République Algérienne Démocratique Et Populaire
Ministère de L'Enseignement Supérieur et de la Recherche Scientifique

Université 8 mai 1945 de Guelma

Faculté Des Mathématiques, Informatique Et Sciences de la Matière
Département d'informatique

THÈSE DE DOCTORAT Troisième Cycle LMD

Option : Informatique

Présentée par

CHENCHE Sara

THÈME

Localisation des stations d'arrêt de lignes d'autobus urbains

---

Soutenue publiquement le : 04/10/2015 devant le jury composé de :

Pr. Hamid SERIDI	Prof	U.8 Mai 1945, Guelma	Président
Pr. Salim HADDADI	Prof	U.8 Mai 1945, Guelma	Directeur de thèse
Pr. Natalia DJELLAB	Prof	U. Badji mokhtar, Annaba	Examinateur
Pr. Meziane AÏDER	Prof	U. USTHB, Alger	Examinateur

Année 2015

Résumé

On a étudié trois problèmes d'optimisation combinatoire: le problème de localisation de stops dans un réseau de transport public (PLS), le problème de recouvrement d'ensemble (PRE) et celui de minimiser le nombre de blocs de 1 consécutifs dans une matrice binaire (PBC). Chacun des trois problèmes est NP-dur. Le premier est réduit au second au chapitre 2, où un état de l'art est proposé pour chacun des trois problèmes. On a ensuite présenté la panoplie des techniques utilisées dans notre étude au chapitre 3. On a proposé deux algorithmes pour le PRE, une heuristique lagrangienne et un algorithme exact de coupes. Chacune des deux méthodes a fait l'objet d'une expérimentation numérique rigoureuse, avec des comparaisons avec des méthodes existantes à l'issue desquelles nos méthodes ont montré leur supériorité. L'algorithme de coupes a fait l'objet d'un article soumis à « Journal of Combinatorial Optimization » publié par Springer. Une heuristique d'amélioration locale est proposée pour le PBC, fondée sur des bases théoriques, elle a fait l'objet d'une publication au journal « Information Processing Letters » publié par Elsevier.

Mots clés : Optimisation combinatoire, Problème de localisation de stops, Problème de recouvrement d'ensemble, Relaxation lgrangienne, Decomposition de Benders.

Abstract

We have studied three combinatorial optimization problems: stop location problem in public transportation networks (StopLoc), set covering problem (SCP), and that which seeks for minimizing the total number of blocks of consecutive ones in a binary matrix (CBM). Each of these three problems is known to be NP-hard. The first one is introduced in the second chapter, whither a state of the art of each one of the three problems is given. In the next chapter 3, we have presented the panoply of techniques used in this study. Therefore we have proposed two algorithms for the SCP, Lagrangian heuristic and an exact cutting algorithm. Either of the two methods was the subject of a rigorous numerical experiment, followed by comparisons with existing methods, thus our methods have shown their superiority. Cutting algorithm is submitted to “Journal of Combinatorial Optimization” published by Springer”. A local improvement heuristic, based on theoretical bases is proposed for the CBM, and it has been published in the journal “Information Processing Letters” published by Elsevier.

Keywords : Combinatorial optimization, Stop location problem, set covering problem, Lagrangian relaxation, Benders decoposition.

ملخـــــــص

يتناول هذا البحث دراسة ثلاث مشاكل في مجال التحسين: مشكل تمركز محطات التوقف على مستوى شبكة النقل العام, مشكل التغطية, و كذلك مشكل التقليل من عدد الاحاد المتتالية في مصفوفة ثنائية. كل من هذه المشاكل الثلاث عرفت بكونها صعبة NP-dur. المشكل الأول مطروح في الوحدة الثانية, أين تم تقديم دراسة حالة لكل من المشاكل الثلاث سالفة الذكر. بعد ذلك و في الوحدة الثالثة تطرقنا الى عرض مختلف التقنيات المستعملة في هذه الدراسة. و كنتيجة, قدمنا خوارزميتان لمشكل التغطية, الأولى هي خوارزمية Lagrangienne , أما الثانية فهي خوارزمية قطع صحيحة.أجريت اختبارات عددية و مقارنات لكلا الطريقتين مع مختلف الطرائق الموجودة مسبقا, وكانت النتيجة أن أثبتت طرقنا المقترحة مستوى عالي. خوارزمية القطع هي مشروع مقال أرسل الى "Journal of Combinatorial Optimization" المنشور من طرف Springer. كذلك خوارزمية تحسين ضمني مبنية على أسس نظرية, اقترحت لمشكل الاحاد المتتالية و نشرت في جريدة "Information Processing Letters" المنشورة من طـرف Elsevier.

الكلمـــــات الرئيسية: التحســـين التوافقي، مشكلة محطات التوقف، مشكلة التغطيةالشاملـــــــة، Relaxation Lagrangienne , Decomposition de Benders

Table des matières

Table des figures

Liste des tableaux

Abbréviations

RTP	**R**éseau de **T**ransport **P**ublic
PLS	**P**roblème de **L**ocalisation de **S**tations d'arrêt
PRE	**P**roblème de **R**ecouvrement d'**E**nsemble
PC1	**P**ropriété de **C**consécutivité des **1**
B1C	**B**loc de **1** **C**onsécutifs
PBC	**P**roblème de minimiser le nombre de **B**locs de 1 **C**onsécutifs
CIR	Problème de minimiser le nombre de blocs de 1 **CIR**culaires
SP	**S**ous **P**rogramme
PM	**P**rogramme **M**aître
PMR	**P**rogramme **M**aître **R**éduit
PLM	**P**rogramme **L**inéaire **Mixte**
PLNE	**P**rogramme **L**inéaire en **N**ombres **E**ntiers
PR	**P**roblème **R**elaxé
DL	**D**ual **L**agrangien
NC	Problème des **N**ombres **C**omposés
SAT	Problème de **SAT**isfaisabilité
BaB	**B**ranch-and-**B**ound **R**éduit

Chapitre 1

Introduction

La recherche opérationnelle a connu une expansion extraordinaire et une imbrication avec plusieurs autres disciplines (sciences de la décision, engineering, économie, médecine, etc). Théorie et algorithmique se sont développés à un rythme effréné ces dernières années à cause précisément de cette interdisciplinarité. D'un autre côté, à cause de la capacité limitée de l'ordinateur actuel, on est tout le temps confronté à la tâche de trouver des méthodes de plus en plus efficaces, c'est à dire obtenant une bonne réponse le plus rapidement possible.

Un problème important est celui de l'organisation du transport public, surtout dans les grandes villes. Souvent, les usagers sont insatisfaits par des coûts jugés élevés, par un mauvais service, et par le non respect des horaires de passage des moyens de transport (bus, train, tramway, etc). Il est très difficile de satisfaire les exigences des usagers (moindre coût, meilleur service) en raison de la complexité et de la taille des problème de planning. Cependant, la théorie et les méthodes de la recherche opérationnelle permettent de proposer un modèle adéquat du problème posé et, en général, de proposer de bonnes solutions. Si l'on se place d'un point de vue orienté usagers, on a trois problèmes qui se posent, localisation des arrêts, gestion des délais et tarification.

Dans cette thèse on ne considèrera que le premier problème, qui consiste à chercher à placer des arrêts le long des lignes d'autobus. On a des points d'arrêt demandés par les clients, et il faut trouver le nombre minimum de stops à placer de manière à recouvrir tous les points demandés.

La thèse est organisée ainsi. Dans le chapitre 2, on étudie trois problèmes. D'abord celui de la localisation des arrêts, dont on étudie la complexité et pour lequel on propose un état de l'art. Puisque le problème posé se réduit à un problème de recouvrement d'ensemble, on accorde une place privilégiée à ce dernier au chapitre 1. On étudie sa complexité et son approximation, on passe en revue quelques applications, et on propose un état de l'art. Une méthode de décomposition est utilisée au chapitre 6 pour résoudre le problème de localisation. Cette dernière est fondée sur une permutation des colonnes de la matrice binaire des contraintes de manière à minimiser les nombre de blocs de 1 consécutifs. Cette propriété et le problème posé feront l'objet d'une étude assez approfondie au chapitre 1.

Le chapitre 3 est un rappel des faits concernant trois concepts omniprésents dans cette thèse : la décomposition de Benders, la relaxation lagrangienne et la méthode de sous-gradients, et la complexité des problèmes.

Les trois derniers chapitres constituent notre contribution. On propose au chapitre 4 une heuristique lagrangienne pour le problème de recouvrement. Celle-ci est fondée sur trois idées-clés. Primo, une technique heuristique pour diminuer la taille de l'instance en fixant la plupart des variables pour obtenir un problème de plus petite taille. Secundo, une phase de construction d'un recouvrement est présentée avec l'idée d'incorporer la notion de regret. Tertio, le recouvrement construit est soumis à une procédure d'amélioration qui consiste à plonger le recouvrement trouvé dans un recouvrement plus large de manière à en extraire le meilleur possible. L'heuristique lagrangienne est codée en C, et est testée sur un grand nombre de données benchmark obtenue auprès d'une bibliothèque bien connue, OR-library. Les résultats obtenus sont comparés à ceux obtenus par six autres méthodes. Au vu de la comparaison, l'heuristique lagrangienne se montre compétitive avec ces dernières.

Le chapitre 5 est dédié à une heuristique d'amélioration locale pour le problème de minimiser le nombre de blocs dans une matrice binaire. La méthode est justifiée par des résultats théoriques et a fait l'objet d'une publication dans la revue « Information Processing Letters » publiée par Elsevier. Cette méthode est codée en C et testée sur des données réelles et des données engendrées aléatoirement. A notre connaissance, notre méthode est la première à être proposée pour ce problème, et donc aucune comparaison n'a été possible.

Une décomposition de Benders est proposée pour le problème de recouvrement au chapitre 6. En effet, ce dernier est transformé en un programme linéaire mixte, dont le nombre de variables entières dépend de la configuration des colonnes de la matrice des contraintes. Deux algorithmes de coupes sont mis au point et codés en C. Ils sont comparés avec l'algorithme du type Branch-and-Bound de Rüf et Schöbel, et se sont montrés très supérieurs aussi bien du point de vue de la qualité des solution obtenues que du point de vue de la vitesse d'exécution.

Une conclusion résume notre apport et propose quelques pistes de recherche pour le futur.

Deux choses avant de clore cette introduction. Il va sans dire qu'il ne nous a pas été possible d'appliquer ce travail sur le réseau de transport de la ville de Guelma. Enfin, nous n'oublierons pas de remercier vivement Niklaus Rüf, de l'université de Kaiserslautern en Allemagne, pour nous avoir grâcieusement fourni le pack de données C1P-data, qui contient des données réelles de problèmes de localisation de gares fournis par une compagnie de chemins de fer allemands Deutsche Bähn, et qui seront testés aux chapitres 4 et 5.

Chapitre 2

Problèmes étudiés

Ce chapitre présente, dans leur généralité, trois problèmes qui seront examinés dans cette thèse. A chaque fois, on commence par poser le problème, puis on présente sa complexité théorique et son approximation, enfin on propose les applications et un état de l'art. De nombreuses références bibliographiques, parmi celles qui sont citées, nous ont servi à la rédaction de ce chapitre, qui peut être regardé comme un travail de synthèse (*survey* diraient les anglo-saxons). Toutes les propositions sont fournies sans preuve, mais signalées par les références adéquates.

2.1 Problème de localisation de stations d'arrêt d'autobus

Les performances d'un réseau de transport public (RTP) sont influencés par de nombreux facteurs (voir [1–4]), et les problèmes posés se situent à au moins trois niveaux (voir figure 2.1). Le premier niveau consiste à tout ce qui touche la conception du réseau, c'est-à-dire le choix des lignes, l'installation des arrêts, etc (voir [5–7]). Le second niveau est celui de l'habillage des lignes, c'est-à-dire l'affectation des conducteurs, les horaires de passages, etc (voir [8–10]). Enfin, le troisième niveau concerne la tarification, c'est-à-dire la définition des zones et des tarifs (voir [11]). Le problème de localisation de stations d'arrêt (PLS) se situe au premier niveau, et c'est cela qui nous concerne.

Un RTP est graphe non orienté $G = (V, E)$ où V est l'ensemble des stations et E l'ensemble des liaisons entre les stations. La figure 2.2 montre un exemple.

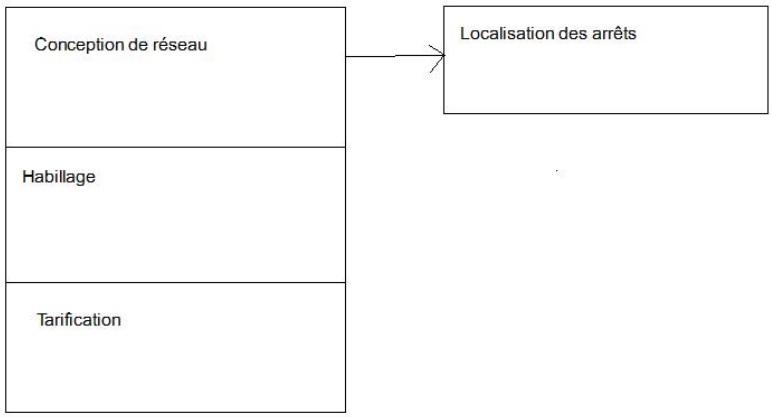

FIG. 2.1: Stratégies et opérations d'un système de transport urbain

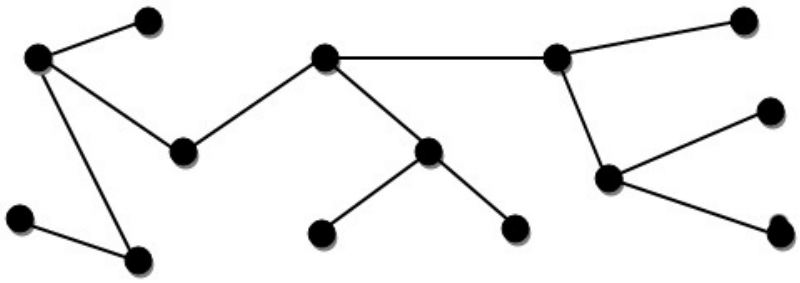

FIG. 2.2: Réseau de transport public

On admet bien qu'un grand nombre de stations d'arrêt est apprécié par les usagers en ce sens qu'elles leur évitent de trop marcher, mais elles ont l'inconvénient d'augmenter les temps de transit (en effet, selon [12], toute station additionnelle augmente le temps de voyage de 2 minutes), ce qui, paradoxalement, peut amener d'autres usagers à éviter le transport public.

2.1.1 Position du problème

Le PLS dans un RTP pose la question de savoir où et combien de stations d'arrêt installer de manière à recouvrir tous les passagers, sachant qu'un usager doit résider à moins d'une distance fixe de son arrêt demandé (voir la figure 2.3). Dons son survey [3], Demetsky assure que pour la plupart des compagnies le rayon de recouvrement est de 400 mètres pour le transport en commun, et 2 kilomètres dans le transport ferroviaire.

En fait, un RTP n'est presque jamais construit à partir du néant. On essaie toujours d'améliorer un réseau existant en ajoutant et/ou en enlevant des stations (voir [13]).

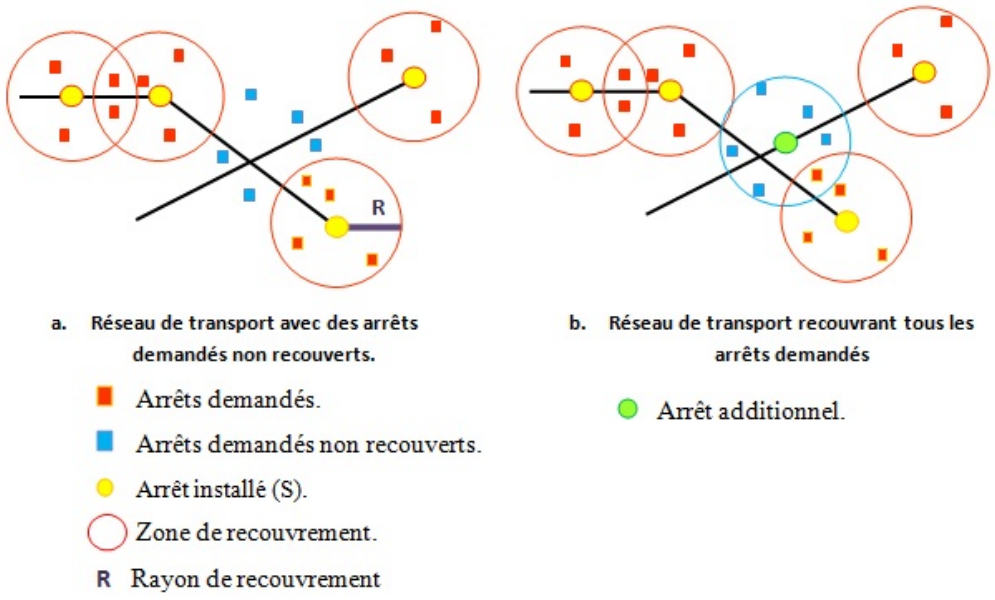

FIG. 2.3: Exemple de localisation de stations d'arrêt

Dans un RTP existant, étant donné un ensemble de points demandés $P = \{p_1 \cdots, p_d\} \subset \mathbb{R}^2$ et un rayon de recouvrement R, dénotons par S l'ensemble des nouvelles stations. Soit $d(s, p_i)$ la distance entre le point demandé et la station qui lui est la plus proche. On dit qu'un point $p_i \in P$ est recouvert par le RTP s'il existe une station $s \in S$ telle que la distance du point p_i à cette station n'est pas plus grande que R.

Ici, on considère le problème de localiser un nombre minimum de stations $S^* \subset S$ qui recouvrent tous les points demandés, autrement dit, chaque usager doit pouvoir atteindre au moins une station, sans que sa distance de marche ne dépasse le rayon de recouvrement R. Le PLS est formulé comme suit [14]

$$\begin{aligned} &\min |S| \\ &d(p, S) \leq R \quad p \in P \\ &S \subset V \end{aligned}$$

En définissant un ensemble de stations candidates $S_{cand} = \{s \in V | \exists p_i \in P, d(p_i, s) \leq R\}$ et une matrice binaire $A = (a_{ps})\, p \in P, s \in S_{cand}$ où

$$a_{ps} = \begin{cases} 1 & \text{si } d(p, s) \leq R \\ 0 & \text{sinon} \end{cases}$$

plusieurs travaux [7, 15–19] ont ramené le PLS au problème de recouvrement

$$\begin{array}{cc} \min \sum_{s \in S_{cand}} x_s & \\ \sum_{s \in S_{cand}} a_{ps} x_s \geq 1 & p \in P \\ x_s \in \{0,1\} & s \in S_{cand} \end{array}$$

où $x = (x_s)$ est le vecteur identificateur de chaque sous-ensemble $S \subset S_{cand}$

$$x_s = \begin{cases} 1 & \text{si } s \in S \\ 0 & \text{sinon} \end{cases}$$

2.1.2 Complexité

Le PLS est un problème à la jonction de plusieurs problèmes difficiles, «k-median», «k-center», «facility location» (voir [20]), recouvrement par des disques (voir [21]). Il est NP-dur [84]. La preuve se trouve dans [14] par réduction en le problème de recouvrement des arêtes par les sommets d'un graphe planaire. Cependant, dans le cas du PLS sur une ligne unique, appelé 1-stop location dans la littérature, est connu comme étant polynomialement résoluble [22], puisque la matrice binaire des contraintes a la propriété de consécutivité des 1.

2.1.3 Etat de l'art

2.2 Problème du recouvrement d'ensemble

Le problème du recouvrement d'ensemble(*set covering problem* en anglais), PRE par abbréviation, est un problème d'optimisation combinatoire célèbre de par ses nombreuses applications et par le fait qu'il se trouve être un modèle de base pour des problèmes plus difficiles, tels que le problème de tournées de véhicules (*vehicle routing problem*), le problème d'habillage de lignes aériennes ou d'autobus (*crew scheduling*). Il est donc central à la modélisation du problème de localisation des arrêts ou stops. Pour cette raison, il recevra une attention particulière. Dans ce chapitre, on présente le problème, ses applications, sa complexité théorique, son approximation ainsi qu'un état de l'art.

