
47	7.65	4.20	3.21	2.92
48	1.37	1.29	0.09	0
49	3.96	1.84	2.03	1.57
50	4.06	3.08	0.94	2.56
51	1.12	0.51	0.60	0
52	0	0	0	0
53	2.22	1.30	0.90	0
54	3.94	2.49	1.39	2.01
55	3.58	3.58	0	2.19
56	4.80	2.64	2.05	1.93
57	4.03	2.51	1.46	1.84
58	4.56	2.85	1.63	2.38
59	4.72	2.48	2.14	1.30
60	6.49	3.82	2.51	2.55
MOY 1	3.72	2.40	1.58	1.26
MOY 2	4.65	2.99	2.02	1.73

Tableau 6.8: Écarts entre les valeurs de BI (noeud 0), BS (noeud 0), ZPL et ZP pour les problèmes (200× 1000) de densité 5%.

6.2.1 Description des critères de branchement

Les tableaux 6.9 à 6.12 représentent les résultats comparatifs des différentes stratégies employées dans notre méthode d'énumération implicite. Deux types de résultats sont alors présentés, soient le temps total d'éxecution (en sec. sur un SUN 3/60) et le nombre de noeuds générés (excluant le noeud 0). On appellera **critère de branchement**, une combinaison du type de branchement et de la stratégie utilisés.

Ainsi, les critères sont identifiées à l'aide de deux indices:

$CRIT_{ij}$, où i indique le type de branchement (sélection du sous-problème)
(1: meilleur d'abord, 2: profondeur d'abord);
et j indique la stratégie utilisée (voir section 5.6.2).

Au bas des tableaux 6.9 à 6.12, deux types de moyennes sont présentées, soient:

- **MOY 1:**
  Moyenne des valeurs sur **tous** les problèmes de la colonne.
- **MOY 2:**
  Moyenne des valeurs sur les problèmes de la colonne possédant un **saut de dualité non-nul**.

Problème	$CRIT_{11}$		$CRIT_{21}$		$CRIT_{12}$		$CRIT_{22}$		$CRIT_{13}$		$CRIT_{23}$	
	t(s)	nbr.	t(s)	nbr.	t(s)	nbr.	t(s)	nbr.	t(s)	nbr.	t(s)	nbr.
1	52	4	55	4	67	6	67	6	110	4	115	4
2	510	92	568	112	399	79	397	78	543	66	570	70
3	60	4	62	4	60	4	61	4	198	4	199	4
4	32	0	32	0	32	0	32	0	32	0	32	0
5	205	22	198	22	199	22	191	20	361	8	361	8
6	90	4	92	4	197	4	94	4	245	4	246	4
7	92	8	99	12	92	8	98	12	284	16	288	18
8	62	4	63	4	62	4	62	4	117	4	117	4
9	36	0	36	0	36	0	36	0	36	0	36	0
10	90	10	92	10	98	10	98	10	266	10	266	10
11	88	6	84	6	77	2	77	2	204	6	211	6
12	128	10	120	10	85	4	86	4	250	16	273	18
13	188	22	165	20	243	34	378	90	387	32	381	28
14	82	4	77	4	77	4	80	4	227	6	228	6
15	81	6	75	6	70	6	71	6	174	8	178	8
MOY 1	**120**	13.1	121	14.6	**120**	12.5	122	16.3	229	12.3	233	12.5
MOY 2	**150**	18.2	153	20.4	151	17.7	154	23.4	274	16.8	280	17.2

Tableau 6.9: Temps de calcul et nombre total de noeuds traités, selon les différents critères, pour les problèmes (200 × 500) de densité 2%.

Problème	$CRIT_{11}$		$CRIT_{21}$		$CRIT_{12}$		$CRIT_{22}$		$CRIT_{13}$		$CRIT_{23}$	
	t(s)	nbr.	t(s)	nbr.	t(s)	nbr.	t(s)	nbr.	t(s)	nbr.	t(s)	nbr.
16	77	0	77	0	77	0	77	0	77	0	77	0
17	100	4	100	4	99	4	97	4	131	4	132	4
18	141	12	143	12	153	14	155	14	266	10	267	10
19	73	0	73	0	73	0	73	0	73	0	73	0
20	114	12	114	12	108	10	109	10	234	10	228	8
21	69	0	69	0	69	0	69	0	69	0	69	0
22	70	0	70	0	70	0	70	0	70	0	70	0
23	122	6	123	6	124	6	123	6	209	6	206	6
24	183	14	182	14	174	18	180	18	318	16	427	54
25	131	10	133	10	133	8	136	10	324	12	332	12
26	169	10	168	10	171	10	171	10	293	16	296	16
27	147	10	133	8	140	6	143	10	307	8	316	8
28	226	28	205	24	198	22	210	28	448	22	454	22
29	178	16	168	14	180	14	232	30	479	24	507	26
30	129	6	129	6	117	6	118	6	303	18	307	18
MOY 1	129	8.5	**126**	8.0	**126**	7.8	131	9.7	240	9.7	251	12.3
MOY 2	154	12.4	**150**	11.6	**150**	11.4	158	14.2	318	14.2	334	18.0

Tableau 6.10: Temps de calcul et nombre total de noeuds traités, selon les différents critères, pour les problèmes (200 $\times$ 1000) de densité 2%.

Probl.	$CRIT_{11}$		$CRIT_{21}$		$CRIT_{12}$		$CRIT_{22}$		$CRIT_{13}$		$CRIT_{23}$	
	t(s)	nbr.	t(s)	nbr.	t(s)	nbr.	t(s)	nbr.	t(s)	nbr.	t(s)	nbr.
31	7071	721	6335	841	4754	432	6672	876	11307	362	11929	434
32	253	20	247	20	226	16	256	30	830	34	884	42
33	205	28	206	28	178	20	175	20	530	32	514	33
34	92	2	96	2	91	2	94	2	129	2	127	2
35	97	8	97	8	97	8	97	8	268	4	269	4
36	878	156	919	157	575	82	750	100	1827	104	1878	112
37	3528	608	2724	480	894	140	894	140	3332	198	3074	192
38	461	48	720	84	597	50	557	54	1358	30	1637	44
39	4482	604	4551	602	4114	536	4693	689	13487	600	11644	536
40	436	62	440	66	544	77	1375	126	1077	30	1211	42
41	1424	134	2932	484	666	76	772	119	1836	62	1874	62
42	1283	162	1714	246	1574	226	1811	264	3672	194	3761	208
43	160	12	149	12	187	24	196	24	452	14	450	14
44	330	44	323	50	281	34	292	34	618	34	632	34
45	676	102	667	114	803	106	1625	242	1438	110	1448	112
MOY 1	1425	181	1475	220	1039	122	1351	182	2811	121	2755	125
MOY 2	1630	208	1687	245	1184	140	1544	209	3213	139	3149	144

Tableau 6.11: Temps de calcul et nombre total de noeuds traités, selon les différents critères, pour les problèmes (200 × 500) de densité 5%.

Problème	$CRIT_{11}$		$CRIT_{21}$		$CRIT_{12}$		$CRIT_{22}$		$CRIT_{13}$		$CRIT_{23}$	
	t(s)	nbr.	t(s)	nbr.	t(s)	nbr.	t(s)	nbr.	t(s)	nbr.	t(s)	nbr.
46	237	18	237	18	378	26	278	26	597	28	599	28
47	257	18	264	18	230	12	230	12	1079	42	1096	66
48	165	6	165	6	163	6	163	6	308	6	308	6
49	253	16	244	16	361	20	576	80	778	20	778	20
50	311	32	327	36	235	20	244	22	534	20	512	20
51	186	6	188	6	179	6	180	6	534	6	533	6
52	145	0	145	0	145	0	145	0	145	0	145	0
53	165	6	165	6	180	6	179	6	200	6	200	6
54	443	52	466	62	318	24	391	40	1767	58	2518	118
55	609	70	594	70	592	74	597	74	1374	42	1422	42
56	756	90	541	56	581	64	523	62	1354	50	1357	50
57	474	40	525	80	252	16	277	22	559	12	567	12
58	252	18	245	18	239	8	291	18	998	38	1105	48
59	283	26	302	40	257	16	351	48	635	22	640	22
60	1269	161	1627	252	770	76	764	100	1987	66	1881	74
MOY 1	387	37.3	402	45.6	**325**	24.9	346	34.8	857	27.7	911	34.5
MOY 2	468	49.2	488	60.5	**383**	32.3	411	45.8	1060	36.2	1134	45.5

Tableau 6.12: Temps de calcul et nombre total de noeuds traités, selon les différents critères, pour les problèmes (200 × 1000) de densité 5%.

6.3 Résultats généraux

Sur les problèmes possédant un saut de dualité nul, l'écart moyen obtenu entre BI(0) et BS(0) est de 0.48%, et le nombre moyen de noeuds générés dans la meilleure solution, de 2.61 noeuds, le pire cas étant de 8 noeuds. D'après ces résultats, notre algorithme réagit très bien face aux problèmes dont le saut de dualité est nul, c'est-à-dire les problèmes faciles.

On observe, aux tableaux 6.5 à 6.8, que l'écart entre la borne supérieure au noeud 0 et la solution optimale (entière), est élevé lorsque le saut de dualité est grand. Ceci s'explique probablement par le fait que notre heuristique primale utilise la solution duale U obtenue par l'algorithme de sous-gradient.