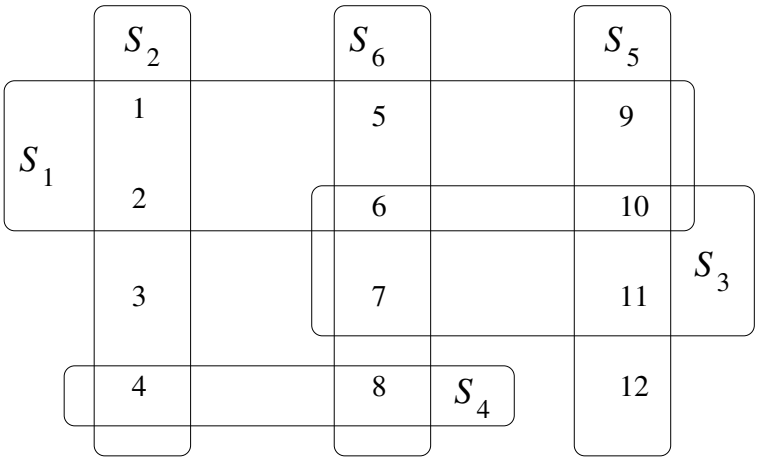

FIG. 2.4: Une instance du PRE

2.2.1 Position du problème

On se donne deux entiers positifs M et N, un ensemble $I = \{1, \cdots, M\}$ dont les éléments sont appelés items, et une famille $S_1, \cdots, S_N$ de N sous-ensembles de I. On appelle recouvrement (de l'ensemble I) toute sous-famille $C = \{S_{j_1}, \cdots, S_{j_p}\}$ comprenant certains sous-ensembles parmi les N donnés tels que

$$\cup_{k=1}^{p} S_{j_k} = I$$

Clairement, pour qu'un recouvrement puisse exister il faut et il suffit que

$$\cup_{j=1}^{N} S_j = I \tag{2.1}$$

Example 2.1. *Prenons* $M = 12$, $N = 6$, $I = \{1, \cdots, 12\}$, $S_1 = \{1, 2, 5, 6, 9, 10\}$, $S_2 = \{1, 2, 3, 4\}$, $S_3 = \{6, 7, 10, 11\}$, $S_4 = \{4, 8\}$, $S_5 = \{9, 10, 11, 12\}$ *et* $S_6 = \{5, 6, 7, 8\}$ *(voir figure 2.4).*

On voit qu'il est facile de chercher un recouvrement arbitraire puisqu'il suffit de tester (2.1). Si la condition est satisfaite alors l'union de tous les sous-ensembles est un recouvrement, et si (2.1) n'est pas satisfaite alors le PRE n'a pas de solution. A titre d'exemple (voir figure 2.5) l'ensemble $C = \{S_2, S_5, S_6\}$ est un recouvrement pour le PRE de l'exemple 2.1.

On peut d'ores et déjà supposer, sans perte de généralité, que chaque $S_j \neq \varnothing$ et que chaque item i appartient à au moins un des sous-ensembles. Si un S_j est vide, il peut être

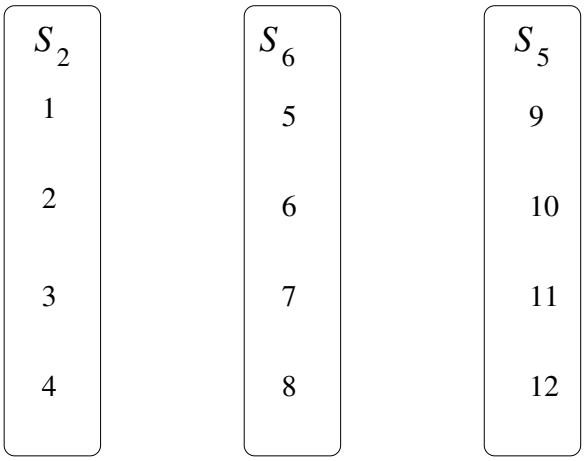

FIG. 2.5: Un recouvrement pour le PRE de l'exemple

supprimé du problème car il ne pourra recouvrir aucun item. Si un item i n'appartient à aucun sous-ensemble alors, clairement, il ne pourra jamais être recouvert, et le PRE n'a pas de solution.

Ce qui complique le problème est l'adjonction d'un nouveau paramètre. A chaque sous-ensemble S_j on attache un coût c_j, ce qui fait que le coût du recouvrement C est $\sum_{k=1}^{p} c_{j_k}$, et alors le PRE pose le problème de chercher un recouvrement de moindre coût.

Remark 2.1. La structure de la figure 2.4 est appelée hypergraphedans le jargon de la théorie des graphes. Les items sont les sommets et les sous-ensembles sont les hyperarêtes de l'hypergraphe. Si on attache un coût à chaque hyperaête, le problème de chercher un recouvrement (dit « des sommets par les hyperarêtes ») de moinde coût consiste à chercher un ensemble d'hyperarêtes recouvrant tous les sommets au moindre coût.

Soient $J = \{1, \cdots, N\}$ et $A = \left(a_j^i\right)$ la $M \times N$-matrice binaire dont les colonnes sont les vecteurs d'incidence des sous-ensembles S_j. On peut formuler le PRE en termes de programme linéaire binaire ainsi

$$\begin{array}{ll} \min \sum_{j \in J} c_j x_j & \\ \sum_{j \in J} a_j^i x_j \geq 1 & i \in I \\ x_j \in \{0,1\} & j \in J \end{array}$$

La variable x_j indique si le sous-ensemble S_j est pris dans la couverture ($x_j = 1$) ou non ($x_j = 0$). Les contraintes $\sum_{j \in J} a_j^i x_j \geq 1$ expriment que chaque item i doit être recouvert.

Example 2.2. *En prenant* $c_1 = 2, c_2 = 5, c_3 = 2, c_4 = 1, c_5 = 3, c_6 = 6$*, le programme linéaire binaire associé au PRE de l'exemple 2.1 est*

$$
\begin{array}{lcccccccc}
\min & 2x_1 & +5x_2 & +2x_3 & +x_4 & +3x_5 & +6x_6 & & \\
& x_1 & +x_2 & & & & & \geq 1 \\
& x_1 & +x_2 & & & & & \geq 1 \\
& & x_2 & & & & & \geq 1 \\
& & x_2 & & +x_4 & & & \geq 1 \\
& x_1 & & & & & +x_6 & \geq 1 \\
& x_1 & & +x_3 & & & +x_6 & \geq 1 \\
& & & x_3 & & & +x_6 & \geq 1 \\
& & & & x_4 & & +x_6 & \geq 1 \\
& x_1 & & & & +x_5 & & \geq 1 \\
& x_1 & & +x_3 & & +x_5 & & \geq 1 \\
& & & x_3 & & +x_5 & & \geq 1 \\
& & & & & x_5 & & \geq 1 \\
& x_1, & x_2, & x_3, & x_4, & x_5, & x_6 & \in \{0,1\}
\end{array}
$$

En examinant rapidement le programme linéaire binaire, on voit que la troisième contrainte ($x_2 \geq 1$ qui implique $x_2 = 1$) exige de prendre le sous-ensemble S_2 dans n'importe quel recouvrement car il est le seul à pouvoir recouvrir l'item 3. Il en est de même pour le sous-ensemble S_5 car il est le seul à pouvoir recouvrir l'item 12. Le recouvrement présenté à la figure 2.5 correspond à $x_2 = x_5 = x_6 = 1$ et $x_1 = x_3 = x_4 = 0$.

Soit A la matrice des contraintes, soient $a^1, \cdots, a^N$ ses colonnes et $a_1, \cdots, a_M$ ses lignes. Il y a correspondance bijective entre items, contraintes du programme linéaire binaire et lignes de la matrice A. Il en est de même pour les sous-ensembles, variables et colonnes de A. Le PRE est donc complètement défini par la donnée de A et du vecteur $c = (c_1, \cdots, c_N)$.

On peut réduirela taille du PRE, c'est-à-dire diminuer le nombre de lignes et de colonnes de A, par de simples implications logiques (voir par exemple [23]). Soit e_k le $k^{\text{ème}}$ vecteur unité (dont toutes composantes sont nulles sauf celle de numéro k qui vaut 1).

Critère 1. Si $a_i \geq a_j$ alors la ligne a_i peut être supprimée car elle est automatiquement recouverte dès que la ligne a_j l'est.

Critère 2. Si $a_i = e_k$ alors la colonne a^k peut être supprimée (cal elle doit faire partie de tout recouvrement puisque c'est la seule pouvant recouvrir l'item i avec forcément $x_k = 1$). Ensuite, chaque ligne a_t telle que $t \in S_k$ peut être effacée puisqu'un tel item est déjà recouvert par la colonne a^k.

Critère 3. Soit $E \subset J = \{1, \cdots, N\}$ un ensemble arbitraire. Si

$$\sum_{j \in E} a^j \geq a^k \text{ et } \sum_{j \in E} c_j \leq c_k$$

alors la colonne a^k ($x_k = 0$) peut être supprimée puisque tout item recouvert par a^k peut être recouvert par les colonnes de E à moindre coût.

Example 2.3. *Illustrons ces règles sur les données de l'exemple 2.1 avec le tableau suivant dont la première colonne donne le numéro de ligne, la première ligne donne le numéro de colonne, la seconde ligne, les coûts et la matrice A dans ce qui reste. Les cases omises sont des zéros.*

	1	2	3	4	5	6
	2	5	2	1	3	6
1	1	1				
2	1	1				
3		1				
4		1		1		
5	1					1
6	1		1			1
7			1			1
8				1		1
9	1				1	
10	1		1		1	
11			1		1	
12					1	

Le critère 2 a pour conséquence d'éliminer les lignes a_1, a_6, a_{10}*, puisque* $a_1 \geq a_2, a_6 \geq a_5$ *et* $a_{10} \geq a_9$*, pour obtenir*

$$\left[\begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ & 2 & 5 & 2 & 1 & 3 & 6 \\ \hline 2 & 1 & 1 & & & & \\ 3 & & 1 & & & & \\ 4 & & 1 & & 1 & & \\ 5 & 1 & & & & & 1 \\ 7 & & & 1 & & & 1 \\ 8 & & & & 1 & & 1 \\ 9 & 1 & & & & 1 & \\ 11 & & & 1 & & 1 & \\ 12 & & & & & 1 & \end{array}\right]$$

Selon le critère 1, on a bien $a_3 = e_2$ *et* $a_{12} = e_5$*. Ce qui nous permet de supprimer les colonnes* a^2 *et* a^5 *(en posant* $x_2 = 1$ *et* $x_5 = 1$*) et de supprimer les lignes* a_2, a_3, a_4 *dont les items sont recouverts par la colonne* a^2 *et les lignes* a_9, a_{11}, a_{12} *dont les items sont recouverts par la colonne* a^5*. On obtient les nouvelles données*

$$\left[\begin{array}{c|cccc} & 1 & 3 & 4 & 6 \\ & 2 & 2 & 1 & 6 \\ \hline 5 & 1 & & & 1 \\ 7 & & 1 & & 1 \\ 8 & & & 1 & 1 \end{array}\right]$$

Selon le critère 3 on a bien

$$a^1 + a^3 + a^4 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \geq a^6 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

et $c_1 + c_3 + c_4 = 5 \leq c_6 = 6$. *On peut donc supprimer la colonne* a^6 *en posant* $x_6 = 0$.

$$\begin{bmatrix} & 1 & 3 & 4 \\ & 2 & 2 & 1 \\ 5 & 1 & & \\ 7 & & 1 & \\ 8 & & & 1 \end{bmatrix}$$

L'application répétée trois fois du second critère permet d'éliminer respectivement la colonne a^1 *(*$x_1 = 1$*) et la ligne* a_5*, la colonne* a^3 *(*$x_3 = 1$*) et la ligne* a_7*, la colonne* a^4 *(*$x_4 = 1$*) et la ligne* a_8*. On a vidé toutes les contraintes et toutes les variables. On a donc résolu le PRE. On a retenu* $x_1 = x_2 = x_3 = x_4 = x_5 = 1$ *et* $x_6 = 0$*. Le recouvrement optimal de coût* $c_1 + c_2 + c_3 + c_4 + c_5 = 13$ *est formé donc des sous-ensembles* $S_1, \cdots, S_5$.

Remark 2.2. Les règles d'inférence prcécédentes peuvent être répétées, dans n'importe quel ordre, sur les nouvelles données. En pratique, sur les problèmes de grande taille dont la densité de la matrice

$$\frac{\sum_{i,j} a^i_j}{M \times N}$$

est faible, les deux premiers critères n'aboutissent que rarement. Elles ne justifient donc pas l'effort qu'elles nécessitent. En revanche, le troisième critère est important car, souvent, il permet de réduire effectivement le nombre de variables d'office nulles (voir l'expérimentation numérique menée au chapitre 4).

2.2.2 Complexité

Chercher le recouvrement de moindre coût n'est pas une tâche facile (même sur le petit exemple présenté). Une méthode « force brute » est donnée ci-après. Il est évident que cet algorithme donne toujours la bonne réponse. Mais quelle est sa complexité ? Le traitement à l'intérieur de la boucle « pour » ne coûte pas cher puisque tester si l'ensemble D constitue un recouvrement se fait en $\mathrm{O}(M \times N)$ et calculer son coût nécessite $\mathrm{O}(N)$ opérations. Le problème est que la boucle est exécutée 2^N fois puisque c'est le nombre de parties d'un ensemble à N éléments. Cette méthode est clairement exponentielle. Une première mauvaise nouvelle est qu'au stade actuel de nos connaissances on ne connaît pas d'algorithme plus efficace que ForceBrute. Les seuls algorithmes que l'on sache

mettre en œuvre sont des améliorations de ForceBrute, de même complexité, en essayant de procéder à une énumération intelligente d'une « faible » portion des parties de l'ensemble $\{S_1, \cdots, S_N\}$.

Algorithme ForceBrute
Entrée $M, N, S_1, \cdots, S_N, c_1, \cdots, c_N$
Sortie recouvrement C et son coût z
$z \longleftarrow \infty$
pour chaque partie D de l'ensemble $\{S_1, \cdots, S_N\}$ **faire**
  si D constitue un recouvrement **alors**
    calculer son coût α
    si $z > \alpha$ **alors** $z \longleftarrow \alpha$, $C \longleftarrow D$
  fin si
fin pour

Soit r un nombre fixé d'avance. Le problème de décision associé au PRE est le suivant.

PRE-décision

Instance $M, N, S_1, \cdots, S_N, c_1, \cdots, c_N$ et r

Question Existe-t-il un recouvrement de coût au plus r ?

PRE-décision est évidemment dans $\mathbb{NP}$ puisque un certificat succinct est un ensemble $C = \{S_{j_1}, \cdots, S_{j_p}\}$ de petite taille digitale, et dont la validation, on l'a déjà remarqué, prend un temps $\mathrm{O}(M \times N)$.

Proposition 2.3. *[24] PRE-décision est $\mathbb{NP}$-complet.*

En fait, la référence [24] montre qu'un cas particulier du PRE est $\mathbb{NP}$-complet. C'est celui où l'on se donne un graphe $(\mathcal{X}, \mathcal{E})$ avec $N = |\mathcal{X}|$ sommets et $M = |\mathcal{E}|$ arêtes. Les items sont alors les arêtes et les sous-ensembles sont en correspondance avec les sommets du graphe. Un sous-ensemble S_i contient toutes les arêtes qui sont adjacentes au sommet i. On cherche un recouvrement (dit « des arêtes par les sommets ») de cardinalité minimum, c'est-à-dire un sous-ensemble C des sommets de cardinalité minimum tel que toute arête du graphe a une extrémité dans C. C'est un PRE bien particulier car chaque sous-ensemble a un coût égal à 1 et chaque ligne de la matrice binaire $\mathbf{A}$ des contraintes a exactement deux 1 par ligne.

Example 2.4. *Pour le graphe de la figure 2.6, on a $M = 7$ et $N = 6$ avec $S_1 = \{e_1, e_2, e_3\}$, $S_2 = \{e_1, e_4\}$, $S_3 = \{e_5\}$, $S_4 = \{e_4, e_5, e_6\}$, $S_5 = \{e_2, e_6, e_7\}$, $S_6 = \{e_3, e_7\}$.*

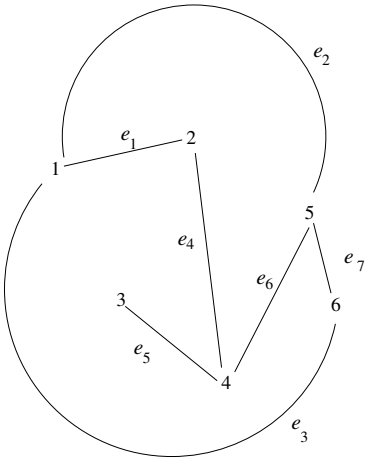

FIG. 2.6: Un graphe est un hypergraphe spécial

Ici

$$\mathbf{A} = \begin{pmatrix} 1 & 1 & & & & \\ 1 & & & & 1 & \\ 1 & & & & & 1 \\ & 1 & & 1 & & \\ & & 1 & 1 & & \\ & & & 1 & 1 & \\ & & & & 1 & 1 \end{pmatrix}$$

On peut voir que $C = \{1, 4, 5\}$ est un recouvrement (de cardinalité minimum).

Si certaines instances particulières peuvent être polynomialement résolues (voir par exemple [25, 26]), la seconde mauvaise nouvelle est que le PRE est fortement $\mathbb{NP}$-complet [24], ce qui signifie qu'il ne peut même pas être résolu à l'aide d'un algorithme pseudo-polynomial.

2.2.3 Approximation

Dans cette section, on diminue notre prétention. Il ne s'agit plus de résoudre le PRE pour obtenir un recouvrement optimal, mais se contenter de chercher un recouvrement approximatif. Cependant, cette solution approximative doit satisfaire deux critères.

1. Elle doit être obtenue à la faveur d'un algorithme polynomial efficace (un tel algorithme est appelé heuristique).

2. Son coût ne doit pas « trop s'éloigner » du coût d'un recouvrement optimal.

Formalisons ce dernier point. Soit $\mathcal{I}$ une instance de PRE et soit $opt(\mathcal{I})$ le coût d'un recouvrement optimal associé à l'instance $\mathcal{I}$. Soient $\mathcal{H}$ une heuristique polynomiale pour PRE et $\mathcal{H}(\mathcal{I})$ le coût du recouvrement approximatif produit par l'heuristique $\mathcal{H}$ appliquée à l'instance $\mathcal{I}$.

Definition 2.4. L'heuristique $\mathcal{H}$ est appelée α-approximation du PRE s'il existe une constante α, $1 < \alpha < \infty$, telle que pour toute instance $\mathcal{I}$ on ait

$$\mathcal{H}(\mathcal{I}) \leq \alpha \cdot opt(\mathcal{I})$$

On trouve également dans la littérature le terme « approximation avec garantie $\alpha - 1$ » car elle signifie que l'heuristique offre une garantie, quelque soit l'instance $\mathcal{I}$, que le coût approximatif $\mathcal{H}(\mathcal{I})$ ne s'écarte pas de plus de $\alpha - 1$ relativement à l'optimum

$$\frac{\mathcal{H}(\mathcal{I}) - opt(\mathcal{I})}{opt(\mathcal{I})} \leq \alpha - 1$$

Se pose alors la question : existe-t-il une α-approximation pour le PRE, et si elle existe est-elle satisfaisante ?

Afin d'aborder cette intéressante question, présentons une heuristique bien connue, l'heuristique gloutonne [27], [28]. Un algorithme est dit glouton si toute décision est prise une fois pour toutes et n'est jamais remise en cause. De cette manière, l'algorithme prend les décisions l'une après l'autre (il les avale tel un glouton) et s'arrête une fois toutes les décisions prises. L'algorithme glouton est le rêve des développeurs.

Commençons par donner l'énoncé, puis expliquons un peu le déroulement de cette heuristique. Au départ, le recouvrement est vide et son coût est nul. L'ensemble U est celui des items non encore recouverts et V est celui des sous-ensembles restant à examiner. Idéalement, on voudrait commencer par le sous-ensemble de moindre coût et de plus grande cardinalité (c'est-à-dire recouvrant le plus d'items) en se restreignant aux items non encore recouverts. Puisque cela n'est pas possible en général, alors contentons-nous de celui qui minimise le rapport coût/cardinalité. Une fois que ce sous-ensemble est identifié, on le prend dans le recouvrement et on met à jour le coût, l'ensemble des items non encore recouverts, et l'ensemble des sous-ensembles restants.