6.4 Comparaison des critères

D'après les résultats des tableaux 6.9 à 6.12 (surtout les deux derniers), on remarque tout d'abord qu'il est préférable de sélectionner un sous-problème suivant le type meilleur d'abord plutôt que profondeur d'abord. En effet, même si ce type de branchement exige plus d'espace mémoire en pratique, il permet de trouver la solution optimale plus rapidement qu'avec le second type, nous permettant ainsi de réduire le nombre de noeuds traités, ainsi que le temps de calcul.

En d'autres mots, si on complète la procédure d'énumération implicite, on observe que :

- Profondeur d'abord: demande moins de gestion.

- Meilleur d'abord: réduit le nombre de noeuds à traiter.

Et que si on arrête la procédure avant d'obtenir la solution optimale, on trouve:

- Profondeur d'abord: plus de solutions réalisables (Borne supérieure).
- Meilleur d'abord: meilleure borne inférieure.

Le tableau des problèmes de 200 contraintes, 500 variables et de densité 5% (tableau 6.11) regroupe les problèmes les plus difficiles des 4 classes. En effet, le saut de dualité moyen est de 4.12 (3.57 pour tous). Il s'agit donc d'une classe de problèmes très importante, et elle permet de mieux départager les différents critères.

Au niveau **temps de calcul**, le critère $CRIT_{12}$ s'avère le plus efficace. Comparable aux critères $CRIT_{11}$, $CRIT_{21}$ et $CRIT_{22}$ pour la résolution des problèmes des classes 1 (tableau 6.9) et 2 (tableau 6.10), il les surpasse tous pour la résolution des problèmes des classes 3 (tableau 6.11) et 4 (tableau 6.12). Les résultats révèlent en effet que l'utilisation du critère $CRIT_{12}$, plutôt qu'un des 5 autres critères, réduit le temps de calcul **moyen** d'au moins 30% (MOY 1) et 31% (MOY 2) lors de la résolution des problèmes de la classe 3 (tableau 6.11), et d'au moins 6.5% (MOY 1) et 7.3% (MOY 2) lors de la résolution des problèmes de la classe 4 (tableau 6.12).

, un temps de calcul **moyen** plus court que le meilleur des cinq autres, de 30% (MOY 1) et 31% (MOY 2) au tableau 6.11, ainsi que 6.5% (MOY 1) et 7.3% (MOY 2) au tableau 6.12.

En ce qui concerne le **nombre de noeuds traités**, le critère $CRIT_{12}$ est encore le plus efficace. Il est cependant suivi de près par le critère $CRIT_{13}$, ce qui indique que la stratégie 3, visant à brancher sur les lignes i_1 et i_2 qui élimine le plus de sous-matrices gloutonnes, est efficace sur ce point. Cependant, le dénombrement de sous-matrices gloutonnes exige une procédure très coûteuse en temps de calcul (voir les tableaux 6.9 à 6.12), ce qui élimine l'intérêt accordée au critère $CRIT_{13}$.

Les tableaux 6.11 et 6.12 regroupant des problèmes plus difficiles, la comparaison des branchements s'avère plus importante. Il semble donc qu'un critère de branchement basé sur l'élimination des colonnes, produisant un écart entre les bornes inférieure et supérieure (stratégie 2), est efficace. Naturellement, il faut comparer ce critère à d'autres afin d'évaluer vraiment son efficacité.

6.5 Branchement sur une variable

Il peut arriver que le fait de brancher sur une variable produise un arbre qui possède une structure débalancée, car la branche créée avec $x_j = 0$ n'élimine qu'une seule colonne alors qu'avec $x_j = 1$ on élimine la colonne j ainsi que toutes les lignes contenues dans I_j, cette seconde branche induit donc une contrainte plus forte. Il est donc intéressant de vérifier si le choix du nombre de variables de branchement influence beaucoup les résultats.

Nous avons résolu les problèmes une nouvelle fois en appliquant une nouvelle stratégie et suivant le type meilleur avant, mais cette fois-ci le branchement s'effectue sur une variable sans chercher de sous-matrices interdites. Nous pourrons

vérifier de cette façon si l'élimination **directe** de sous-matrice constitue une bonne stratégie ou non.

Afin de bien démontrer l'importance d'un branchement judicieux, nous définissons d'abord un nouvelle stratégie, qui consiste simplement à choisir une variable quelconque, possédant un coût réduit nul.

STRATÉGIE 4

Brancher sur une variable quelconque $j \in SOL$, où $SOL = \{j \in N : \bar{c}_j = 0\}$.

On appellera $CRIT_{14}$ le critère appliquant la stratégie 4 et utilisant un type de branchement meilleur d'abord.

Puis, nous développons une nouvelle stratégie très voisine de la stratégie 2, celle-ci consiste à brancher sur une variable recouvrant au moins une ligne sur-recouverte et est présentée ci-dessous:

STRATÉGIE 5

Soient les paramètres suivants:

- SOL $= \{j \in N : \bar{c}_j = 0\}$, ensemble des variables dont le coût réduit est nul.
- BASE, l'ensemble des variables de base qui définissent BS.
- SUR $= \{i \in M : \sum_{j \in BASE} a_{ij} x_j > 1\}$, ensemble des ligne sur-recouvertes.
- BSUR$_i$ $= \{J_i \cap$ BASE $\cap$ SOL $\}, i \in$ SUR .

Cette stratégie consiste à choisir $j_k \in N$, respectant les conditions suivantes:

(a) $j_k \in$ BSUR$_l$, où $u_l = \max_{i \in \text{VALIDE}} u_i$

(b) VALIDE $= \{i \in$ SUR : $|\text{BSUR}_i| > 0\}$

L'écart entre BI et BS est principalement causé par le surrecouvrement des lignes dont le multiplicateur est non-nul (propriété 3.4). On cherche donc à éliminer une variable sur la ligne où se produit le plus grand écart.

Afin de bien démontrer l'importance d'un branchement judicieux, nous allons comparer l'effet des critères $CRIT_{12}$ et $CRIT_{15}$, au critère $CRIT_{14}$ qui consiste à choisir une variable **quelconque** de coût réduit nul.

Les tableaux 6.13 à 6.16 contiennent les résultats des critères $CRIT_{14}$ ET $CRIT_{15}$ comparés au meilleur critère actuel, soit $CRIT_{12}$.

L'application du critère de branchement $CRIT_{14}$ donne des résultats comparable aux 2 autres pour la résolution des problèmes 1 à 30, qui sont relativement faciles. Cependant, l'efficacité de ce critère se détériore beaucoup lors de la résolution de problèmes plus difficiles, c'est-à-dire, les problèmes 31 à 60. Ceci démontre que pour la résolution des problèmes "faciles" (faible saut de dualité), il suffit de prendre des heuristiques duale et primale efficaces, et les résultats dépendent peu du critère de branchement utilisé.

Pour résoudre des problèmes "difficiles" (saut de dualité élevé), le critère de branchement doit être choisi judicieusement car celui-ci influence beaucoup la qualité des résultats.

Les problèmes traités étant de faible densité, il se peut fort bien que l'intersection de deux lignes ne contienne généralement qu'une seule colonne. Les critères $CRIT_{12}$ et $CRIT_{15}$ ne sont donc pas très différents. Afin de pouvoir mieux évaluer l'effet du nombre de variables impliquées sur l'efficacité du branchement, nous avons réappliqué le critère $CRIT_{12}$ en forçant le branchement sur l'intersection de lignes contenant au moins 2 éléments non-nuls. Ce critère n'a pas donné de très bon résultats, prenant en moyenne 4.6% fois plus de temps et générant 11% plus de noeuds qu'avec le critère $CRIT_{12}$. Il semble donc plus important de choisir a priori la ligne sur-recouverte la plus pénalisante, que de chercher à brancher sur plusieurs variables.

Problèmes 1 à 15	$CRIT_{15}$		$CRIT_{14}$		$CRIT_{12}$	
	t(s)	nbr.	t(s)	nbr.	t(s)	nbr.
MOY 1	111	12.7	135	28.3	120	12.5
MOY 2	136	16.8	170	31.1	151	17.7

Tableau 6.13: Temps de calcul moyen et nombre moyen de noeuds traités, selon les différents critères, pour les problèmes (200 $\times$ 500) de densité 2%.

Problèmes 16 à 30	$CRIT_{15}$		$CRIT_{14}$		$CRIT_{12}$	
	t(s)	nbr.	t(s)	nbr.	t(s)	nbr.
MOY 1	127	7.3	138	8.5	**126**	7.8
MOY 2	151	10.6	162	11.6	**150**	11.4

Tableau 6.14: Temps de calcul moyen et nombre moyen de noeuds traités, selon les différents critères, pour les problèmes (200 × 1000) de densité 2%.

Problèmes 31 à 45	$CRIT_{15}$		$CRIT_{14}$		$CRIT_{12}$	
	t(s)	nbr.	t(s)	nbr.	t(s)	nbr.
MOY 1	**959**	120	4064	431	1039	122
MOY 2	**1092**	138	5079	476	1184	140

Tableau 6.15: Temps de calcul moyen et nombre moyen de noeuds traités, selon les différents critères, pour les problèmes (200 × 500) de densité 5%.