Il s'agit bien là bien d'un algorithme glouton puisque, initialement le recouvrement est vide, et à chaque passage dans la boucle, on introduit un sous-ensemble jusqu'à ce que tous les items soient recouverts. Il est facile de voir que c'est un algorithme polynomial.

Algorithme GLOUTON
Entrée $M, N, S_j, c_j, j \in J$
Sortie C, z
  $C \longleftarrow \varnothing$
  $z \longleftarrow 0$
  $U \longleftarrow I$
  $V \longleftarrow J$
  tant que $U \neq \varnothing$ **faire**
    chercher l'ensemble S_k tel que

$$k = \arg\min_{j \in V} \frac{c_j}{|S_j \cap U|}$$

    $C \longleftarrow C \cup \{S_k\}$
    $z \longleftarrow z + c_k$
    $U \longleftarrow U - S_k$
    $V \longleftarrow V - \{k\}$
  fin tant que
  retourner C, z

Example 2.5. *Cherchons, à l'aide de cette heuristique, un recouvrement approximatif pour le PRE de l'exemple 2.2.*

Regardons le premier tableau de l'exemple 2.3. Si l'on compare les rapports, on observe que le sous-ensemble S_1 *est celui qui donne le plus petit* $(2/6)$*. On fait* $C = \{S_1\}, V = \{2,3,4,5,6\}, U = \{3,4,7,8,11,12\}$ *, et* $z = 2$*, avec les nouvelles données*

$$\begin{bmatrix}
 & 2 & 3 & 4 & 5 & 6 \\
 & 5 & 2 & 1 & 3 & 6 \\
3 & 1 & & & & \\
4 & 1 & & 1 & & \\
7 & & 1 & & & 1 \\
8 & & & 1 & & 1 \\
11 & & 1 & & 1 & \\
12 & & & & 1 &
\end{bmatrix}$$

Le plus petit rapport est $1/2$ *pour* S_4 *:* $C = \{S_1, S_4\}, V = \{2, 3, 5, 6\}, U = \{3, 7, 11, 12\}$, *et* $z = 3$

$$\begin{bmatrix} & 2 & 3 & 5 & 6 \\ & 5 & 2 & 3 & 6 \\ 3 & 1 & & & \\ 7 & & 1 & & 1 \\ 11 & & 1 & 1 & \\ 12 & & & 1 & \end{bmatrix}$$

On continue ainsi avec $C = \{S_1, S_3, S_4\}, V = \{2, 5, 6\}, U = \{3, 12\}$, *et* $z = 5$

$$\begin{bmatrix} & 2 & 5 & 6 \\ & 5 & 3 & 6 \\ 3 & 1 & & \\ 12 & & 1 & \end{bmatrix}$$

Il ne reste plus qu'à introduire aux deux prochaines itérations les sous-ensembles S_2, S_5 *pour obtenir le recouvrement approximatif (qui se trouve être optimal).*

Definition 2.5. Etant donné un entier positif n, le $n^{\text{ème}}$ nombre harmonique est

$$H(n) = \sum_{i=1}^{n} \frac{1}{i}$$

avec, par convention, $H(0) = 0$.

Proposition 2.6. *[29] Pour tout* n *entier positif on a* $\ln(n+1) \leq H(n) \leq 1 + \ln(n)$.

Proposition 2.7. *[27, 30] L'heuristique gloutonneest une* α*-approximation du PRE avec* $\alpha = H(M) \simeq \ln(M)$.

On préfère, de loin, une valeur constante de α. Ici, α dépend du nombre d'items de l'instance. Ce qui fait que, l'on peut trouver une instance telle que le coût approximatif peut être arbitrairement éloigné du coût optimal. Regardons cet exemple dû à Chvátal [27].

Example 2.6. *Prenons comme données :* $I = \{1, \cdots, M\}$ *et* $J = \{1, \cdots, M+1\}$ *avec* $S_j = \{j\}, c_j = 1/j$ *pour* $j = 1, \cdots, M$ *et* $S_{M+1} = I, c_{M+1} = 1$.

$$\begin{bmatrix} & 1 & 2 & \cdots & M & M+1 \\ & 1 & 1/2 & \cdots & 1/M & 1 \\ 1 & 1 & & & & 1 \\ 2 & & 1 & & & 1 \\ \vdots & & & \ddots & & \vdots \\ M & & & & 1 & 1 \end{bmatrix}$$

Alors que le recouvrement optimal est $C^* = \{S_{M+1}\}$ *de coût* 1 *alors que l'application de l'heuristique de Chvátal va produire le recouvrement* $C = \{S_1, S_2, \cdots, S_M\}$ *de coût*

$$1 + \frac{1}{2} + \cdots + \frac{1}{M} = H(M)$$

L'heuristique de Chvátal n'offre donc pas une bonne garantie. Peut-on trouver une heuristique offrant une meilleure garantie que $\ln(M)$? La troisième mauvaise nouvelle concernant le PRE est que la réponse à cette question est non : la meilleure garantie possible est $O(\ln M)$ [29, 31]. Rappelons que cette garantie concerne la pire des instances. Ce qui signifie qu'il peut très bien exister des heuristiques donnant de très bons résultats sur la plupart des instances mais sans aucune bonne garantie théorique. C'est-à-dire qu'on peut tomber sur une instance $\mathcal{I}$ pathologique pour laquelle le coût approximatif $\mathcal{H}(\mathcal{I})$ s'éloigne autant que l'on veut du coût optimal $opt(\mathcal{I})$.

Definition 2.8. Etant donné $\epsilon > 0$, un schéma polynomial d'approximation est une heuristique $\mathcal{H}$ telle que pour toute instance $\mathcal{I}$ on ait

$$\mathcal{H}(\mathcal{I}) \leq (1 + \epsilon) \cdot opt(\mathcal{I})$$

et dont le temps d'exécution est majoré par un polynôme en la taille de $\mathcal{I}$.

Par suite du résultat précédent de Feige [29], il s'en suit une quatrième mauvaise nouvelle.

Proposition 2.9. *[32–34] Il ne peut exister de schéma polynomial d'approximation du PRE à moins que* $\mathbb{NP} = \mathbb{P}$.

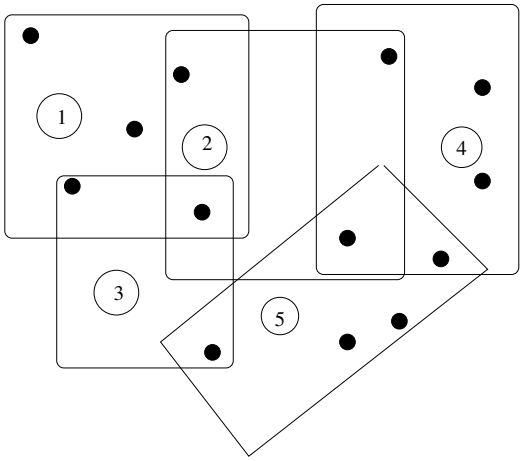

FIG. 2.7: Emplacements possible pour les antennes-relais

2.2.4 Applications

Le PRE a de nombreuses applications pratiques dont on va passer quelques unes en revue.

a) Localisation des antennes-relais de téléphonie mobile

Un territoire comprend M localités. Un travail technique liminaire a permis de préciser les N emplacements possibles pour les antennes. On connaît le coût de construction c_j de l'emplacement numéro j et on sait l'ensemble S_j des localités qu'il peut couvrir. Un exemple est présenté à la figure 2.7 avec des localités (petits cercles en gras) et cinq emplacements possibles (en cercles maigres numérotés). Le problème est de construire des antennes à certains emplacements de manière à pouvoir recouvrir toutes les localités au moindre coût. On voit qu'une solution possible est de construire trois antennes aux emplacements 1, 4 et 5, permettant de couvrir toutes les localités.

b) Interrogation d'une banque de données

La banque totalise N fichiers $S_j, j = 1, \cdots, N$ chacun de longueur c_j. Elle reçoit M requêtes, sachant que chaque requête est contenue dans au moins un des fichiers. Le problème est de chercher un ensemble de fichiers à consulter de longueur totale minimum qui satisfasse toutes les requêtes.

c) Rotation d'équipages (voir [35–44])

Une compagnie aérienne assure M vols (un vol est une liaison directe sans escale). Ces vols peuvent être combinés pour former N lignes $S_1, \cdots, S_N$ (une ligne est un ensemble

de vols ponctués par des escales) de sorte qu'un équipage donné peut servir sur tous les vols de la ligne. Le nombre de lignes est généralement beaucoup plus grand que celui des vols. Il s'agit alors de chercher le nombre minimum d'équipages qui puisse assurer tous les vols. Un exemple est donné au tableau 2.1.

	S_1	S_2	S_3	S_4
Alger-Paris	1			
Alger-Tunis		1		
Alger-Oran			1	1
Oran-Londres			1	
Oran-Tunis				1
Tunis-Le Caire		1		1

TAB. 2.1: Exemple de programme de vols dans une compagnie aérienne.

En vérité, le PRE n'est que le modèle de base du problème d'habillage. Ce dernier nécessite de compliquer le PRE par l'adjonction possible de nombreuses contraintes socio-économques. A titre d'exemple, on peut imposer qu'au moins 75% des lignes retenues finissent par un retour au point de départ. On peut exiger aussi par exemple, à cause de la contrainte de fatigue du personnel naviguant, qu'aucun équipage ne fasse plus d'une distance donnée.

d) **Autres applications**

On peut citer le problème de localisation de services d'urgence tels que pompiers, ambulanciers, etc. (voir [45–48]). On a aussi le problème de l'habillage des horaires de lignes d'autobus (voir [49]). Les problèmes de tournées de véhicules, tel que le ramassage scolaire, la distribution de courrier, la livraison par camion, etc. ([50]), le problème du découpage électoral (voir [51]), celui de la localisation des arrêts de bus ([15]), l'affectation de trafic dans un satellite ([52]), la construction de gabarits pour l'analyse de données ([53]), le séquençage d'ADN ([54]), l'ordonnancement ([55]), la sélection de la taille de lingots dans la sidérurgie ([56]), et bien d'autres encore (voir [57–63]).

2.2.5 Etat de l'art

Le PRE a fait l'objet d'une littérature abondante qui dépasse largement ce qu'on a pu recenser.

a) Il y a des études bibliographiques commentées ou *surveys* qui présentent à la date de leur parution l'état de l'art actuel. On peut citer [64–68].

b) Plusieurs travaux présentent des algorithmes exacts pour chercher un recouvrement optimal. Balas et Carrera [69] proposent une méthode Branch-and-Bound (BaB) basée sur une méthode de sous-gradients dynamique. Balas et Ho [70] utilisent des coupes associées à des heuristiques et une méthode de sous-gradients. Beasley [71] décrit un algorithme de type BaB. Beasley et Jörnsten [72] l'améliorent ensuite. La méthode BaB suggérée par Fisher et Kedia [73] est enrichie par l'usage d'heuristiques duales pour améliorer la borne inférieure. Haddadi [74] met au point une méthode BaB fondée sur un partitionnement original des contraintes. Puis, avec Benchettah [75], il décrit une méthode BaB basée sur une décomposition lagrangienne. Mannino et Sassano [76] réalisent un algorithme de type BaB enrichi par des procédures primales et duales servant à améliorer les deux bornes, afin de résoudre des instances difficiles provenant du problème de l'arbre de Steiner. Hoffman et Padberg mettent au point un algorithme du type Branch-and-Cut [77]. Enfin, Yeh utilise la programmation dynamique [78]. Hélas, tous ces algorithmes exacts ne sont applicables qu'à des instances de petite taille. D'ailleurs, lors de leur étude comparative des algorithmes exacts, Caprara et al [64] ont conclu que Cplex, un logiciel de commerce, semblait être le plus rapide. Une conclusion plutôt frustrante.

c) Compte tenu, justement, de ce qui vient d'être remarqué, plusieurs heuristiques sont proposées pour obtenir une solution approximative en peu de temps. On distingue des heuristiques qui construisent le recouvrement (voir [27, 79]), des heuristiques qui améliorent, souvent localement, un recouvrement obtenu à l'aide d'une heuristique de construction ([80–82]). Les heuristiques les plus efficaces du point de vue de la qualité du recouvrement obtenu semblent être les heuristiques lagrangiennes [83–86]. Celles-ci relâchent une partie (ou la totalité) des contraintes, puis résolvent le lagrangien dual, et tentent enfin d'exploiter la solution du problème relaxé pour en extraire un recouvrement.

d) Le PRE a aussi été un banc d'essai des méthodes évolutionnaires : recuit simulé ([87, 88]), génétique ([89–91]), électromagnétisme ([92]), réseau de neuronnes ([57]), GRASP ([93]), colonie de fourmis ([94]), probabiliste ([95, 96]), et autres ([39, 97–101]). Des méthodes hybrides existent également ([102, 103]).

f) On trouve des articles qui étudent des questions purement théoriques : arrondi de la solution continue trouvée ([104]), facettes du polytope associé au PRE ([61, 105]), recouvrements disjoints au maximum ([106]), approximation ([107]), programmation par contraintes ([108]).

g) Enfin, on a des tentatives de généralisation du PRE ([109–113]).

2.3 Problème des blocs consécutifs d'une matrice binaire

La seule donnée, dans cette section, est une matrice binaire. On cherche alors une permutation des colonnes qui vérifie une propriété donnée. On verra que si ce problème est difficile même sur des instances très spéciales, il peut être approximé avec une garantie de 50%. On présente ensuite quelques unes de ses applications et un état de l'art.

2.3.1 Position du problème

Posons $I = \{1, \cdots, m\}$ et $J = \{1, \cdots, n\}$. Soit donnée une $m \times n$-matrice binaire $\mathbf{A} = \left\{a_j^i\right\}, i \in I$ et $j \in J$. On verra plus tard, que du point de vue des problèmes qu'on va étudier, on peut sans perte de généralité supposer qu'aucune ligne et aucune colonne de $\mathbf{A}$ n'est nulle. De même, on peut supposer qu'il n'y a ni ligne, ni colonne, formée uniquement de 1.

Definition 2.10. Un bloc de 1 consécutifs (on dira simplement « bloc » par la suite) est une séquence de 1 situés consécutivement sur la même ligne. Formellement, c'est une séquence $a_p^i, a_{p+1}^i, \cdots, a_q^i$ de la ligne i satisfaisant : i) $a_j^i = 1$, $p \leq j \leq q$, ii) soit $p = 1$ soit $a_{p-1}^i = 0$, iii) soit $q = n$ soit $a_{q+1}^i = 0$.

Example 2.7. *Soit*

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

La séquence a_2^1, a_3^1, a_4^1 est un bloc sur la première ligne alors que la séquence $a_1^1, a_2^1, a_3^1, a_4^1$ n'en est pas un car i) n'est pas satisfaite ($a_1^1 = 0$). De même, la séquence a_3^1, a_4^1 n'est

pas un bloc sur la ligne 1 *car ii) n'est pas satisfaite* ($p = 3 \neq 1$ *et* $a^1_{p-1} = a^1_2 \neq 0$). *La matrice* **A** *totalise finalement* 7 *blocs.*

Supposons que π soit une permutation des éléments de l'ensemble J des numéros des colonnes (on peut dire, par abus, que c'est une permutation des colonnes de **A**) et soit $\mathbf{A}_\pi$ la matrice induite.

Example 2.8. *Si* $\pi = id = 1\ 2\ 3\ 4\ 5\ 6$ *est la permutation identité alors* $\mathbf{A}_{id} = \mathbf{A}$, *et si* $\pi = 3\ 2\ 4\ 1\ 5\ 6$ *alors*

$$\mathbf{A}_\pi = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

On voit que $\mathbf{A}_\pi$ *a* $4\,(= m)$ *blocs, soit un bloc par ligne.*

Definition 2.11. On dit qu'une matrice binaire **A** a la propriété de consécutivité des 1 (on adoptera l'abbréviation PC1) s'il existe une permutation π des colonnes de **A** telle que $\mathbf{A}_\pi$ ait au plus un bloc par ligne (donc exactement un bloc par ligne, à cause de notre hypothèse selon laquelle il n'y a pas de ligne nulle).

La matrice de l'exemple précédent a donc la propriété PC1. Hélas, toutes les matrices binaires ne l'ont pas. Il suffit de regarder

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

Considérons le problème suivant.

Reconnaissance de la propriété PC1 **(RPC1)**
Instance $m, n \in \mathbb{N}$ et $\mathbf{A} \in \{0,1\}^{m \times n}$
Question A a-t-elle la propriété PC1 ?

Ce problème a été résolu par Booth et Lueker [114]. En utilisant une structure spéciale, les PQ-arbres, ils ont mis au point un algorithme de reconnaissance linéaire en $\mathrm{O}\,(f)$ où $f = \sum_{ij} a^i_j$. Non seulement cet algorithme résout **RPC1** (en répondant « oui » ou « non ») mais donne également la permutation voulue.

Definition 2.12. Une matrice $\mathbf{A}$ est dite totalement unimodulairesi toute sous-matrice carrée extraite de $\mathbf{A}$ a un déterminant égal à -1 ou 0 ou 1.

Cette définition implique que tout coefficient d'une matrice totalement unimodulaire doit appartenir à $\{-1, 0, 1\}$. La propriété de totale unimodularité est extrêmement importante en optimisation (voir [115, 116]) car si $\mathbf{A}$ est totalement unimodulaire et si $\mathbf{b} \in \mathbb{Z}^m$ alors le polyèdre

$$\{\mathbf{x} \in \mathbb{R}^n | \mathbf{A}\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq 0\}$$

associé à la relaxation linéaire du programme linéaire en nombres entiers

$$\begin{array}{rl} & \max \mathbf{c}^T\mathbf{x} \\ (\mathbf{P}) & \mathbf{A}\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \in \mathbb{Z}^n_+ \end{array}$$

a tous ses points extrêmes entiers. Ainsi, pour résoudre $\mathbf{P}$, un programme linéaire en nombres entiers (problème difficile) , il suffit de résoudre sa relaxation linéaire (facile)

$$\begin{array}{rl} & \max \mathbf{c}^T\mathbf{x} \\ (\mathbf{PL}) & \mathbf{A}\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq 0 \end{array}$$

Proposition 2.13. *[116] Une matrice binaire* $\mathbf{A}$ *ayant la propriété PC1 est totalement unimodulaire.*

Soit $\mathbf{A}$ une matrice binaire et soit $\mathbf{A}_\pi$ la matrice induite par une permutation π de ses colonnes. Le nombre $n(\mathbf{A}_\pi)$ compte tous les coefficients $a^i_{\pi(j)}$ tels que

$$a^i_{\pi(j)} = 1 \text{ et } (\text{soit } a^i_{\pi(j+1)} = 0 \text{ soit } j = n) \tag{2.2}$$

Par exemple, pour la matrice $\mathbf{A}_{id} = \mathbf{A}$ de l'exemple 2.7, quels sont les coefficients vérifiant (2.2) ? On a $a^1_4, a^2_1, a^2_6, a^3_2, a^3_5, a^4_1, a^4_5$. Ainsi $n(\mathbf{A}_{id}) = 7$. De même, si $\pi = 3\ 2\ 4\ 1\ 5\ 6$ alors $n(\mathbf{A}_\pi) = 4$. Ce nombre, on le voit, compte finalement le nombre de blocs en en identifiant la fin (soit un 1 suivi d'un 0, soit un 1 sur la dernière colonne).

Posons maintenant un problème très important, qui va nous préoccuper par la suite. Ce problème est motivé par la question suivante : si $\mathbf{A}$ n'a pas la propriété PC1, peut-on

trouver une permutation π de sorte que $\mathbf{A}_\pi$ soit la plus « proche » possible d'avoir cette propriété. Ce problème est appelé *Consecutive Block Minimization* dans la littérature. A défaut d'avoir trouvé son nom en français, on l'a appelé « problème de minimisation du nombre de blocs consécutifs (**PBC**) ». Le problème de décision associé à **PBC** est

---

PBC-décision

Instance $m, n \in \mathbb{N}$, $\mathbf{A} \in \{0,1\}^{m \times n}$ et $s \in \mathbb{N}$ donné

Question Existe-t-il π telle que $n(\mathbf{A}_\pi) \leq s$?

---

et sa version d'optimisation

PBC

Instance $m, n \in \mathbb{N}$ et $\mathbf{A} \in \{0,1\}^{m \times n}$

Question Chercher π telle que $n(\mathbf{A}_\pi)$ soit minimum

---

2.3.2 Complexité et approximation

Proposition 2.14. *[117]* ***PBC-décision*** *est* $\mathbb{NP}$*-complet.*

PBC est donc $\mathbb{NP}$-dur. Haddadi montre qu'il reste difficile même sur des instances très particulières.

Proposition 2.15. *[118]* ***PBC*** *est* $\mathbb{NP}$*-dur même lorsque la matrice binaire n'a que deux* 1 *par ligne.*

Pour la preuve il utilise la transformation décrite ci-après qui permet de ramener polynomialement le problème de la chaîne hamiltonienne de longueur maximum sur un graphe complet dont les longueurs w_{ij} des arêtes sont restreintes aux valeurs $1, 2$ (qui est clairement $\mathbb{NP}$-dur) à **PBC** restreint aux matrices binaires ayant seulement deux 1 par ligne.

---

procedure *transformation*
input : $n \times n$-matrice $\mathbf{W}$ telle que $w_{ii} = 0$ et $w_{ij} = w_{ji} \in \{1,2\}, i \neq j$
output : $r \times n$-matrice binaire $\mathbf{A}$ ayant deux 1 par ligne
begin
$r \leftarrow \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_j^i$; $t \leftarrow 0$;
for $i = 1$ **to** r **do for** $j = 1$ **to** n **do** $A_j^i \leftarrow 0$;
for $i = 1$ **to** $n-1$ **do for** $j = i+1$ **to** n **do for** $k = 1$ **to** w_j^i **do**
begin $t \leftarrow t+1$; $A_i^t \leftarrow 1$; $A_j^t \leftarrow 1$ **end**
end

Considérons un autre nombre $N(\mathbf{A}_\pi)$ associé à la matrice $\mathbf{A}$. C'est le nombre de coefficients a^i_j de $\mathbf{A}_\pi$ tels que

$$a^i_{\pi(j)} = 1 \text{ et } (\text{soit } a^i_{\pi(j+1)} = 0 \text{ soit } (j = n \text{ et } a^i_{\pi(1)} = 0))$$

qui compte le nombre de blocs de 1 circulaires. Imaginez que la matrice soit posée sur un cylindre avec les colonnes 1 et n adjacentes. En langage courant, un bloc de 1 circulaires est soit un bloc de 1 consécutifs soit l'union de deux blocs consécutifs (séparés par au moins un 0) avec l'un débutant à la colonne 1 et l'autre finissant à la colonne n. On voit que la matrice de l'exemple 2.7 a 6 blocs de 1 circulaires et que la matrice

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

en a 3. Posons le problème des 1 circulaires.

CIR

Instance $m, n \in \mathbb{N}$ et $\mathbf{A} \in \{0,1\}^{m \times n}$

Question Chercher π telle que $N(\mathbf{A}_\pi)$ soit minimum

Posons le problème de l'approximation de **PBC**. Existe-t-il une α-approximation pour **PBC** ? Haddadi a donné la réponse, en résolvant un problème qui était ouvert.

Proposition 2.16. *[119] **PBC** est* 1.5*-approximable.*

Pour la preuve, il utilise deux transformations. La première transforme polynomialement **PBC** à **CIR**. La seconde ramène polynomialement **CIR** au problème du voyageur de commerce (voir [120]). Comme ces transformations préservent les rapports d'approximation, et comme l'instance du voyageur de commerce obtenue vérifie l'inégalité triangulaire, alors l'application séquentielle de ces deux ransformations puis de l'heuristique de Christofides [121] fournit une 1.5-approximation pour **PBC**.

2.3.3 Applications et état de l'art

Le problème **PBC** est bien connu de la communauté scientifique de par ses nombreuses applications. Pour l'illustration, on va en décrire une. Celle de la recherche et extraction d'information [122] en organisant les fichiers de manière efficace sur un support.

On a n enregistrements $e_1, \cdots, e_n$ sur un support donné et m requêtes $r_1, \cdots, r_m$ à satisfaire. Chaque requête demande un ensemble d'enregistrements.

$$\begin{array}{cccccc} & e_1 & e_2 & \cdots & e_n \\ r_1 & 1 & 1 & & \\ r_2 & 0 & 1 & & \\ \vdots & \vdots & & \ddots & \\ r_n & 1 & 0 & \cdots & \end{array}$$

La matrice binaire du problème est formée de l'élément général a^i_j défini par 1 si la requête r_i demande l'enregistrement e_j et 0 sinon. Le problème se ramène à chercher une réorganisation des enregistrements la plus linéaire possible, c'est-à-dire de sorte que les requêtes soient satisfaites par des enregistrements qui soient le plus possible contigus (avec le moins possible de coupures).

On retrouve des applications également en ordonnancement ([123]), en cartographie ([124, 125]), sériation ([126]), radiothérapie ([127]), microbiologie ([128]), organisation de fichiers ([129]), compression ([130, 131]), archéologie ([132]) et génétique ([133]). Des questions théoriques sont abordées. Bendali-Amor et Quilliot [134] étudient la compatibilité entre la notion de relation d'ordre et celle de matrice ayant la propriété PC1. Fulkerson et Gross ([135]) caractérisent un graphe d'intervalle en termes de matrice ayant la propriété PC1. Tucker [136] et Narayaswamy et Subashini [137] en donnent d'autres caractérisations. Oswald et Reinelt [138] abordent des aspects de la théorie polyédrale en rapport avec PBC. Plusieurs considérations algorithmiques sont examinées dans [139–142].

Le problème **CIR** est également étudié dans [143, 144]. Telles et Meidanis reprennent ce qui a été fait par Booth et Lueker en proposant la structure des PQR-arbres.

Plusieurs problèmes connexes sont posés et étudiés. Hajiaghayi et Ganjali [145] étudient le problème (connu comme étant $\mathbb{NP}$-complet [24]) consistant à chercher le plus petit entier positif k tel qu'il existe une sous-matrice de $\mathbf{A}$ d'ordre $m \times k$ qui ait la propriété PC1. Dom et al ([146]) étudient l'approximabilité et la complexité paramétrique de ce problème. Veldhorst [147] s'attaque au problème (connu aussi comme étant $\mathbb{NP}$-complet [24]) de chercher le nombre minimum de 0 à remplacer par des 1 de sorte que la matrice binaire obtenue ait la propriété PC1, et montre qu'il ne peut être approximé avec garantie

à moins que $\mathbb{P} = \mathbb{NP}$. Et comme d'habitude, on étudie des cas particuliers ([148]) et des généralisations ([149, 150]).

Chapitre 3

Outils théoriques

Ce chapitre présente un condensé des concepts utiles à la compréhension de notre contribution que sont les chapitres 4 et 6. Sa rédaction repose sur de nombreuses références bibliographiques.

3.1 Méthode de décomposition de Benders

Soit donné un programme mathématique, très souvent un programme linéaire mixte, c'est-à-dire dont certaines variables sont continues et d'autres discrètes. La méthode de décomposition de Benders [151] procède en deux étapes : – d'abord scinder le programme en sous-problèmes ayant chacun une structure spéciale, donc relativement faciles à résoudre, mais liés entre eux par des variables dites couplantes ; – faire coopérer les sous-problèmes sous la coordination d'un programme-maître (dont les variables de décision sont les variables couplantes). C'est la démarche classique de décentralisation, celle par exemple d'un chef d'entreprise collaborant avec ses chefs de département : chaque département est plus facile à gérer que l'entreprise globale, mais les départements ne sont pas libres de faire ce qu'ils veulent, ils doivent collaborer pour une politique globale sous la direction du chef d'entreprise.

De nombreuses publications présentent soit la méthode et ses prolongements ([152–154]), soit des applications sur des problèmes concrets dans plusieurs domaines tels que

la conception de réseaux ([155–158]), l'affectation ([36, 159]), le routage et l'ordonnancement ([160, 161]), et divers ([162]). La rédaction de cette section s'inspire essentiellement de la référence [163].

3.1.1 Programme mixte

A l'origine, Benders [151] a conçu cette méthode pour un programme mixte du type

$$(\mathbf{P}) \begin{cases} \min \mathbf{c}^T \mathbf{x} + f(\mathbf{y}) \\ \mathbf{B}\mathbf{x} + F(\mathbf{y}) \geq \mathbf{b} \\ \mathbf{x} \geq \mathbf{0}, \mathbf{y} \in S \end{cases}$$

avec pour données une $m \times n$-matrice $\mathbf{B}$ à coefficients réels, $\mathbf{c} \in \mathbb{R}^n, \mathbf{b} \in \mathbb{R}^m, F$ un vecteur de $\mathbb{R}^m$ dont chaque composante est une fonction continue sur S et f une fonction continue sur S. On a deux vecteurs de variables $\mathbf{x} \in \mathbb{R}^n, \mathbf{y} \in \mathbb{R}^p$. L'ensemble S est un sous-ensemble arbitraire de $\mathbb{R}^p$ (souvent $S = \mathbb{Z}_+^p$).

Observons que le problème $\mathbf{P}$ serait un simple programme linéaire, donc facile à résoudre, s'il ne mettait en jeu que le vecteur $\mathbf{x}$. C'est le vecteur $\mathbf{y}$ des variables $y_1, \cdots, y_p$ qui est compliquant. L'idée géniale de Benders est alors de séparer itérativement le traitement des variables $x_1, \cdots, x_n$ de celui de $y_1, \cdots, y_p$ en deux étapes :

- le vecteur $\mathbf{y}$ étant fixé, résoudre un programme linéaire en $\mathbf{x}$;
- résoudre un problème en $\mathbf{y}$ par approximations affines successives. Les contraintes affines rajoutées à chaque itération sont appelées « coupes de Benders ».

3.1.2 Génération de l'enveloppe affine

Considérons le problème d'optimisation

$$(\mathbf{PM}) \quad \min \{ f(\mathbf{y}) + g(\mathbf{y}) \, | \mathbf{y} \in S \}$$

où la fonction f est déjà définie, et $g : S \mapsto \mathbb{R}$.

Proposition 3.1. *[151] Les problèmes **P** et **PM** sont équivalents si et seulement*

$$g(\mathbf{y}) = \min \left\{ \mathbf{c}^T \mathbf{x} | \mathbf{B}\mathbf{x} \geq \mathbf{b} - F(\mathbf{y}), \mathbf{x} \geq \mathbf{0} \right\} \tag{3.1}$$

Pour quelles valeurs de $\mathbf{y}$ la fonction $g(\mathbf{y})$ est-elle définie ? Clairement, son domaine de définition est

$$\mathcal{D}(g) = \{\mathbf{y} \in S | \text{il existe } \mathbf{x} \geq \mathbf{0} \text{ tel que } \mathbf{Bx} \geq \mathbf{b} - F(\mathbf{y})\}$$

Le lemme de Farkas (dont on peut trouver la preuve dans [116]) va nous permettre d'expliciter ce domaine.

Lemma 3.2. *Il existe* $\mathbf{x} \geq \mathbf{0}$ *tel que* $\mathbf{Ax} = \mathbf{b}$ *si et seulement si pour tout* $\mathbf{u}$ *tel que* $\mathbf{A}^T\mathbf{u} \geq \mathbf{0}$ *on a* $\mathbf{b}^T\mathbf{u} \geq \mathbf{0}$.

Pour $\mathbf{y}$ fixé, on peut ramener le système de m inégalités $\mathbf{Bx} \geq \mathbf{b} - F(\mathbf{y})$ à un système de m équations par adjonction de m variables d'écart non négatives. Soit $\mathbf{e} = (e_1, \cdots, e_m)$ le vecteur des variables d'écart. De cette manière, $\mathbf{Bx} \geq \mathbf{b} - F(\mathbf{y})$ est équivalent à

$$\mathbf{Bx} - \mathbf{e} = \mathbf{b} - F(\mathbf{y}), \mathbf{e} \geq \mathbf{0} \tag{3.2}$$

En appliquant le lemme de farkas au système en (3.2), on obtient que pour tout $\mathbf{u} \geq \mathbf{0}$ tel que $\mathbf{B}^T\mathbf{u} \leq \mathbf{0}$ on a $(\mathbf{b} - F(\mathbf{y}))^T \mathbf{u} \leq \mathbf{0}$. Par ailleurs, le cône polyédral convexe

$$\left\{\mathbf{u} | \mathbf{B}^T\mathbf{u} \leq \mathbf{0}, \mathbf{u} \geq \mathbf{0}\right\}$$

a un nombre fini de générateurs (ou directions extrémales). Appelons-les $\mathbf{v}_1, \cdots, \mathbf{v}_p$. D'où la représentation explicite du domaine de définition de g

$$\mathcal{D}(g) = \left\{\mathbf{y} \in S | (\mathbf{b} - F(\mathbf{y}))^T \mathbf{v}_i \leq \mathbf{0}, i = 1, \cdots, p\right\}$$

Comme on a défini g en (3.1), pour $\mathbf{y}$ fixé, comme valeur d'un programme linéaire, on peut très bien, de façon équivalente, la définir comme valeur de son dual

$$g(\mathbf{y}) = \max\left\{(\mathbf{b} - F(\mathbf{y}))^T \mathbf{u} | \mathbf{B}^T\mathbf{u} \leq \mathbf{c}, \mathbf{u} \geq \mathbf{0}\right\} \tag{3.3}$$

Or, les contraintes de ce dual en (3.3) forment un polyèdre

$$\mathcal{P} = \left\{\mathbf{u} | \mathbf{B}^T\mathbf{u} \leq \mathbf{c}, \mathbf{u} \geq \mathbf{0}\right\}$$

indépendant de $\mathbf{y}$, et dont tout point peut être exprimé en fonction des points extrêmes qui sont en nombre fini. Soient $\mathbf{u}_1, \cdots, \mathbf{u}_q$ ces points extrêmes. Le problème **PM**, appelé programme-maître, s'écrit alors (dans sa forme duale)

$$(\mathbf{PM}) \begin{cases} \min z & \\ (\mathbf{b} - F(\mathbf{y}))^T \mathbf{v}_i & \leq 0 \quad i = 1, \cdots, p \\ f(\mathbf{y}) + (\mathbf{b} - F(\mathbf{y}))^T \mathbf{u}_j & \leq z \quad j = 1, \cdots, q \\ \mathbf{y} \in S & \end{cases}$$

Par construction, ce problème est équivalent à **P**. Il a un trop grand nombre de contraintes (p et q sont très grands) pour être considéré tel quel. Pratiquement, on débute avec seulement quelques contraintes (ou pas du tout) et on en génère d'autres par résolutions successives du programme dual en (3.3) tout en espérant que l'algorithme convergera après l'adjonction seulement d'un petit nombre de coupes. La qualité de tout algorithme du type Benders dépend essentiellement de l'efficacité de la routine de résolution du programme-maître.

3.1.3 Algorithme

L'algorithme est fondé sur la coopération de deux programmes. Le premier est appelé « sous-programme » (c'est un programme linéaire)

$$(\mathbf{SP}) \begin{cases} \max (\mathbf{b} - F(\mathbf{y}))^T \mathbf{u} & \\ \mathbf{B}^T \mathbf{u} & \leq \mathbf{0} \\ \mathbf{u} & \geq 0 \end{cases}$$

et le second est appelé « programme-maître réduit »

$$(\mathbf{PMR}) \begin{cases} \min z & \\ (\mathbf{b} - F(\mathbf{y}))^T \mathbf{v}_i & \leq 0 \quad i \in I \\ f(\mathbf{y}) + (\mathbf{b} - F(\mathbf{y}))^T \mathbf{u}_j & \leq z \quad j \in J \\ \mathbf{y} \in S & \end{cases}$$

Ce dernier est dit réduit car il ne contient que quelques unes des contraintes du programme-maître **PM**. Ici $I \subset \{1, \cdots, p\}$ et $J \subset \{1, \cdots, q\}$. En pratique, les cardinalités de I et J

doivent être très petites relativement à p et q respectivement, sans quoi la résolution de **PMR** serait problématique.

L'algorithme est décrit dans ce qui suit.

Algorithme de Benders

Initialisation

$I \longleftarrow \varnothing$

$J \longleftarrow \varnothing$

Choisir $\mathbf{y}^{(0)} \in S$ arbitraire

$z^{(0)} \longleftarrow -\infty$

$k \longleftarrow 0$

Boucle

* Résoudre le problème **SP** avec $\mathbf{y} = \mathbf{y}^{(k)}$

– Si le polyèdre $\mathcal{P} = \varnothing$ alors STOP : le problème P n'a pas de solution optimale finie.

– Si le problème **SP** n'a pas de solution optimale finie, c'est qu'on a trouvé un rayon extrémal le long duquel la fonction objectif augmente indéfiniment. Soit $\mathbf{v}_i$ cette direction extrémale. Posons $I \longleftarrow I \cup \{i\}$

– Si $\left(\mathbf{b} - F\left(\mathbf{y}^{(k)}\right)\right)^T \mathbf{u}_j = z^{(k)} - f\left(\mathbf{y}^{(k)}\right)$ alors STOP : si $\mathbf{x}^{(k)}$ est la solution duale associée au problème **SP** alors $\left(\mathbf{x}^{(k)}, \mathbf{y}^{(k)}\right)$ est solution optimale du problème **P** ;

– Sinon on pose $J \longleftarrow J \cup \{j\}$

* Résoudre le problème **PMR**

– Si **PMR**n'a pas de solution alors STOP : Le problème n'en a pas non plus.

– Si **PMR**n'a pas de solution optimale finie, posons $z^{(k)} \longleftarrow -\infty$ et $\mathbf{y}^{(k+1)} \longleftarrow \mathbf{y}^{(k)}$

–Sinon soit $\left(z^{(k+1)}, \mathbf{y}^{(k+1)}\right)$ une solution optimale

* $k \longleftarrow k + 1$

Deux remarques pour finir.

1. L'algorithme de Benders est fini car le nombre q de points extrêmes et le nombre p de rayons extrémaux du polyèdre $\mathcal{P}$ sont finis.
2. La suite $\left\{z^{(k)}\right\}$ est croissante puisque à chaque itération l'on rajoute une contrainte et, par conséquent, $\left(z^{(k+1)}, \mathbf{y}^{(k+1)}\right)$ est une solution réalisable pour le programme-maître de l'itération k. Donc $z^{(k+1)} \geq z^{(k)}$.

3.2 Relaxation lagrangienne et méthode de sous-gradients

Un fait bien connu est que la plupart des problèmes d'optimisation difficiles sont constitués de contraintes faciles à « prendre en compte » auxquelles sont ajoutées des contraintes

compliquantes. Les contraintes difficiles sont alors dualisées, ce qui produit un problème appelé dual lagrangien, qu'on résout souvent à l'aide d'une méthode de sous-gradients, et dont la solution constitue une borne inférieure (dans un cas de minimisation) de la valeur du problème posé. Cette section présente la relaxation lagrangienne dans un cadre général (voir [164–168]). Pour une application de cette technique à un problème précis, le PRE, voir le chapitre 4.

3.2.1 Problème primal

Soient $\mathbf{A} \in \mathbb{R}^{m\times n}, \mathbf{D} \in \mathbb{R}^{p\times n}, \mathbf{b} \in \mathbb{R}^m, \mathbf{e} \in \mathbb{R}^p, \mathbf{c} \in \mathbb{R}^n$ donnés et $\mathbf{x} \in \mathbb{R}^n$ un vecteur de variables. On pose le problème d'optimisation combinatoire

$$(\mathbf{P}) \begin{cases} z = & \min \mathbf{c}^T\mathbf{x} & \\ & \mathbf{Ax} & = \mathbf{b} \\ & \mathbf{Dx} & \leq \mathbf{e} \\ & \mathbf{x} & \in \mathbb{Z}^n_+ \end{cases}$$

Soit le problème

$$(\mathbf{R}) \begin{cases} z_R = & \min \mathbf{c}^T\mathbf{x} & \\ & \mathbf{Ax} & = \mathbf{b} \\ & \mathbf{Dx} & \leq \mathbf{e} \\ & \mathbf{x} & \geq \mathbf{0} \end{cases}$$

obtenu en relâchant les contraintes d'intégralité de $\mathbf{P}$, qui est appelé relaxation linéaire de $\mathbf{P}$.