Problèmes 46 à 60	$CRIT_{15}$		$CRIT_{14}$		$CRIT_{12}$	
	t(s)	nbr.	t(s)	nbr.	t(s)	nbr.
MOY 1	308	23.7	541	45.7	325	24.9
MOY 2	360	30.4	634	61.9	383	32.3

Tableau 6.16: Temps de calcul moyen et nombre moyen de noeuds traités, selon les différents critères, pour les problèmes (200×1000) de densité 5%.

6.6 Problèmes de hautes densités

Il est intéressant de voir comment se comporte notre algorithme lors de la résolution de problèmes de plus hautes densités. Pour ce faire, nous avons généré 2 séries de 15 problèmes de 100 contraintes et 500 variables, dont les densités sont respectivement 10% et 15%.

Nous désirons donc comparer les critères $CRIT_{15}$ et $CRIT_{12}$ afin de voir comment ils se comportent, et s'ils peuvent être départagés pour la résolution des problèmes de haute densité. Les résultats se retrouvent aux tableaux 2.17 et 2.18.

On observe aux tableaux 2.17 et 2.18 un temps de résolution intéressant et un nombre de noeuds généres assez bas. Le saut de dualité absolu est relativement bas quoi que le saut relatif (en %) est élevé pour les problèmes 76 à 90, dont la densité est de 15%. L'algorithme se comporte donc bien face aux problèmes de haute densité. Il est cependant difficile d'établir un rapport précis entre la densité des problèmes et leur difficulté (en terme de saut de dualité), parce que ces problèmes sont constitués de 100 contraintes et 500 variables, donc moins difficiles à résoudre que les problèmes de 200 contraintes par 500 variables, étant donné que le nombre de contraintes est moitié moins élevé.

Problème	$CRIT_{15}$		$CRIT_{12}$		ZP-ZPL
	t(s)	nbr.	t(s)	nbr.	(%) .
61	71	2	74	2	0
62	103	12	105	12	2.07
63	90	8	99	12	1.38
64	82	2	81	2	0
65	68	0	68	0	0
66	116	14	139	20	4.08
67	64	0	64	0	0
68	60	0	60	0	0
69	140	20	121	16	3.18
70	155	20	139	18	2.89
71	111	12	110	12	1.08
72	107	12	112	16	3.32
73	95	10	94	10	1.08
74	130	8	134	10	2.46
75	103	10	101	12	1.76
MOY 1	100	8.7	100	9.5	1.55
MOY 2	115	12.6	115	13.8	2.33

Tableau 6.17: Temps de calcul et nombre total de noeuds traités, selon les deux différents critères, pour les problèmes (100×500) de densité 10%.

Problème	$CRIT_{15}$		$CRIT_{12}$		ZP-ZPL
	t(s)	nbr.	t(s)	nbr.	(%)
76	60	0	60	0	0
77	61	0	61	0	0
78	161	10	193	20	6.18
79	89	0	89	0	0
80	226	26	269	34	6.48
81	94	0	94	0	0
82	73	0	73	0	0
83	174	16	196	20	4.03
84	158	12	231	20	2.56
85	173	26	160	12	5.26
86	127	10	120	8	2.60
87	128	8	158	14	2.56
88	186	32	185	20	3.51
89	260	32	239	30	6.22
90	206	20	194	24	6.21
MOY 1	145	12.8	155	13.5	3.04
MOY 2	180	19.2	194	20.2	4.56

Tableau 6.18: Temps de calcul et nombre total de noeuds traités, selon les deux différents critères, pour les problèmes (100 × 500) de densité 15%.

Les tableaux 6.17 et 6.18 révèlent que le critère $CRIT_{15}$ demeure plus efficace que le critère $CRIT_{12}$ pour résoudre ces problèmes de plus haute densité.

Le branchement sur une variable fortement impliquée dans l'écart (BS-BI) a très probablement comme effet l'augmentation de la borne inférieure dans chacune des 2 branches formées à partir du sous-problème courant. C'est la raison pour laquelle les arbres formés sont relativement bien balancés.

6.7 Amélioration du critère de branchement

Nous avons pu remarquer au cours des analyses que le choix de brancher sur une seule variable ou sur plusieurs variables avait peu d'influence sur l'efficacité du branchement. Le choix de la stratégie semble en fait plus déterminant. Nous avons observé qu'un branchement, visant à éliminer les colonnes, dont la variable associée est fortement impliquée dans l'écart qui existe entre les bornes inférieure et supérieure, est efficace.