Les contraintes faciles sont $\mathbf{Dx} \leq \mathbf{e}$ (par exemple $\mathbf{D}$ est une matrice graphique totalement unimodulaire). Autrement dit le problème

$$\begin{cases} z = & \min \mathbf{c}^T\mathbf{x} & \\ & \mathbf{Dx} & \leq \mathbf{e} \\ & \mathbf{x} & \in \mathbb{Z}^n_+ \end{cases}$$

est un programme linéaire qu'on résout polynomialement. Mais quand on ajoute les contraintes $\mathbf{Ax} = \mathbf{b}$ le problème durcit.

3.2.2 Le dual et ses propriétés

L'idée est alors de dualiser les m contraintes difficiles en associant à chacune un multiplicateur de Lagrange et en définissant la fonction de Lagrange (ou problème relaxé), avec $\pi \in \mathbb{R}^m$

$$(\mathbf{PR}(\pi)) \begin{cases} w(\mathbf{x},\pi) = & \min \mathbf{c}^T\mathbf{x} + \pi^T(\mathbf{Ax}-\mathbf{b}) \\ & \mathbf{Dx} & \leq \mathbf{e} \\ & \mathbf{x} & \in \mathbb{Z}_+^n \end{cases}$$

Pour simplifier l'exposé, on peut supposer, sans perte de généralité, que :

1. le problème **P** a toujours une solution. Ce qui signifie que

$$\left\{\mathbf{x} | \mathbf{Ax} = \mathbf{b}, \mathbf{Dx} \leq \mathbf{e}, \mathbf{x} \in \mathbb{Z}_+^n\right\} \neq \varnothing$$

2. L'ensemble

$$S = \left\{\mathbf{Dx} \leq \mathbf{e}, \mathbf{x} \in \mathbb{Z}_+^n\right\}$$

des solutions réalisables de $\mathbf{PR}(\pi)$ a un nombre fini de points. Cette seconde hypothèse a pour but d'avoir une valeur $w(\mathbf{x},\pi)$ finie pour n'importe quel π.

Supposons que $\mathbf{x}^*$ soit une solution optimale de **P** et $z^* = \mathbf{c}^T\mathbf{x}^*$. Il est facile de voir que $w(\mathbf{x},\pi) \leq z^*$ pour tout π en observant que

$$w(\mathbf{x},\pi) = \min \mathbf{c}^T\mathbf{x} + \pi^T(\mathbf{Ax}-\mathbf{b}) \leq \mathbf{c}^T\mathbf{x}^* + \pi^T(\mathbf{Ax}^*-\mathbf{b}) = z^*$$

puisque $\mathbf{Ax}^* - \mathbf{b} = \mathbf{0}$.

Définissons maintenant le lagrangien dual de **P**. C'est le problème de déterminer une valeur θ telle que

$$(\mathbf{DL})\ \theta = \max_{\pi\in\mathbb{R}^m} L(\pi) = \max_{\pi\in\mathbb{R}^m} \min_{\mathbf{x}\in S} w(\mathbf{x},\pi)$$

L'intérêt de la méthode de relaxation lagrangienne réside dans les propriétés suivantes de la fonction lagrangienne $L(\pi)$.

Propriété 1. [167] Pour tout $\mathbf{x} \in S$ et pour tout $\pi \in \mathbb{R}^m$ on a $L(\pi) \leq z$.

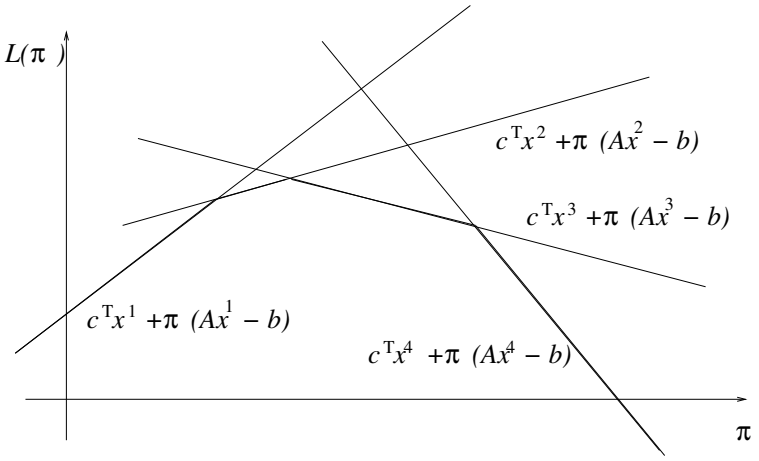

FIG. 3.1: Fonction L linéaire par morçeaux en π

En particulier, $L(\pi) \leq z^*$, et donc pour tout π, $L(\pi)$ constitue une borne inférieure de la valeur optimale z^* de **P**. L'idéal est de trouver θ la meilleure borne inférieure possible, en résolvant **DL**.

Propriété 2. [167] La fonction $L(\pi)$ est une fonction linéaire par morceaux et concave.

En effet, supposons que $|S| = r$ et soient $\mathbf{x}^1, \cdots, \mathbf{x}^r$ ses éléments. On peut voir que $L(\pi)$ est l'enveloppe inférieure de r hyperplans dans $\mathbb{R}^{m+1}$ (variables π et z)

$$\begin{aligned} z &= \mathbf{c}^T\mathbf{x}^1 + \pi^T\left(\mathbf{A}\mathbf{x}^1 - \mathbf{b}\right) \\ &\vdots \\ z &= \mathbf{c}^T\mathbf{x}^r + \pi^T\left(\mathbf{A}\mathbf{x}^r - \mathbf{b}\right) \end{aligned}$$

Pour $\pi \in \mathbb{R}^m$ posons

$$T(\pi) = \left\{\mathbf{x}^k \in S | \mathbf{c}^T\mathbf{x}^k + \pi^T\left(\mathbf{A}\mathbf{x}^k - \mathbf{b}\right) = L(\pi)\right\}$$

Un exemple d'une fonction L à une seule variable π linéaire par morceaux est donné à la figure 3.1. Dans cet exemple, il y a une seule contrainte $\mathbf{A}\mathbf{x} = b$ et donc $\mathbf{A}$ est un vecteur-ligne de $\mathbb{R}^n$ et b un réel arbitraire.

Definition 3.3. Un m-vecteur $\mathbf{u}$ est un sous-gradient de la fonction concave L en $\pi \in \mathbb{R}^m$ si

$$L(\mathbf{t}) - L(\pi) \leq \mathbf{u}^T(\mathbf{t} - \pi)$$

pour tout $\mathbf{t} \in \mathbb{R}^m$. L'ensemble, noté $\delta L(\pi)$, de tous les sous-gradients en π est appelé sous-différentiel de L en π. C'est un ensemble compact et convexe [169].

Proposition 3.4. *[169] Une condition nécessaire et suffisante pour que la fonction L ait un maximum en un point π est que $\mathbf{0} \in \delta L(\pi)$.*

Propriété 3. [167] Pour tout $\mathbf{x}^k \in T(\pi)$ le vecteur $\mathbf{A}\mathbf{x}^k - \mathbf{b}$ est un sous-gradient de L en π.

Autrement dit, pour tout $\pi \in \mathbb{R}^m$ et pour toute solution $\mathbf{x}^k$ du problème $\mathbf{PR}(\pi)$ on a $\mathbf{A}\mathbf{x}^k - \mathbf{b}$ qui est un sous-gradient de L en π. Ce qui revient aussi à dire que pour $\pi \in \mathbb{R}^m$ l'ensemble

$$\left\{\mathbf{A}\mathbf{x}^k - \mathbf{b} | \mathbf{x}^k \in T(\pi)\right\}$$

est une partie du sous-différentiel $\delta L(\pi)$ de L en π.

Proposition 3.5. *[167] Pour $\pi \in \mathbb{R}^m$ les vecteurs $\mathbf{A}\mathbf{x}^k - \mathbf{b}$, pour $\mathbf{x}^k \in T(\pi)$, forment un ensemble générateur du sous-différentiel $\delta L(\pi)$.*

Ainsi donc, tout sous-gradient de L en π est une combinaison convexe des vecteurs $\mathbf{A}\mathbf{x}^k - \mathbf{b}, \mathbf{x}^k \in T(\pi)$.

La fonction $L(\pi)$, on vient de le voir, possède toutes les propriétés requises pour son optimisation, sauf une. Hélas, étant linéaire par morceaux, elle n'est pas partout différentiable. Plus précisément, elle est différentiable sauf en un nombre fini de points.

3.2.3 Résolution du dual par la méthode de sous-gradients

On peut réécrire le dual lagrangien **DL** ainsi (en vertu de la propriété 2) :

$$(\overline{\mathbf{D}}) \quad \begin{array}{l} \theta = \max L \\ L \leq \mathbf{c}^T\mathbf{x}^k + \pi^T\left(\mathbf{A}\mathbf{x}^k - \mathbf{b}\right) \quad k = 1, \cdots, r \end{array}$$

Le problème $\overline{\mathbf{D}}$ est un programme linéaire en L et π, dont le dual est

$$(\overline{\mathbf{P}}) \quad \begin{array}{rll} \theta = & \min \sum_{k=1}^r \lambda_k \mathbf{c}^T\mathbf{x}^k & \\ & \sum_{k=1}^r \lambda_k \mathbf{A}\mathbf{x}^k = \mathbf{b} & k = 1, \cdots, r \\ & \sum_{k=1}^r \lambda_k = 1 & \\ & \lambda_k \geq 0 & k = 1, \cdots, r \end{array}$$

Pour trouver la valeur θ, il faut résoudre $\overline{\mathbf{P}}$, un programme linéaire. Mais, pratiquement, ce dernier a un nombre astronomique de contraintes. On utilise alors pour le résoudre une technique bien connue, appelée méthode de génération de contraintes. On voit bien que chercher la valeur exacte de θ peut être prohibitif.

Souvent, on se contente d'une valeur approchée de θ qu'on obtient par un procédé itératif. Puisque la fonction L est concave alors tout maximum local est global, et on peut se contenter de chercher un maximum local en utilisant la méthode de sous-gradients (qui mime la méthode de gradients), en débutant avec $\pi^{(0)}$ et en générant une séquence $\left\{\pi^{(s)}\right\}$ avec la règle

$$\pi^{(s+1)} = \pi^{(s)} + t_k \left(\mathbf{A}\mathbf{x}^k - \mathbf{b}\right)$$

où $\mathbf{x}^k$ est une solution optimale de $\mathbf{PR}\left(\pi^{(k)}\right)$, $\mathbf{A}\mathbf{x}^k - \mathbf{b}$ un sous-gradient de L en $\pi^{(k)}$, et t_k un nombre positif qui est le pas de progression dans la direction du sous-gradient.

Held et al [169] ont obtenu une condition suffisante de convergence de la séquence $\left\{\pi^{(s)}\right\}$ qui leur a permis de déterminer le pas t_k de façon originale, qu'il sont expérimentalement validé, et qui a été validé par de nombreux autres travaux par la suite.

Proposition 3.6. *[169] Si* $t_k \longrightarrow 0$ *et si*

$$\sum_{i=0}^{k} t_k \longrightarrow \infty$$

alors $L\left(\pi^{(k)}\right) \longrightarrow \theta$.

Ils proposèrent alors de prendre pour pas

$$t_k = \lambda_k \frac{z^* - L\left(\pi^{(k)}\right)}{\left\|\mathbf{A}\mathbf{x}^k - \mathbf{b}\right\|^2}$$

où z^* est la valeur optimale de $\mathbf{P}$, $L\left(\pi^{(k)}\right)$ la valeur optimale de $\mathbf{PR}\left(\pi^{(k)}\right)$ et $\left\|\mathbf{A}\mathbf{x}^k - \mathbf{b}\right\|^2$ la norme au carré du sous-gradient L en $\pi^{(k)}$. Comme la valeur de z^* est à priori inconnue, on peut se contenter d'une estimation. En pratique, on prend une borne supérieure de z^*, en cherchant une bonne solution réalisable à l'aide d'une heuristique et en prenant pour estimation de z^* la valeur de cette solution.

Le coefficient λ_k est appelé coefficient de relaxation. L'usage est de débuter avec $\lambda_0 = 2$ puis le diviser par 2 après β itérations, puis $\lceil \beta/2 \rceil$ itérations, etc. Finalement, on prend

comme meilleure approximation de θ la valeur

$$\max_{k\geq 0} L\left(\pi^{(k)}\right) \tag{3.4}$$

Remark 3.7. La relaxation lagrangienne (et la borne θ produite par la méthode de sous-gradients en (3.4)) a un intérêt considérable en optimisation.

1. Elle fournit une borne inférieure fondamentale pour une méthode Branch-and-Bound.
2. La résolution du problème relaxé $\mathbf{PR}(\pi^{(k)})$ donne une information utile à la procédure de séparation dans une méthode Branch-and-Bound.
3. Une bonne solution réalisable du problème $\mathbf{P}$ peut être obtenue par perturbation de la solution de $\mathbf{PR}(\pi^{(k)})$ à n'importe quelle itération k de la méthode de sous-gradients.

3.3 Classes de complexité

Puisque dans cette thèse on évoque la complexité de certains problèmes d'optimisation combinatoire, le but de cette section est d'introduire, sans formalisme, la notion de complexité d'un problème. Celle-ci a pour but d'évaluer les algorithmes en termes d'« efficacité » et de discriminer les problèmes en ceux qui sont « faciles » et ceux qui sont « difficiles ou durs » selon qu'ils peuvent être résolus par des algorithmes efficaces ou non. Ici, algorithme et problème ne sont pas seulement deux mots du métalangage mathématique, mais deux objets mathématiques que l'on étudie. La rédaction de cette section s'inspire substantiellement de la référence [170].

3.3.1 Problèmes indécidables et problèmes intraitables

Le problème d'existence d'un algorithme pour un problème donné suppose que les ressources disponibles (temps et espace de calcul) sont illimitées. Cette question entre dans le cadre de la théorie de la calculabilité. Elle nous permet de comprendre les possibilités et, surtout, les limites de l'ordinateur. Précisément, l'ordinateur ne peut pas être utilisé pour tout faire.

Vers le début du XX^{e} siècle, sur la base de travaux en logique formelle, plusieurs mathématiciens commencèrent à s'intéresser aux propriétés mathématiques des algorithmes. Turing [171] conceva astucieusement une machine abstraite qui constitua le fondement théorique de l'ordinateur actuel, et qui influença la perception de ce qui est essentiel dans un ordinateur. Il considéra alors un problème, devenu célèbre, qui pose la question de savoir si, une fois lancé, son automate était capable de finir les calculs et s'arrêter. En analysant ce problème, il prouva de manière fondamentale qu'il ne peut exister d'algorithme permettant d'affirmer qu'un énoncé mathématique est vrai ou faux [171]. Turing démontra par là les limites de son automate : il y a donc des problèmes qui ne peuvent être résolus ni par la machine de Turing, ni par conséquent par l'ordinateur actuel. De tels problèmes sont dits indécidablesou sans preuve. Un autre exemple est fourni dans la référence [172]. Il est possible que des cas particuliers de ces problèmes puissent être résolus par tâtonnement mais jamais par un algorithme. Les problèmes indécidables sont donc définitivement insolubles.

Alors que le terme d'indécidabilité provient du jargon de l'informatique, le terme sans preuve provient de celui de la logique. Il y a des énoncés qui ne peuvent avoir aucune preuve. Prenons un exemple : tout nombre entier positif pair est somme de deux nombres premiers. Par exemple $8 = 3 + 5$, $28 = 11 + 17$. Il est prouvé [173] que cet énoncé est ... sans preuve. On ne peut montrer ni qu'il est vrai ni qu'il est faux : il est sans preuve ou indécidable.

En principe, en dehors des problèmes indécidables, les problèmes qui restent sont justiciables d'un algorithme et peuvent être résolus automatiquement. Cependant, on sait que certains problèmes (en logique mathématique, en théorie des automates, en théorie des langages formels, jeux mathématiques) ne peuvent être résolus que par des algorithmes « exponentiels », c'est-à-dire qui consomment un temps démesurément long. Les preuves apportées montrent que ces problèmes ne peuvent pas être résolus par un algorithme efficace même au cas extrême où l'on utilise un ordinateur non déterministe capable de poursuivre un nombre illimité de calculs en parallèle. Ils sont donc intrinsèquement difficiles (on dit intraitablesou à preuve difficile). Des exemples de problèmes intraitables sont présentés dans les références [174–176].

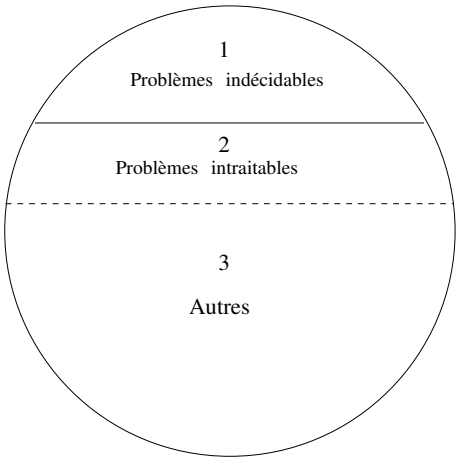

FIG. 3.2: Première tentative de classification des problèmes

3.3.2 Cadre de la théorie de la complexité

Contrairement à la théorie de la calculabilité, la théorie de la complexité considère que les ressources disponibles sont en quantité limitée. La question qu'elle se pose est alors de savoir si un problème donné peut être résolu en un temps raisonnable. Or, parmi les problèmes du groupe 3 de la figure 3.2, il y en que l'on sait résoudre à l'aide d'algorithmes efficaces, et puis il y en a que l'on ne sait résoudre qu'à l'aide de méthodes qui demandent un temps exagérément long. Ce qui ne veut pas dire qu'un algorithme efficace n'existe pas pour ces derniers : la frontière entre les groupes 2 et 3 est floue.

L'étude formelle de la complexité des algorithmes et des problèmes suppose un modèle de calcul (souvent la machine de Turing) et une chaîne de caractères dans un alphabet fini (qui constitue le codage de l'instance en machine). L'analyse de la complexité d'un algorithme consiste alors à calculer le temps d'exécution $t(n)$ comme le nombre de déplacements de la tête de lecture/écriture de la machine de Turing en fonction de la longueur n de la chaîne de caractères qui décrit l'instance. On dit, dans ce cas, que l'on calcule la complexité dans le modèle de Turing. Cette analyse est fastidieuse. Souvent, on se contente de calculer le temps de calcul $t(n)$ comme fonction du nombre n d'opérations élémentaires (addition, soustraction, multiplication, division, comparaison, etc.). Cette complexité est appelée complexité dans le modèle arithmétique.

3.3.3 Problème de décision et complémentaire

Un problème de décisionest un problème dont la réponse à la question posée est « oui » ou « non ». A titre d'exemple, considérons le problème de décision suivant. On se donne une matrice carrée A d'ordre n et un vecteur $\mathbf{b}$ de dimension n, tous deux à coefficients réels, et on pose la question de savoir si le système à n équations et n inconnues correspondant a une solution ou non.

Système d'équations linéaires (**SEL**)
Instance Entier $n \geq 2$, $\mathbf{A} \in \mathbb{R}^{n \times n}$ et $\mathbf{b} \in \mathbb{R}^n$
Question Existe-t-il $\mathbf{x} \in \mathbb{R}^n$ tel que $\mathbf{Ax} = \mathbf{b}$?