Rappelons que l'écart entre BS et BI est donné par:

$$\text{(BS-BI)} = \sum_{j \in BASE} PENAL_j, \text{ où}$$

BASE = ensemble des indices des variables qui définissent la borne supérieure.

$$\text{SUR} = \{i \in M : |J_i \cap BASE| > 1\}$$

ensemble des lignes sur-recouvertes.

$$\text{PENAL}_j = c_j - \sum_{I_j \setminus SUR} u_i, j \in N.$$

PENAL_j est donc une indication du degré d'influence de la variable j sur l'écart (BS-BI).

La nouvelle stratégie suggérée consiste à brancher sur la variable de plus grande pénalité :

STRATÉGIE 6

$$PENAL_k = \max_{j \in BASE \cap N} PENAL_j.$$

ceci définit le critère $CRIT_{16}$.

Nous avons de nouveau résolu les problèmes 1 à 60, et les résultats se retrouvent aux tableaux 6.19 à 6.22. On remarque sur le tableau 6.21, c'est à dire celui dont les problèmes possèdent le plus haut saut de dualité, donc le plus important, que cette stratégie est efficace puisque elle réduit considérablement le temps de calcul, ainsi que le nombre de noeuds générés.

Ceci donne possiblement un indice sur la voie à suivre dans l'amélioration du critère de branchement. L'utilisation maximal de l'information disponible est donc suggérée. Plutôt que de n'utiliser que la solution du sous-gradient pour choisir la ou les variables de branchement, on exploite également la solution produite par l'heuristique primale qui nous a procuré la borne supérieure.

Problèmes 1 à 15	$CRIT_{15}$		$CRIT_{16}$	
	t(s)	nbr.	t(s)	nbr.
MOY 1	111	12.7	**107**	13.7
MOY 2	136	16.8	**133**	18.7

Tableau 6.19: Comparaison de $CRIT_{15}$ et de $CRIT_{16}$ pour les prob. 200 $\times$ 500 de densité 2%.

Problèmes 16 à 30	$CRIT_{15}$		$CRIT_{16}$	
	t(s)	nbr.	t(s)	nbr.
MOY 1	127	7.3	**125**	7.2
MOY 2	151	11.6	**146**	10.8

Tableau 6.20: Comparaison de $CRIT_{15}$ et de $CRIT_{16}$ pour les prob. 200 $\times$ 1000 de densité 2%.

Problèmes 31 à 45	$CRIT_{15}$		$CRIT_{16}$	
	t(s)	nbr.	t(s)	nbr.
MOY 1	959	120	**752**	81.3
MOY 2	1092	138	**852**	93.0

Tableau 6.21: Comparaison de $CRIT_{15}$ et de $CRIT_{16}$ pour les prob. 200 × 500 de densité 5%.

Problèmes 46 à 60	$CRIT_{15}$		$CRIT_{16}$	
	t(s)	nbr.	t(s)	nbr.
MOY 1	308	23.7	**272**	19.5
MOY 2	360	30.4	**311**	25.3

Tableau 6.22: Comparaison de $CRIT_{15}$ et de $CRIT_{16}$ pour les prob. 200 × 1000 de densité 5%.

6.8 Comparaison avec le branchement sur une contrainte

Nous avons déjà présenté, à la section 3.7.2, le principe du branchement sur une contrainte. Ce critère, étant utilisé par Fisher et Kedia [9] ainsi que par Chan, Bean et Yano [5], semble efficace, et il serait intéressant de le comparer au critère $CRIT_{16}$.

Nous définissons donc une nouvelle stratégie, décrivant la méthodes utilisé pour choisir la contrainte de branchement.

STRATÉGIE 7

Soient les paramètres suivants :

- SOL $= \{j \in N : \bar{c}_j = 0\}$, ensemble des variables dont le coût réduit est nul.
- BASE, l'ensemble des variables de base qui définissent BS.
- $d_i = \sum_{j \in BASE} a_{ij}, i \in M$. Nombre de fois que la ligne i est recouverte par une colonne de BASE.
- $PENAL_i = u_i(d_i - 1), i \in M$. Contribution de la ligne i dans l'écart (BS-BI).

Cette stratégie consiste à choisir la contrainte k, telle que :

- $PENAL_k = \max_{i \in M} PENAL_i$.

L'ordre de branchement sur les variables de la contrainte est donné, par la procédure suivante :

(a) LISTE = J_k.

(b) Brancher sur l, telle que $\bar{c}_l = \min_{j \in LISTE} \bar{c}_j$.

(c) LISTE = LISTE$\backslash l$.

(d) Si LISTE = $\emptyset$, alors FIN ; sinon, retourner en (b).

ceci définit le critère $CRIT_{17}$.

Nous avons résolu les problèmes 1 à 60 une nouvelle fois en utilisant le critère $CRIT_{17}$, et les résultats comparatifs ($CRIT_{17}$ vs $CRIT_{16}$) se trouvent aux tableaux 6.23 à 6.26.