Considérons un autre problème de logique booléenne appelé problème de satisfaisabilité. On se donne un ensemble $\mathcal{X} = \{x_1, x_2, \cdots, x_n\}$ de variables booléennes. Un littéral est une variable booléenne x ou son complément $\overline{x}$. Une affectation de vérité est une application $a : \mathcal{X} \to \{v, f\}$ qui associe à chaque variable booléenne la valeur «vrai» ou «faux». Si $a(x) = v$ le littéral x est vrai et si $a(x) = f$ alors x est faux. Le littéral $\overline{x}$ est vrai si et seulement si x est faux. Une clause sur $\mathcal{X}$ est une disjonction de littéraux. Une clause est vraie si et seulement si au moins un des littéraux de la clause est vrai. Une proposition est un ensemble de clauses $\mathcal{C} = \{C_1, \cdots, C_m\}$ (qu'on écrit aussi $\mathcal{C} = C_1 \wedge \cdots \wedge C_m$) sur $\mathcal{X}$ en conjonction. La proposition $\mathcal{C}$ est satisfaisable si et seulement si il existe une affectation de vérité a pour les variables de $\mathcal{X}$ qui satisfait chacune des m clauses de $\mathcal{C}$. S'il n'existe aucune affectation de vérité satisfaisant $\mathcal{C}$ on dit que $\mathcal{C}$ est non satisfaisable ou contradictoire.

Satisfaisabilité (**SAT**)
Instance Entier $n \geq 2$, $\mathcal{X} = \{x_1, \cdots, x_n\}$ et $\mathcal{C} = \{C_1, \cdots, C_m\}$
Question $\mathcal{C}$ est-t-elle satisfaisable ?

En d'autres termes existe-t-il une affectation de vérité aux variables booléennes de $\mathcal{X}$ qui satisfasse la proposition $\mathcal{C}$? Par exemple, si $\mathcal{X} = \{x_1, x_2\}, C_1 = x_1 \vee \overline{x}_2, C_2 = \overline{x}_1 \vee x_2$ (2 clauses et 2 littéraux par clauses) et $\mathcal{C} = C_1 \wedge C_2$ en posant $a(x_1) = a(x_2) = v$ (a est l'affectation de vérité) on voit que $\mathcal{C}$ est satisfaisable.

Etant donné un problème, généralement on peut lui associer un problème de décision. La théorie formelle de la complexité se restreint aux problèmes de décision pour au moins trois raisons :

1. la machine de Turing n'a que deux états d'arrêt « oui » et « non ». Elle ne peut donc résoudre qu'un problème de décision ;
2. on n'a pas à s'inquiéter de la taille digitale (qui est 1) de la réponse (puisque c'est « oui » et « non ») ;
3. si l'on sait résoudre polynomialement un problème alors on peut faire de même pour le problème de décision associé. Si l'on peut fournir la preuve que le problème de décision associé est difficile alors le problème l'est aussi.

Tout problème de décision P admet un complémentairenoté coP. C'est le problème de décision ayant même instance générique mais la question est « inversée ». La réponse au complémentaire est « oui » si et seulement si la réponse au problème en question est « non ».

A titre d'exemple, le complémentaire du problème de décision **SEL** est

---

co**SEL**

Instance Entier $n \geq 2, \mathbf{A} \in \mathbb{R}^{n \times n}$ et $\mathbf{b} \in \mathbb{R}^n$

Question N'existe-t-il aucun $\mathbf{x} \in \mathbb{R}^n$ tel que $\mathbf{Ax} = \mathbf{b}$?

---

et le complémentaire de **SAT** est

---

co**SAT**

Instance Entier $n \geq 2$, $\mathcal{X} = \{x_1, \cdots, x_n\}$ et $\mathcal{C} = \{C_1, \cdots, C_m\}$

Question $\mathcal{C}$ est-t-elle une contradiction ?

3.3.4 Classe $\mathbb{P}$

La complexité d'un problème de décision P n'est pas la complexité $t_{\mathrm{A}}(n)$ d'un algorithme A particulier qui résout P. Il est vrai que $t_{\mathrm{A}}(n)$ constitue une borne supérieure de la complexité du problème. Ce qui est surtout intéressant c'est de connaître une borne inférieure $t\,(\mathrm{P})$ de la complexité du problème car cela signifiera que tout algorithme A qui résout P aura une complexité supérieure à $t\,(\mathrm{P})$. Formalisons cela.

Definition 3.8. La complexité d'un problèmede décision **P** est le nombre

$$t\,(\mathbf{P}) = \min_{\{\mathrm{A} | \mathrm{A} \text{ résout } \mathbf{P}\}} t_{\mathrm{A}}(n)$$

Un problème de décision **P** est dit facile (ou de complexité polynomiale) si $t(\mathbf{P})$ est une fonction polynomiale (autrement dit il existe un algorithme polynomial pour le résoudre). On note $\mathbb{P}$ l'ensemble de tous les problèmes faciles.

Il est possible qu'un problème donné puisse être résolu par plusieurs algorithmes polynomiaux de diverses complexités. Mais ceci ne nous intéresse pas. Ce qui nous intéresse ici est seulement la distinction entre les algorithmes polynomiaux et les algorithmes exponentiels. On sait, par exemple, que l'algorithme d'élimination de Gauss résout **SEL** en $O(n^3)$. Par conséquent **SEL** $\in \mathbb{P}$. Néanmoins, on ne sait pas si $O(n^3)$ constitue une borne inférieure de la complexité de **SEL** car il est possible qu'il y ait un algorithme de plus faible complexité.

Si P $\in \mathbb{P}$ alors il existe un algorithme polynomial A pour résoudre P. En résolvant P à l'aide de A on résout simultanément coP car la réponse à coP est «oui» si et seulement si la réponse à P est «non». D'où la propriété suivante : la classe co$\mathbb{P}$est l'ensemble des complémentaires des problèmes de $\mathbb{P}$.

Proposition 3.9. *[24] Si* $\boldsymbol{P} \in \mathbb{P}$ *alors co*$\boldsymbol{P} \in \mathbb{P}$ *(et donc co*$\mathbb{P} \subset \mathbb{P}$*).*

Cette propriété a une conséquence importante (il suffit de voir que co-coP = P).

Proposition 3.10. *[24]* co$\mathbb{P} = \mathbb{P}$.

Les problèmes faciles se confondent donc dans ce sens avec leurs complémentaires. Essayons de reclasser les problèmes à la lumière de ce qui vient d'être présenté (voir figure 3.3).

- Les problèmes du groupe 1 ne peuvent être résolus par algorithme.
- Ceux du groupe 2 ne peuvent être résolus que par des algorithmes exponentiels.
- Ceux du groupe 4 sont résolus (ainsi que leurs complémentaires) par des algorithmes efficaces.
- S'il est clair que les groupes 2 et 4 sont disjoints, la frontière entre les groupes 2 et 3, ainsi que la frontière entre les groupes 3 et 4, est floue. En tout cas, il y a des problèmes du groupe 3 (SAT en fait partie) qui ne semblent être ni dans le groupe 2, ni dans le groupe 4. On va les étudier dans la suite de cette section (bien qu'il soit possible que le groupe 3 soit vide).

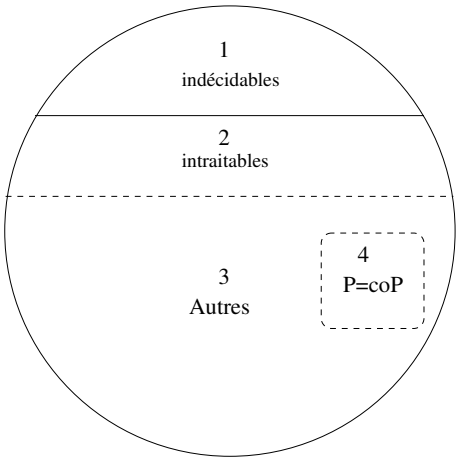

FIG. 3.3: Seconde tentative de classification

3.3.5 Classe $\mathbb{NP}$

Etant donné un problème de décision P, la première chose à savoir est « est-ce que P $\in \mathbb{P}$? ». Pour certains problèmes, plutôt peu nombreux (groupe 4 de la figure 3.3), la réponse est « oui ». Par exemple, SEL $\in \mathbb{P}$ puisque l'on sait que l'algorithme de Gauss le résoud en $O(n^3)$ où n est le nombre d'inconnues. Mais pour la plupart des problèmes connus du groupe 3 de la figure 3.3 on ne sait pas répondre à la question posée. Par exemple, on ne sait si SAT $\in \mathbb{P}$ ou non. Dans l'état de nos connaissances actuelles on ne connaît pas d'algorithme polynomial pour résoudre SAT et donc on n'a pas de preuve qu'il appartient à $\mathbb{P}$. On n'a pas non plus de preuve que SAT $\notin \mathbb{P}$ (c'est-à-dire qu'il est intraitable comme les problèmes du groupe 2 de la figure 3.3). En ce sens SAT semble « plus facile » que les problèmes intraitables. Cette propriété de SAT que l'on vient d'évoquer est partagée par de nombreux problèmes constituant une classe appelée $\mathbb{NP}$.

Introduisons le concept de certificat. Etant donnée une instance à réponse « oui » d'un problème de décision quelconque, un certificat pour cette instance est un objet mathématique (nombre, vecteur, liste, ensemble, chemin, etc.) destiné à valider la réponse « oui ». Un certificat est dit succinctsi sa taille digitale est majorée par un polynôme en la taille digitale de l'instance (on dit plus simplement qu'il est de petite taille). De même, on dira de la validation qu'elle est facile si elle prend un temps majoré polynomialement en la taille digitale de l'instance.

Definition 3.11. On dit qu'un problème de décision **P** a la propriété du certificat succinct si à toute instance à réponse «oui» on peut (i) associer un certificat succinct et

(ii) valider facilement la réponse.

Remarquons que dans cette définition on ne cherche pas à savoir comment le certificat a été obtenu et on ne compte pas son temps d'obtention.

Example 3.1. *Le problème* ***SEL*** *a la propriété du certificat succinct. Un certificat pour* ***SEL*** *est un vecteur* $\mathbf{x} \in \mathbb{R}^n$ *solution du système* $\mathbf{Ax} = \mathbf{b}$. *Il n'est pas du tout évident qu'il soit succinct mais s'il l'est, la validation de la réponse «oui» (c'est-à-dire la vérification que* $\mathbf{x}$ *constitue bien une solution) est facile car il s'agit de calculer* $\mathbf{Ax}$ *(en* $\mathrm{O}(n^2)$*) et comparer* $\mathbf{Ax}$ *et* $\mathbf{b}$.

Arrêtons-nous un instant à cette notion de taille digitale du certificat qui doit être majorée polynomialement en la taille digitale de l'instance du problème en question car c'est un point crucial. Un certificat pour SEL est un vecteur $\mathbf{x} \in \mathbb{R}^n$ supposé être une solution du système $\mathbf{Ax} = \mathbf{b}$. Ce vecteur a n composantes (nombre dont la taille digitale est évidemment majorée polynomialement en la taille digitale de l'instance de SEL). Mais une composante quelconque du vecteur $\mathbf{x}$ a-t-elle une taille digitale majorée polynomialement en la taille de l'instance de SEL ? C'est cette question précisément qui n'a pas une réponse évidente car les composantes de la solution $\mathbf{x}$ sont obtenus à la faveur d'un procédé algorithmique (algorithme d'élimination de Gauss ou autre) qui pourrait éventuellement mettre en jeu des nombres de taille arbitrairement grande. Qu'en est-t-il du certificat pour SEL ? Plus précisément est-ce que toute composante du certificat $\mathbf{x}$ a une petite taille digitale (c'est-à-dire une taille digitale majorée par un polynôme en la taille de l'instance « $n, \mathbf{A}, \mathbf{b}$ » de SEL). En d'autres termes le certificat $\mathbf{x}$ est-il succinct ? La réponse est oui (voir [170]).

Insistons bien sur la notion de certificat par une explication imagée. Imaginons un enseignant (on utilise le terme superviseur dans le jargon de la théorie de la complexité) qui propose, comme devoir à domicile, de résoudre le problème SEL sur une certaine instance. On suppose, puisque SEL est un problème de décision, que l'enseignant se contente d'une réponse « oui » ou « non ». Un étudiant se présente au bureau de l'enseignant et lui dit : « j'ai résolu le problème. La réponse est oui ». L'enseignant répond : « je ne cherche pas à savoir comment vous avez fait mais je veux être convaincu ». L'étudiant lui présente alors une feuille de papier sur laquelle il a inscrit les composantes du vecteur $\mathbf{x}$ supposé résoudre SEL. Cette inscription ne doit pas tenir trop de place. L'espace utilisé

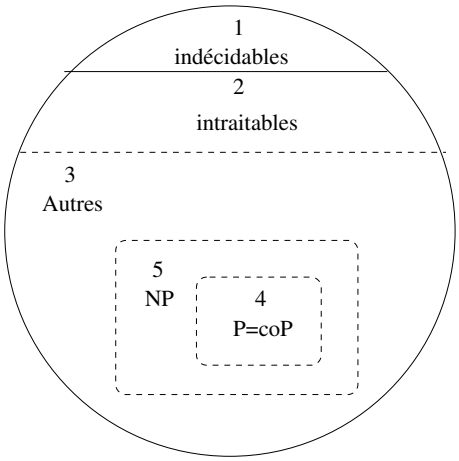

FIG. 3.4: Introduction de la classe $\mathbb{NP}$

pour l'inscription du certificat doit être majoré par un polynôme en la taille digitale de l'instance. L'enseignant calcule $\mathbf{Ax}$ puis il compare $\mathbf{Ax}$ et $\mathbf{b}$ et trouve l'égalité. Cette validation ne doit pas prendre trop de temps. Le temps utilisé pour la validation doit être majoré par un polynôme en la taille digitale de l'instance. A ces conditions l'enseignant est convaincu.

Ce que l'on doit retenir est que :

- il importe peu comment est obtenu le certificat (il peut être éventuellement obtenu par un procédé quelconque ou même par tâtonnement) et peu importe son temps d'obtention (il peut même être obtenu par un algorithme exponentiel) ;
- la taille du certificat est bornée par un polynôme en la taille digitale n de l'instance ;
- la vérification du certificat se fait en temps polynomial en n.

La propriété du certificat succinct (appelée principe du superviseur dans certains ouvrages) a été introduite par Edmonds [177].

Definition 3.12. La classe $\mathbb{NP}$ est celle des problèmes ayant la propriété du certificat succinct.

Remark 3.13. Les problèmes de la classe $\mathbb{P}$ ont été défini de manière naturelle comme étant ceux qui peuvent être résolus par algorithme polynomial. Mais la définition de la classe $\mathbb{NP}$ semble curieuse au premier abord car elle invoque un principe (du certificat succinct) qui semble futile, mais dont la consistance apparaîtra plus loin.

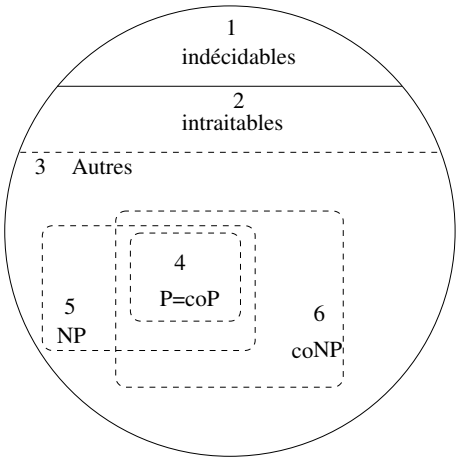

FIG. 3.5: Introduction de la classe coNℙ

3.3.6 Classe coNℙ et bonne caractérisation

La classe coNℙest définie naturellement comme l'ensemble des complémentaires des problèmes de Nℙ. Une question intéressante est la suivante : « Etant donné un problème P $\in$ Nℙ, est-ce que son complémentaire coP $\in$ Nℙ ? ». Question équivalente à « Est-ce que P $\in$ Nℙ$\cap$ coNℙ ? ». Un problème P de la classe Nℙ$\cap$ coNℙ est donc celui pour lesquel le complémentaire coP a la propriété du certificat succinct. C'est Edmonds [177] qui a introduit cette classe Nℙ$\cap$ coNℙ et a appelé les problèmes de cette classe problèmes bien caractérisés. Cette classe n'est pas vide puisqu'il est clair que tout problème de la classe ℙ est bien caractérisé. Cette classe a une grande importance en optimisation (dualité).

Est-ce que Nℙ = coNℙ ? On n'a pas de réponse à cette question. Il y a des problèmes dans Nℙ dont on ne sait pas si le complémentaire est dans Nℙ, puisque l'on ne sait pas produire de certificat succinct. C'est précisément cette asymétrie qui donne de la consistance à la définition de la classe Nℙ en termes de certificat succinct. L'exemple de coSAT est édifiant. Si l'on sait que SAT est dans Nℙ (puisqu'un certificat succinct est formé par une certaine affectation de vérité aux variables booléennes et il est facile de valider la satisfaction de la proposition $\mathcal{C}$) on ne sait si coSAT est dans Nℙ. Comment produire un certificat succinct ? Il n'y a qu'une seule façon connue de convaincre le superviseur qu'une proposition donnée est une contradiction, c'est de lui dresser une table de vérité. Un certificat loin d'être succinct puisque sa taille est en $\mathrm{O}\left(2^n\right)$, où n est le nombre de variables booléennes. Pourtant le fait que l'on ne sache pas produire de certificat succinct pour coSAT ne signifie pas qu'il n'appartient pas à Nℙ.

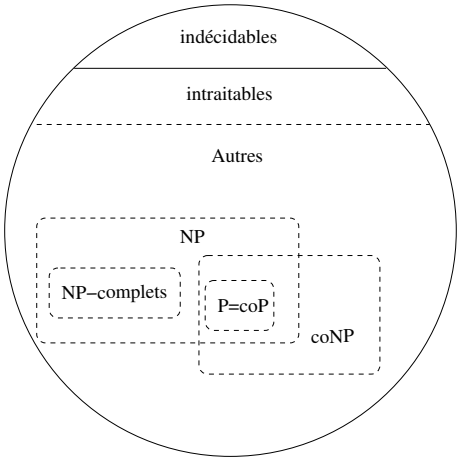

FIG. 3.6: Introduction de la classe des problèmes $\mathbb{NP}$-complets

3.3.7 Classe des problèmes $\mathbb{NP}$-complets

La plupart des problèmes de décision de la classe $\mathbb{NP}$ ont résisté à toute tentative de résolution par algorithme polynomial (on a l'exemple de SAT) malgré les recherches ininterrompues. Ces problèmes constituent une classe spéciale dans $\mathbb{NP}$. Les fondements théoriques ayant abouti à la définition de cette classe, appelée classe des problèmes de décision $\mathbb{NP}$-complets, est due à Cook [178]. C'est lui qui prouva l'existence du premier problème NP-complet (SAT). Pour cela, Cook montre que tout calcul de la machine de Turing non déterministe peut être décrit par une proposition de telle sorte que lorsque la machine traite un problème de la classe $\mathbb{NP}$ alors la réponse est « oui » si et seulement si la proposition est satisfaisable. De cette preuve il s'en suit que si l'on a un algorithme efficace pour SAT alors on aura un algorithme efficace pour tout problème appartenant à $\mathbb{NP}$, et s'il existe un problème intraitable dans $\mathbb{NP}$ alors SAT est intraitable.