La résolution des problèmes faciles (tableaux 6.23 et 6.24) a été plus rapide avec l'utilisation du critère $CRIT_{16}$, plutôt qu'avec le critère $CRIT_{17}$. Cependant, les problèmes difficiles (tableaux 6.25 et 6.26) ont été résolus plus efficacement avec le critère $CRIT_{17}$.

Deux commentaires importants peuvent être tirés de ces résultats:

- Tel qu'observé dans les tests précédents, il semble bien qu'un choix judicieux du critère de branchement est important lorsque l'on traite des problèmes difficiles (possédant un haut saut de dualité), même si la procédure utilisée est compliquée (et coûteuse en temps de calcul). Autrement dit, il peut être

avantageux de passer du temps pour améliorer une procédure de branchement, sachant que les problèmes à traiter seront difficiles.

- Une procédure de branchement compliquée (et coûteuse) peut augmenter inutilement le temps de résolution des problèmes faciles.

Problèmes 1 à 15	$CRIT_{17}$		$CRIT_{16}$	
	t(s)	nbr.	t(s)	nbr.
MOY 1	119	37.1	**107**	13.7
MOY 2	148	51.5	133	18.7

Tableau 6.23: Comparaison de $CRIT_{16}$ et de $CRIT_{17}$ pour les prob. 200 × 500 de densité 2%.

Problèmes 16 à 30	$CRIT_{17}$		$CRIT_{16}$	
	t(s)	nbr.	t(s)	nbr.
MOY 1	140	20.9	**125**	7.2
MOY 2	168	30.5	**148**	10.8

Tableau 6.24: Comparaison de $CRIT_{16}$ et de $CRIT_{17}$ pour les prob. 200 × 1000 de densité 2%.

Problèmes 31 à 45	$CRIT_{17}$		$CRIT_{16}$	
	t(s)	nbr.	t(s)	nbr.
MOY 1	467	94.4	752	81.3
MOY 2	523	107	852	93.0

Tableau 6.25: Comparaison de $CRIT_{16}$ et de $CRIT_{17}$ pour les prob. 200 × 500 de densité 5%.

Problèmes 46 à 60	$CRIT_{17}$		$CRIT_{16}$	
	t(s)	nbr.	t(s)	nbr.
MOY 1	260	34.1	272	19.5
MOY 2	290	43.0	311	25.3

Tableau 6.26: Comparaison de $CRIT_{16}$ et de $CRIT_{17}$ pour les prob. 200 × 1000 de densité 5%.

6.9 Conclusion

L'étude de critères, axés sur l'élimination des sous-matrices cycliques ont donné des résultats satisfaisants, mais ne furent pas les plus efficaces. Sachant fort bien que le saut de dualité est créé par des sous-matrices cycliques contenues dans la matrice des contraintes, nous avons étudié un critère de branchement cherchant à éliminer indirectement des sous-matrices cycliques.

Cependant, une stratégie visant à éliminer d'un sous-problème, les colonnes les plus impliquées dans l'écart existant entre les bornes inférieure et supérieure, semble plus efficace.

Enfin, une stratégie visant à brancher sur une contrainte ($CRIT_{17}$) semble être la meilleure méthode, parmis celles analysées.

La résolution des problèmes faciles semble dépendre beaucoup plus de l'efficacité des heuristiques utilisées pour calculer les bornes inférieure et supérieure, que du critère de branchement. Par contre, le choix d'un bon critère de branchement s'est avéré très important pour résoudre des problème possédant un haut saut de dualité.

Il serait peut-être intéressant de jumeler les stratégies utilisées par les critères $CRIT_{16}$ et $CRIT_{17}$, afin de construire un nouveau critère, encore plus efficace.

Conclusion

Notre objectif principal était d'évaluer l'efficacité de critères de branchement et de trouver une bonne règle de sélection des sous-problèmes, basés principalement sur l'élimination des sous-matrices cycliques. Pour ce faire, nous avons utilisé un algorithme de sous-gradient pour trouver une borne inférieure.

Les résultats obtenus révèlent tout d'abord que la sélection des sous-problèmes (type de branchement) suivant la règle **meilleur d'abord** est plus efficace qu'avec la règle **profondeur d'abord.** L'étude de performance, des critères de branchement, montre que les résultats obtenus ne furent pas ceux espérés, croyant que le branchement sur une sous-matrice (intersection de 2 lignes) serait plus efficace que le branchement sur une seule variable. Il semble en fait que le choix (stratégie utilisée) de la ou les variable(s) de branchement est plus important que le nombre de variables impliquées. Nous avons observé également que le choix du critère de branchement est très pertinent pour résoudre efficacement des problèmes difficiles, mais très peu pour les problèmes faciles. Enfin, le critère $CRIT_{17}$ est le plus efficace de l'ensemble des critères testés, celui-ci étant basé sur l'élimination de la contrainte la plus influente sur l'écart existant entre les bornes inférieure et supérieure.