Une question cruciale est d'établir une relation entre $\mathbb{P}$ et $\mathbb{NP}$. De la définition 3.12 il s'en suit que $\mathbb{P} = \text{co}\mathbb{P} \subset \mathbb{NP}$ car si $\text{P} \in \mathbb{P}$ alors on est en mesure d'exhiber un certificat succinct pour coP à l'aide de l'algorithme polynomial. Par exemple, un certificat succinct pour coSEL est un vecteur $\mathbf{y} \in \mathbb{R}^n$ et le superviseur sera convaincu en vérifiant en temps polynomial que $\mathbf{y}^t\mathbf{A} = \mathbf{0}$ et $\mathbf{y}^t\mathbf{b} \neq 0$. Là également il n'est pas évident que le certificat soit succinct. On trouvera les arguments dans [170]. La partie qui demeure sans réponse au stade de nos connaissances actuelles est « est-ce que $\mathbb{NP} \subset \mathbb{P}$? ». C'est une des questions centrales posées aujourd'hui en mathématiques et en informatique. Le « Clay Mathematic Institute » de Cambridge offre un prix d'un million de dollars pour la résolution de cette question. Si la réponse est oui, alors la frontière entre les groupes

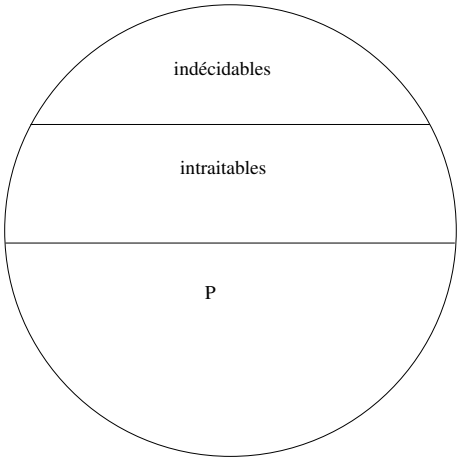

FIG. 3.7: Hiérarchie résultant de la conjecture $\mathbb{P} = \mathbb{NP}$

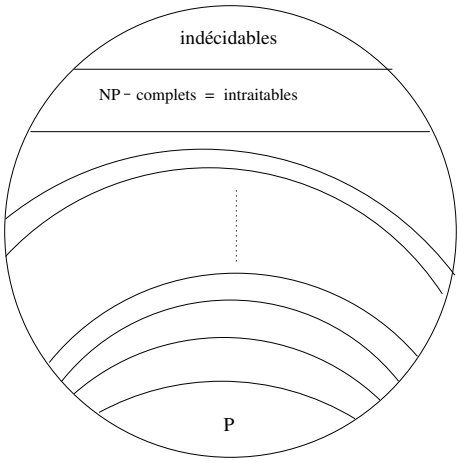

FIG. 3.8: Hiérarchie résultant de la conjecture $\mathbb{P} \neq \mathbb{NP}$

2 et 3 évoquée au début de cette section se précise, et toutes les classes introduites se confondent en une seule : $\mathbb{P}$ (voir [179]). Il n'y aura plus jamais que les problèmes indécidables, les problèmes intraitables et les problèmes faciles (figure 3.7).

Inversement, si la réponse est non, alors il existe [180], [181] une hiérarchie infinie de classes de problèmes, les uns plus difficiles que les autres, les problèmes de $\mathbb{P}$ étant les plus faciles, et les problèmes $\mathbb{NP}$-complets les plus difficiles (figure 3.8).

Chapitre 4

Une heuristique lagrangienne pour le PRE

4.1 Introduction

On se donne n nombres positifs $c_1, \cdots, c_n$ et n sous-ensembles $I_1, \cdots, I_n$ de l'ensemble $M = \{1, \cdots, m\}$ tels que $\cup_{j=1}^{n} I_j = M$. Chaque coût c_j est attaché au sous-ensemble I_j. Les éléments de M sont appelés items. Soit $N = \{1, \cdots, n\}$. Un recouvrement est toute partie $C \subset N$ telle que $\cup_{j \in C} I_j = M$. Le coût du recouvrement C est $\sum_{j \in C} c_j$. Le PRE est le problème de chercher un recouvrement à moindre coût. Soient $J_i = \{j \in N | I_j \ni i\}$ l'ensemble des sous-ensembles contenant (ou recouvrant) l'item i. Une formulation du PRE en programme binaire est la suivante

$$\begin{array}{ll} \min \sum_{j \in N} c_j x_j & \\ \sum_{j \in J_i} x_j \geq 1 & i \in M \\ x_j \in \{0,1\} & j \in N \end{array} \tag{4.1}$$

où x est le vecteur indicateur du recouvrement, i.e.

$$x_j = \begin{cases} 1 & \text{si le sous-ensemble } I_j \text{ est dans le recouvrement} \\ 0 & \text{sinon} \end{cases}$$

On va utiliser indistinctement l'ensemble C et le vecteur x pour référer au recouvrement. Un sous-ensemble $j \in C$ est redondant si $C \setminus \{j\}$ est encore un recouvrement. Un recouvrement qui contient un sous-ensemble redondant est dit redondant. Un recouvrement est dit premier s'il n'est pas redondant.

Le PRE est fortement $\mathbb{NP}$-dur [24], ce qui signifie qu'on ne peut espérer un algorithme pseudo-polynomial à moins que $\mathbb{NP} = \mathbb{P}$. De plus, il ne peut y avoir de schéma d'approximation polynomial à moins que $\mathbb{NP} = \mathbb{P}$, et la meilleure garantie possible d'approximation est $\Theta(\log m)$ [29]. En dépit de ces faits décourageants, le PRE est très étudié à cause de ses nombreuses applications pratiques dans plusieurs domaines tels que l'habillage [84, 85], la localisation des services d'urgence [45, 47, 48, 182], le découpage politique [51], la simplification d'expressions booléennes [183], la production d'acier [56, 184, 185], l'affectation de trafic dans les systèmes de communication par satellite [186], l'attaque et la défense d'un réseau [187], le transport de pétrole brut [188], l'ordonnancement [189], la sélection optimale de portefeuilles en finance [62], la localisation de stops [18].

La plupart des algorithmes et des heuristiques sont consignés dans les surveys [64, 65]. D'une manière surprenante, Caprara et al y rapportèrent que le logiciel de commerce Cplex est supérieur à toutes les méthodes exactes basées sur le recherche arborescente [71–73, 76]. Quelques résultats théoriques existent à propos de la structure faciale du polytope qui est l'enveloppe convexe des recouvrements [105, 190–192].

Puisque les méthodes exactes sont applicables seulement à des instances de petite taille, pour traiter les grandes instances survenant en pratique il faut utiliser des heuristiques capables de déterminer des solutions quasi optimales en un temps raisonnable. Il y en a en gros deux classes, les heuristiques lagrangiennes qui essaient d'exploiter la solution du problème relaxé durant la méthode de sous-gradients appliquée au dual lagrangien du PRE [83–86] et les heuristiques inspirées de la nature ou méta-heuristiques [88, 90, 92, 96, 101, 193]. Bien qu'efficaces dans l'identification de très bonnes solutions, ces dernières souffrent de la non reproductibilité des résultats, à cause de leur nature aléatoire inhérente, et de la grande difficulté d'analyse de l'algorithme (complexité et analyse au pire cas).

La section suivante rappelle quelques faits basiques à propos de la relaxation lagrangienne et de la méthode de sous-gradients. Ensuite, on exploite une idée simple et utile qui consiste à utiliser l'information fournie par la méthode de sous-gradients pour

éliminer la plupart des variables. En pratique, cette procédure d'élimination résulte en l'élimination du problème de toutes les variables qui n'aparaissent pas dans la solution optimale. Cela attirera certainement l'attention des praticiens. En effet, si le but est de trouver une solution approximative du PRE, alors au lieu d'avoir affaire à l'instance originale, on peut restreindre notre attention au problème réduit qui est une petite partie de l'instance originale. Dans la section qui suit, on propose une heuristique de construction basée sur l'idée de regret, ensuite on propose une heuristique d'amélioration fondée sur le plongement d'un recouvrement dans un recouvrement plus large, et donc redondant, pour en extraire le meilleur recouvrement possible. Les deux procédures sont intégrées dans une heuristique lagrangienne. La cinquième section concerne l'expérimentation numérique et la comparaison avec des méthodes connues sur des données benchmark. La sixème section constitue une conclusion.

4.2 Quelques faits basiques à propos de la relaxation lagrangienne et de la méthode de sous-gradients

On rappelle quelques faits (voir la section 3.2 et [101, 194] pour plus de détails). Un vecteur $\pi \in \mathbb{R}_+^m$ de multiplicateurs de Lagrange est associé aux m constraintes (4.1). Le problème relaxé est

$$\mathrm{LR}(\pi) \begin{cases} \min z(\pi) = \sum_{j \in N} \left(c_j - \sum_{i \in I_j} \pi_i \right) x_j + \sum_{i \in M} \pi_i \\ x_j \in \{0,1\}, j \in N \end{cases}$$

Il est bien connu que, pour π fixé, $z(\pi)$ constitue une borne inférieure de la valeur optimale du PRE. Habituellement, au lieu de calculer la meilleure borne inférieure qui est la valeur optimale $z(\pi^*)$ du dual lagrangien

$$\max_{\pi \in \mathbb{R}_+^m} z(\pi)$$

on se contente de calculer une valeur approchée z_{LB} au moyen d'une méthode itérative appelée méthode de sous-gradients. Celle-ci génère une séquence de vecteurs $\pi^{(0)}, \pi^{(1)}, \cdots$ et

$$z_{LB} = \max_{k \geq 0} z\left(\pi^{(k)}\right)$$

A chaque itération r, pour $\pi^{(r)}$ fixé, le problème relaxé $\mathrm{LR}(\pi^{(r)})$ est facilement résolu en posant pour $j \in N$

$$\begin{cases} x_j^{(r)} = 1 & \text{si } c_j - \sum_{i \in I_j} \pi_i^{(r)} < 0 \\ x_j^{(r)} = 0 & \text{si } c_j - \sum_{i \in I_j} \pi_i^{(r)} > 0 \\ x_j^{(r)} = 0 \text{ ou } 1 & \text{si } c_j - \sum_{i \in I_j} \pi_i^{(r)} = 0 \end{cases} \tag{4.2}$$

Habituellement, un nombre maximum d'itérations est fixé, et la méthode prend un temps $O(m \times n \times d)$ où

$$d = \sum_{j \in N} |I_j|$$

L'algorithme basique est appliqué à la section suivante, et on en donne un pseudo-code.

4.3 Une heuristique simple pour réduire le nombre de sous-ensembles

Supposons que K est un sous-ensemble propre de N avec $K = \{j_1, \cdots, j_p\}$. Considérons le problème

$$(\text{coreSCP}) \begin{cases} \min \sum_{k \in K} c_k x_k & \\ \sum_{k \in J_i \cap K} x_k \geq 1 & i \in M \\ x_k \in \{0, 1\} & k \in K \end{cases}$$

Le problème coreSCP est un PRE avec le même nombre d'items que le PRE original mais avec moins de sous-ensembles (ou variables). Tandis que le PRE original a toujours une solution optimale, il n'y a aucune garantie que coreSCP ait une solution réalisable. Cependant, quand coreSCP a une solution réalisable, celle-ci est automatiquement réalisable pour le PRE original. De plus, le coût d'un recouvrement optimal de coreSCP constitue une borne supérieure du coût optimal du PRE original.

Le problème coreSCP, que l'on se propose de construire en spécifiant l'ensemble K, satisfait deux bonnes propriétés :

(i) Il a beaucoup moins de variables que le PRE original ($p \ll n$), et

(ii) il contient potentiellement le recouvrement optimal (ou le meilleur connu) du PRE original.

L'ensemble K est facilement construit. On applique une méthode de sous-gradients au PRE original. Quand elle s'arrête, on calcule pour chaque indice $j \in N$ la valeur

$$f_j = \frac{1}{T} \sum_{r=0}^{T} x_j^{(r)}$$

où $x_j^{(r)}, j \in N$, est calculée à chaque itération r par les formules en (4.2). Clairement on a $0 \leq f_j \leq 1, j \in N$. La valeur f_j la fréquence d'implication de la variable x_j dans la solution du problème relaxé. Elle constitue en quelque sorte un score du sous-ensemble j qui indique que plus grande est la valeur f_j, plus grand est notre désir de mettre le sous-ensemble j dans l'ensemble K de candidats potentiels au recouvrement optimal. Il est surprenant de voir que la valeur f_j est nulle pour la plupart des sous-ensembles, ce qui indique que la variable x_j n'est jamais sélectionnée par l'algorithme de sous-gradients parceque sont coût réduit

$$c_j - \sum_{i \in I_j} \pi_i^{(r)}$$

est positif à chaque itération r.

Comme conséquence de cette observation, notre idée est d'inclure dans le problème coreSCP uniquement les sous-ensembles j correspondant aux valeurs non nulles f_j (voir la figure 4.1). Soit p leur nombre. On affirme que ces sous-ensembles « ont le plus de chance » d'appartenir à un recouvrement optimal du PRE d'origine. On n'a aucune preuve formelle de cette affirmation. Cependant, l'expérimentation numérique confirme l'affirmation en montrant que le problème coreSCP défini de cette manière contient toujours le recouvrement optimal (ou le meilleur connu) du PRE original.

L'idée d'éliminer des variables du problème posé n'est pas nouvelle, et des idées similaires sont connues. Caprara et al [84] utilisent la borne lagrangienne et les coûts réduits dans ce but. Ceria et al [85] éliminent les sous-ensembles dont le coût réduit correspondant

est en dessous d'un certain seuil. En ce qui nous concerne, l'approche est différente. On élimine toute variable qui n'est jamais sélectionnée par l'algorithme de sous-gradients. Bien que l'idée sous-jacente soit simple, elle est appliquée pour la première fois, et résulte en un PRE beaucoup plus petit (coreSCP) qui contient le recouvrement optimal du PRE original comme on le verra plus loin lors de l'expérimentation numérique. Les détails sont donnés dans un pseudo-code.

Puisque notre but, dans ce chapitre, est d'approximer le PRE, on ne s'occupera plus de ce dernier pour se concentrer sur le plus petit problème coreSCP.

Construction de coreSCP
Input $m, n, c_j, I_j, j \in N$
Output P // pour construire coreSCP
Initialisation
- Calculer $J_i, i \in M$
- En utilisant la méthode gloutonne, construire une borne supérieure z_{UB} de la valeur du recouvrement optimal
- $\pi_i^{(0)} \longleftarrow \min_{j \in J_i} c_j / |I_j|$ pour $i \in M$
- $z_{LB} \longleftarrow -\infty$
- $\rho \longleftarrow 2$ // Le coefficient de relaxation est divisé par deux après un certain nombre fixé d'itérations
- $f_j \longleftarrow 0$ pour $j \in N$

Boucle
A chaque itération $r = 0, 1, \cdots, T$
- Résoudre le problème LR($\pi^{(r)}$)
- $f_j \longleftarrow f_j + x_j^{(r)}$ pour $j \in N$
- Si $z_{LB} < z\left(\pi^{(r)}\right)$ alors $z_{LB} \longleftarrow z\left(\pi^{(r)}\right)$
- // Calculer le sous-gradient
  $\sigma_i^{(r)} \longleftarrow 1 - \sum_{j \in J_i} x_j^{(r)}$ pour $i \in M$
- // Calculer les nouveaux multiplicateurs de Lagrange
  $\pi_i^{(r+1)} \longleftarrow \max\left\{0, \pi_i^{(r)} + \rho \frac{z_{UB} - z\left(\pi^{(r)}\right)}{\left\|\sigma_i^{(r)}\right\|^2} \sigma_i^{(r)}\right\}$ pour $i \in M$

$f_j \longleftarrow f_j / T$ pour $j \in N$
$P \longleftarrow \{j \in N | f_j > 0\}$
Output P

4.4 L'heuristique lagrangienne

Rappelons que le problème coreSCP est défini avec m items et p sous-ensembles. Soit $P = \{1, \cdots, p\}$ et redéfinissons pour $i \in M, J_i = \{j \in P | I_j \ni i\}$ comme étant les sous-ensembles de P recouvrant l'item i. Avant de décrire l'heuristique lagrangienne, on commence par présenter ses deux cruciales composantes.

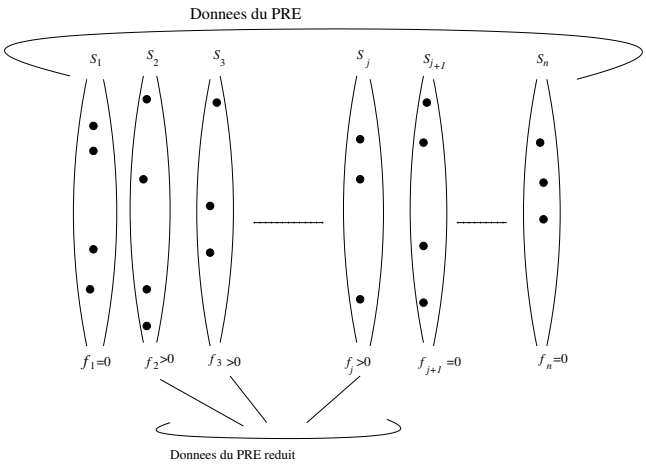

FIG. 4.1: Le problème coreSCP est obtenu en restreignant le PRE original à certains sous-ensembles

4.4.1 Phase de construction - recouvrir avec regret

Notre but est de construire un bon recouvrement du problème coreSCP qui sera soumis plus tard à une phase d'amélioration locale. Comme première tâche on applique l'heuristique gloutonne. Soit $CoverSize$ la cardinalité (nombre de sous-ensembles) du recouvrement glouton. L'heuristique de construction commence avec un recouvrement vide ($x \equiv 0$) dans lequel une petite partie S des p sous-ensembles est insérée. Le paramètre $\%CoverSizeCons$ contrôle le nombre $s = |S|$ de sous-ensembles insérés dans le recouvrement vide avec $s = \%CoverSizeCons \times CoverSize$. Définissons $C1 = P \setminus S$ comme l'ensemble des sous-ensembles qui ne font pas partie du recouvrement partiel, et soit $R1$ l'ensemble des items non recouverts par les sous-ensembles de S. Notre seconde tâche consiste à résoudre (ou approximer) le problème

$$(\text{scp1}) \begin{cases} \min \sum_{j \in C1} c_j x_j & \\ \sum_{j \in J_i \cap C1} x_j \geq 1 & i \in R1 \\ x_j \in \{0,1\} & j \in C1 \end{cases}$$

Soit E le recouvrement obtenu. Alors $C = S \cup E$ est un recouvrement, peut-être redondant, de coreSCP. Evidemment, plus la valeur du paramètre $\%CoverSizeCons$ est petite, plus grande est la taille du problème scp1. L'originalité de la construction réside dans le choix de l'ensemble S. En effet, les sous-ensembles de S avec lesquels on commence la construction du recouvrement seront insérés avec le principe du regret. Plus tard, lors de l'expérimentation numérique, on compare la phase de construction dans les

deux cas, avec et sans regret, et on montre que le principe de regret donne de meilleurs résultats. Il n'est nul besoin d'enlever les sous-ensembles redondants du recouvrement obtenu lors de la phase de construction puisque, de toute façon, il sera soumis à une phase d'amélioration qui se chargera implicitement de cette tâche. L'expérimentation numérique montre que l'heuristique de construction, à elle seule, est capable d'identifier de bons recouvrements. Expliquons maintenant le concept de regret dans l'heuristique gloutonne.

Les heuristiques gloutonnes sont des algorithmes qui construisent une solution réalisable, pas à pas, en prenant des décisions irréversibles. C'est là où réside leur intérêt (rapidité) aussi bien que leur désavantage (les bonnes décisions prises au début ont des conséquences déplorables sur les décisions à prendre à la fin). L'idée d'incorporer le regret dans les heuristiques gloutonnes n'est pas nouvelle, et résulte généralement en de meilleurs résultats (voir [195, 196] pour des applications concernant les problèmes du voyageur de commerce et de l'affectation généralisée).

L'heuristique gloutonne [27] pour le PRE considère les sous-ensembles avec la question « quel sous-ensemble doit être pris en premier dans le recouvrement ? ». La question est résolue de la manière suivante. En commençant avec les sous-ensembles originaux $I_j, j \in P$ et un recouvrement vide, 1) calculer un score pour chaque sous-ensemble, 2) prendre le sous-ensemble ayant le plus petit score dans le recouvrement, 3) supprimer les items recouvets, et répéter les trois étapes avec les sous-ensembles modifiés jusqu'à ce que tous les items soient recouverts.

Dans notre version (avec regret), au lieu de considérer les sous-ensembles, on propose de considérer les items avec la question « quel item doit être recouvert en premier ? » Pour traiter cette question, on calcule une valeur pour chaque item, appelée regret, à partir des scores des sous-ensembles. Pour calculer le regret de l'item i on considère tous les sous-ensembles le recouvrant. Le regret est la différence entre le plus petit et le second plus petit score des sous-ensembles. Donc, l'item avec le plus grand regret sera recouvert en premier par le sous-ensemble ayant le plus petit score. Les détails sont donnés dans un pseudo-code.

On verra plus tard, l'évidente supériorité de l'heuristique gloutonne avec regret.