Bien entendu, ces conclusions portent sur les classes de problèmes traités, et les règles utilisées pour les générer. Il serait intéressant d'éxpérimenter ces critères sur des problèmes réels ou des problèmes dont la fonction de coût est générée de façon différente. En effet, notre fonction de coût ne dépend pas du tout des contraintes que recouvrent les variables, et cela entraîne probablement l'élimination de plusieurs colonnes par la procédure de pénalités. Une fonction de coût définie de telle sorte que le coût d'une variable soit proportionnel au nombre de lignes recouvertes (en y ajoutant une perturbation) serait sûrement plus réaliste.

Nous prévoyons comparer une nouvelle fois ces critères en utilisant d'autres heuristiques duales efficaces, telles que 3-OPT [9, Fisher et Kedia] et MAM [5, Chan, Bean et Yano]. Ceci permettrait ainsi d'évaluer l'effet d'un changement au niveau de la borne inférieure sur les résultats, et voir si on arrive aux mêmes conclusions qu'avec l'utilisation d'un algorithme de sous-gradient. L'utilisation d'une autre heuristique primale efficace est aussi envisagée.

Bibliographie

1. Balas, E. et Ho, A.: Set Covering Algorithms Using Cutting Planes, Heuristics, and Subgradient Optimization : A Computational Study. **Math. Programming Study 12** (1980). pp. 37–60.

2. Balinski, M. L. et Quandt, R. E.: On an Integer Program for a Delivery Problem. **Operations Research 12** (1964). pp. 300–304.

3. Berge, C.: Balanced Matrices. **Math. Programming 2** (1972). pp. 19–31.

4. Camerini, P. M., Fratta, L. et Maffioli, F.: On Improving Relaxation Methods by Modified Gradient Techniques. **Math. Programming Study 3** (1975). pp. 26–34.

5. Chan, T. J., Bean, J. C., et Yano, C. A.: A Multiplier-Adjustment-Based Branch-and-Bound Algorithm for the Set Partitioning Problem. **Working Paper 87-OR-08, School of Engineering and Applied Science, Southern Methodist University, Dallas** .

6. Chvátal, V: A Greedy Heuristic for the Set-Covering Problem. **Math. of Operations Research 4** (1979). pp. 233–235.

7. Desrochers, M. et Soumis, F.: A Column Generation Approach to the Urban Transit Crew Scheduling Problem. **Transpn. Science 23** (1989). pp. 1–13.

8. Etcheberry, J.: The Set-Covering Problem : A New Implicit Enumeration Algorithm. **Operations Research 25** (1977). pp. 760–763.

9. Fisher, M. L. et Kedia, P.: Optimal Solution of Set Covering/Partitioning Problem Using Dual Heuristics. **Working Paper 88-11, College of Business Administration, Northeastern University, Boston** .

10. Gondran, M. et Minoux, M.: Graphes et Algorithmes. **Éditions Eyrolles** (1985). pp. 487–524.

11. Held, M., Wolfe, P. et Crowder, H. P.: Validation of Subgradient Optimisation. **Math. Programming Study 6** (1974). pp. 62–68.

12. Ho, A. C.-K.: Studies in Set Covering and Disjunctive Programming. **U-M-I Dissert. Infor. Service** (1981). pp. 1–39.

13. Hoffman, A. J., Kolen, A.W.J. et Sakorovitch, M.: Totally-Balanced and Greedy Matrices. **SIAM Alg. Disc. Meth. 6** (1985). pp. 721–730.

14. Ibaraki, T.: Enumerative Approaches to Combinatorial Optimization, Part 1. **Annals of Operations Research 10** (1987). pp. 196–206.

15. Kolen, A.: Solving Covering Problems and the Uncapacitated Plant Location Problem on Trees. **European Journal of Operational Research 12** (1983). pp. 266–278.

16. Nemhauser, G.L et Wolsey, L. A.: Integer and Combinatorial Optimization. **Wiley-Interscience** (1988). pp.1–10, 535–570.

17. Ryan, D. M. et Forster, B. A.: An Integer Programming Approach to Scheduling. **Dans Wren, A. Éditeur, Computer Scheduling of Public Transport, Urban Passenger Vehicle and Crew Scheduling.** North Holland (1981). pp. 269–280.

18. Smith, B. M. et Wren, A.: A Bus Crew Scheduling System using a Set Covering Formulation. **Transpn. Research. 22A** (1988). pp.97–108.