Heuristique gloutonne avec regret
Input Données du problème coreSCP
Output Recouvrement x et son coût

Pas 0 $x_j \longleftarrow 0$ pour $j \in P$
$cost \longleftarrow 0$

Pas 1 // Calculer le score σ_j de chaque sous-ensemble j
Si $I_j = \varnothing$ alors $\sigma_j = \infty$ sinon $\sigma_j = c_j / |I_j|$, pour $j \in P$

Pas 2 // Calculer le regret ρ_i de chaque item i
pour $i \in M$ (i non recouvert)
$a \longleftarrow \sigma_{j^*} = \min_{j \in J_i} \sigma_j$
$b \longleftarrow \min_{j \in J_i, j \neq j^*} \sigma_j$
$\rho_i \longleftarrow b - a$
$store_i \longleftarrow j^*$ // Se souvenir du sous-ensemble qui doit recouvrir l'item i

Pas 3 // Soit i^* l'item avec le plus grand regret
$j^* \longleftarrow store_{i^*}$
$x_{j^*} \longleftarrow 1$
$cost \longleftarrow cost + c_{j^*}$
$I_j \longleftarrow I_j \diagdown I_{j^*}$ pour $j \in P$

Répéter les pas 1, 2, 3 jusqu'à ce que $I_j = \varnothing, j \in P$
Output x et $cost$

4.4.2 Phase d'amélioration

La phase d'amélioration prend le recouvrement C obtenu durant la phase de construction et calcule un recouvrement plus large C' en ajoutant à C un certain nombre de sous-ensembles de $P \setminus C$. Le recouvrement résultant C' est donc redondant. Ensuite on en extrait le meilleur recouvrement en résolvant un petit PRE (comme il a été fait dans [86]). Le paramètre $\%CoverSizeImp$ contrôle le nombre $t = \%CoverSizeImp \times CoverSize$ de sous-ensembles ajoutés. Bien sûr, plus grande est la valeur de $\%CoverSizeImp$, plus grande est la taille du problème scp2. Les t sous-ensembles de $P \setminus C$ sont sélectionnés avec la plus petite valeur de $c_j \setminus |I_j|$.

Soit $C2 \subset C'$ l'ensemble des sous-ensembles redondants et $R2 \subset M$ l'ensemble des items non recouverts par les sous-ensembles de $C' \setminus R$. Pour extraire le meilleur recouvrement de C' on résout le problème

$$(\text{scp2}) \begin{cases} \min \sum_{j \in C2} c_j x_j & \\ \sum_{j \in J_i \cap C2} x_j \geq 1 & i \in R2 \\ x_j \in \{0,1\} & j \in C2 \end{cases}$$

Soit V le recouvrement optimal trouvé. Alors $C' \setminus R \cup V$ est le meilleur recouvrement extrait de C'. Il est clair que C' est au moins aussi bon que C. Cette heuristique d'amélioration s'est montré efficace comme on le verra plus loin.

4.4.3 L'heuristique lagrangienne

Elle consiste en une méthode de sous-gradients appliquée au dual lagrangien de coreSCP, dans laquelle sont appliquées à chaque itération les heuristiques de construction et d'amélioration. Les détails sont donnés dans un pseudo-code. Notons que l'heuristique gloutonne avec regret est appliquée avec une modification du pas 1 où les coûts sont remplacés par les coûts réduits comme suit :

Pas 1 Pour $j \in P$ si $I_j = \varnothing$ alors si $r_j = c_j - \sum_{i \in I_j} \pi_i^{(r)} > 0$ alors $\sigma_j = r_j / |I_j|$ sinon $\sigma_j = r_j \times |I_j|$

Heuristique lagrangienne
Input Données du problème coreSCP, $\%CoverSizeCons$, $\%CoverSizeImp$
Output Meilleur recouvrement C^* et son coût c^*

Initialisation

- En utilisant l'heuristique gloutonne, trouver un recouvrement C^* et son coût c^* // Soit $CoverSize$ sa cardinalité
- $s \longleftarrow \%CoverSizeCons \times CoverSize$
- $t \longleftarrow \%CoverSizeImp \times CoverSize$
- $\pi_i \longleftarrow 0$ pour $i \in M$
- $\rho \longleftarrow 2$

Boucle Pour chaque itération $r = 0, 1, \cdots, T$

- En utilisant l'heuristique avec regret, insérer s sous-ensembles dans l'ensemble vide C
- Définir et résoudre le problème scp1
- Soit C le recouvrement obtenu
- Mettre à jour le meilleur recouvrement C^* et son coût c^*
- Construire un recouvrement C' en ajoutant t sous-ensembles de $P \backslash C$ à C
- Extraire le meilleur recouvrement C'' de C' en résolvant le problème scp2
- Mettre à jour le meilleur recouvrement C^* et son coût c^*
- Résoudre la problème relaxé
- Calculer le sous-gradient
- Calculer les multiplicateurs de Lagrange
- $\rho \longleftarrow \rho \times 0.95$

Output C^* et c^*

Nom	Densité (%)	Nombre d'items	Nombre de sous-ensembles
A	2	300	3000
B	5	300	3000
C	2	400	4000
D	5	400	4000
E	10	500	5000
F	20	500	5000
G	2	1000	10000
H	5	1000	10000

TAB. 4.1: Caractéristiques des instances

4.5 Expérimentation numérique

Notre méthode est séparée en deux codes. Le premier, appelé « préprocessing », admet en entrée les données du PRE original, exécute la méthode de réduction à la fin de la méthode de sous-gradients, et met comme résultat les données du problème coreSCP dans un fichier. Ce fichier sera lu plus tard par le second code qui exécute l'heuristique lagrangienne durant la seconde méthode de sous-gradients. Les deux codes sont écrits en C et testés sur un HP Compaq (fréquence de 2GHz et RAM de 2 Gb). Les détails d'implémentation seront fournis de manière à permettre au lecteur de reproduire les résultats obtenus.

Le site web de la bibliothèque OR-library (située à Imperial College à Londres et gérée par J. E. Beasley) http ://people.brunel.ac.uk/˜mastjjb/jeb/ fournit les benchmark. Ceux-là sont groupés en huit ensembles avec cinq instances dans chacun. Ces instances sont bien connues et amplement utilisées pour évaluer et comparer les heursitiques. Leurs caractéristiques sont présentées au tableau 4.1.

4.5.1 L'heuristique de réduction de la taille

Durant cette phase, le nombre d'itérations de la méthode de sous-gradients est fixé à 200. Le coefficient de relaxation ρ débute avec une valeur 2 puis est divisé par 2 toutes les 50 itérations (Notons que l'on ne s'intéresse nullement à la borne inférieure fournie). Le résultat de cette phase est la définition du problème coreSCP qui est stocké dans un fichier. Au tableau 4.2, Les deuxième et septième (resp. troisième et huitième) colonnes donnent le nombre de sous-ensembles du problème coreSCP (resp. le temps de calcul de la phase de préprocessing). Il y peut être remarqué que les temps de calcul sont

	Preproces.		Rés. coreSCP avec Cplex			Preproces.		Rés. coreSCP avec Cplex	
	p	Temps	Coût	Temps		p	Temps	Coût	Temps
A1	259	0.08	√	1.16	B1	178	0.08	√	1.25
A2	268	0.07	√	0.91	B2	221	0.08	√	2.79
A3	284	0.07	√	0.93	B3	200	0.10	√	1.34
A4	287	0.05	√	0.10	B4	219	0.11	√	4.71
A5	268	0.08	√	0.18	B5	185	0.09	√	1.27
C1	321	0.11	√	1.23	D1	251	0.15	√	2.59
C2	346	0.11	√	2.16	D2	238	0.12	√	12.13
C3	354	0.08	√	2.48	D3	276	0.14	√	7.85
C4	348	0.09	√	1.44	D4	238	0.13	√	5.71
C5	302	0.10	√	1.20	D5	215	0.12	√	1.29
E1	242	0.40	√	347.67	F1	269	0.85	√	215.61
E2	309	0.41	√	844.77	F2	242	0.82	√	118.37
E3	267	0.43	√	127.31	F3	290	0.83	√	27.24
E4	264	0.40	√	260.77	F4	281	0.84	√	412.16
E5	277	0.39	√	107.67	F5	278	0.83	√	666.93
G1	663	0.34	√	Limit	H1	639	0.86	√	Limit
G2	638	0.35	√	Limit	H2	600	0.87	√	Limit
G3	625	0.36	√	Limit	H3	532	0.86	√	Limit
G4	613	0.36	√	Limit	H4	580	0.85	√	Limit
G5	636	0.34	√	Limit	H5	541	0.82	√	Limit

TAB. 4.2: Détails de la procédure de réduction

insignifiants (moins d'une seconde) et que le problème coreSCP résultant est beaucoup plus petit que le PRE original. Par exemple, tandis que le nombre original de variables de l'instance H1 est 10.000, le problème coreSCP correspondant a seulement 639 variables. Accessoirement, on a vérifié que le problème coreSCP ne contient aucun sous-ensemble dominé (un sous-ensemble I_j est dit dominé par I_k si $I_j \subseteq I_k$ et $c_j \geq c_k$). Pour vérifier que coreSCP contient le meilleur recouvrement connu, il est soumis à Cplex durant un temps limité à 18.000 secondes. Cette étape peut être considérée comme un « certificat », et ne doit pas être considérée comme une méthode d'approximation du PRE, bien que les instances des groupes A, B, C et D, peuvent être considérées comme approximées d'une façon satisfaisante (voir le tableau 4.2 où le symbole √ signifie que le coût du recouvrement trouvé par Cplex est égal au coût du meilleur recouvrement connu. A partir d'ici, on ne s'occupera plus que des ensembles d'instances E, F, G et H, qui sont les plus difficiles, notant que notre heuristique est capable d'identifier toutes les solutions optimales pour les instances des groupes A, B, C et D.

		Gloutonne		Gloutonne avec regret	
	Meilleur	Coût	Taille	Coût	Taille
E1	29	30	30	31	29
E2	30	36	32	33	29
E3	27	31	29	31	28
E4	28	32	32	29	26
E5	28	33	32	30	28
Deviat. moyenne		14.08		8.80	
F1	14	16	16	15	15
F2	15	16	15	16	16
F3	14	17	17	15	14
F4	14	17	17	16	15
F5	13	16	16	15	15
Deviat. moyenne		13.22		10.12	
G1	176	203	130	187	108
G2	154	182	129	163	108
G3	166	192	130	184	110
G4	168	191	127	177	106
G5	168	194	127	181	110
Deviat. moyenne		15.68		7.20	
H1	63	76	67	69	56
H2	63	74	65	68	58
H3	59	65	60	64	58
H4	58	69	62	62	55
H5	55	63	60	60	58
Deviat. moyenne		14.34		8.38	

TAB. 4.3: Comparaison des heuristiques gloutonnes

4.5.2 L'heuristique gloutonne avec regret

Le tableau 4.3 propose la comparaison des deux heuristiques gloutonnes. Les temps de calcul sont insignifiants pour être rapportés. On voit que l'heuristique avec regret obtient de meilleurs recouvrements. Les déviations moyennes par rapport au meilleur recouvrement sont 14.83 et 8.63. De plus, les recouvrements obtenus à l'aide de l'heuristique avec regret sont de plus petite cardinalité car ils sont souvent premiers.

4.5.3 L'heuristique de construction

On a identifié une bonne valeur de $\%CoverSizeCons$ qui est 0.2 ou 20%. Elle est obtenue comme compromis entre la qualité du recouvrement et son temps de calcul. Une valeur plus petite (0.1 par exemple) permet de construire de meilleurs recouvrements mais au

	Construction sans regret			Construction avec regret		
	Taille scp1	Coût	Temps	Taille scp1	Coût	Temps
E1	207x236	29*	2.76	233x236	29*	5.86
E2	223x303	31	11.04	249x303	31	Lim
E3	240x262	28	9.78	270x262	27*	2.91
E4	215x258	29	4.05	255x258	29	9.10
E5	221x271	28*	1.58	243x271	28*	3.38
Deviat. moyenne		2.12			0.66	
F1	212x266	14*	2.20	218x266	14*	2.99
F2	223x239	15*	1.76	238x239	15*	2.27
F3	224x287	15	2.27	225x287	15	4.50
F4	224x278	14*	4.29	226x278	14*	3.50
F5	214x275	14	19.86	214x275	14	19.92
Deviat. moyenne		0.23			0.23	
G1	490x637	180	Lim	507x637	176*	Lim
G2	486x613	155	Lim	529x613	156	Lim
G3	474x599	169	Lim	531x599	170	Lim
G4	502x588	173	Lim	524x588	171	Lim
G5	506x611	170	Lim	547x611	170	Lim
Deviat. moyenne		1.78			1.34	
H1	424x626	66	Lim	469x626	65	Lim
H2	420x587	66	Lim	494x587	65	Lim
H3	450x520	62	Lim	516x520	61	Lim
H4	444x568	59	Lim	517x568	60	Lim
H5	444x529	58	Lim	509x529	57	Lim
Deviat. moyenne		4.36			3.36	

TAB. 4.4: Comparaison des heuristiques de construction avec et sans regret

prix d'une plus grande taille du problème scp1. Pour voir l'utilité d'incorporer le regret dans l'heuristique de construction, on compare les deux cas (avec et sans regret) et on montre la supériorité de la première. On voit au tableau 4.4, que la déviation moyenne par rapport au coût du meilleur recouvrement est respectivement pour les deux heuristiques 2.12 et 1.40 (le symbole $*$ indique que le meilleur recouvrement est trouvé).

4.5.4 L'heuristique d'amélioration

On a trouvé qu'une bonne valeur du paramètre $\%CoverSizeImp$ est 1.2. Une valeur plus grande (1.5 par exemple) donne une meilleure amélioration mais en résolvant un problème scp2 plus grand. Après qu'un recouvrement soit trouvé par l'heuristique de construction, après avoir ajouté un nombre fixé de sous-ensembles au recouvrement pour le rendre redondant, le problème scp2 est résolu par Cplex durant 20 secondes au plus.

		Construction		Amélioration		
	Meilleur	Taille scp1	Coût	Taille scp2	Coût	Time
E1	29	233x236	29*	342x59	-	6.35
E2	30	249x303	31	320x61	-	20.62
E3	27	270x262	27*	285x54	-	3.42
E4	28	255x258	29	419x64	-	10.34
E5	28	243x271	28*	365x61	-	4.20
F1	14	218x266	14*	400x32	-	3.13
F2	15	238x239	15*	486x28	-	2.34
F3	14	225x287	15	409x34	14*	4.71
F4	14	226x278	14*	500x34	-	3.96
F5	13	214x275	14	500x33	-	20.45
G1	176	507x637	176*	566x236	-	22.14
G2	154	529x613	156	575x233	155	23.63
G3	166	531x599	170	552x235	-	23.39
G4	168	524x588	171	592x231	-	Lim
G5	168	547x611	170	555x230	168*	26.99
H1	63	469x626	65	642x128	64	Lim
H2	63	494x587	65	617x121	64	Lim
H3	59	516x520	61	528x111	60	34.47
H4	58	517x568	60	572x116	59	Lim
H5	55	509x529	57	577x116	-	Lim

Tab. 4.5: Résultats des heuristiques de consruction et d'amélioration

Les résultats sont montrés au tableau 4.5. Le symbole $*$ a le même sens qu'au tableau 4.4 tandis que le symbole - signifie que le meilleur recouvrement extrait n'est pas meilleur que le recouvrement construit. La dernière colonne du tableau 4.5 donne le temps total (Lim = 40 secondes). On a identifié de bons recouvrements juste en approximant deux PRE plus petits scp1 et scp2.

4.5.5 L'heuristique lagrangienne

Encouragés par les résultats obtenus par les heuristiques de construction et d'amélioration, on décida de les imbriquer dans une seconde méthode de sous-gradients. Observons que le seul but de cette dernière est d'obtenir des recouvrements partiels distincts durant la phase de contruction par le biais de l'heuristique avec regret appliquée avec les coûts réduits. Par ailleurs, les heuristiques de contruction et d'amélioration sont efficaces mais prennent un temps important (grâce à l'usage de Cplex). Pour ces raisons on a choisi des règles spéciales. Le nombre d'itérations est fixé à 18, en commençant avec $\rho = 2$ puis

	Meilleur	Coût	Temps		Meilleur	Coût	Temps
E1	29	√	87.12	F1	14	√	77.75
E2	30	√	148.00	F2	15	√	46.95
E3	27	√	112.85	F3	14	√	80.92
E4	28	√	136.34	F4	14	√	123.66
E5	28	√	58.30	F5	13	√	154.16
G1	176	√	187.52	H1	63	√	201.05
G2	154	√	190.48	H2	63	√	196.73
G3	166	167	196.88	H3	59	60	194.32
G4	168	√	194.85	H4	58	√	199.83
G5	168	√	197.29	H5	55	√	199.24

TAB. 4.6: Résultats de l'heuristique lagrangienne sur les ensembles E, F, G et H

ρ est multiplié par 0.95 à chaque itération pour obtenir une convergence plus rapide. Typiquement, le problème scp1 a un petit nombre de contraintes alors que scp2 a un petit nombre de variables. Puisque scp1 est plus difficile (avec un plus grand nombre de variables) on attribua 16 secondes à Cplex à la première itération puis 8 plus tard. Les temps limites sont 4 et 2 secondes pour scp2. Un simple calcul montre que notre méthode nécessite autour de 190 secondes. Les résultats de l'heuristique lagrangienne sont consignés au tableau 4.6. On voit qu'on a échoué seulement sur deux instances, G3 and H3.

4.5.6 Comparaison avec des heuristiques connues

Six méthodes sont comparées avec notre heuristique, appelée LH. Il y a deux heuristiques lagrangiennes, appelées CFT [84] et CNS [85], et quatre méta-heuristiques, BC [90], BJT [88], LDW [96] et YKI [101]. Puisque toutes les sept méthodes sont capables d'identifier les meilleurs recouvrements pour les ensembles A-F, seulement les résultats obtenus sur les ensembles G et H seront comparés.

Le tableau 4.7 compare les méthodes du point de vue de la qualité de la solution trouvée. On voit que les méthodes CFT, LDW et YKI sont les plus efficaces, bien qu'une seule exécution de YKI peut échouer. Notre méthode échoue seulement deux fois et peut être classée au quatrième rang.

Les méta-heuristiques BC, BJT et YKI exécutent 10 essais chacune mais seulement un temps moyen est rapporté. Selon nous, si le meilleur recouvrement après 10 essais

	BC	BJT	CFT	CNS	LDW	YKI	LH
G1	√	√	√	√	√	√	√
G2	155	155	√	155	√	√	√
G3	√	√	√	167	√	√	167
G4	√	√	√	170	√	√	√
G5	√	√	√	169	√	√	√
H1	64	64	√	64	√	√	√
H2	64	√	√	64	√	√	√
H3	√	√	√	60	√	√	60
H4	√	√	√	59	√	√	√
H5	√	√	√	√	√	√	√

TAB. 4.7: Qualité des recouvrements obtenus par les différentes heuristiques sur les ensembles G et H

est rapporté, il faut donner le temps total, et non pas le temps moyen. Le tableau 4.8 donne les temps de calcul exprimés en secondes des sept méthodes tels que trouvés dans la littérature. Pour les méthodes BC, BJT et YKI, indiquées par le symbole [10] dans le tableau, les temps moyens sont multipliés par 10. S'agissant de la méthode LDW, le temps total de calcul n'est pas connu, mais seulement le temps d'obtention de la solution. Pour être capable de comparer les temps de calcul, ceux-ci doivent être normalisés. La référence[197] a évalué les performances d'un grand nombre d'ordinateurs en utilisant des logiciels d'algèbre linéaire. Le tableau 4.9 montre les machines utilisées par les auteurs des heuristiques. Puisque certaines d'entre elles ne figurent pas dans l'étude de Dongarra, des ordinateurs similaires sont pris en compte. A partir de l'information fournie par la référence citée, si on considère le DEC station 5000/240 comme ordinateur de référence, on voit dans la dernière colonne du tableau 4.9 que, par exemple, notre machine est environ 146 fois plus rapide. Les temps normalisés (temps équivalents DEC station 5000/240) sont fournis au tableau 4.10. On peut y voir que les heuristiques lagrangiennes CFT et CNS sont les plus rapides. Encore une fois, notre méthode peut être classée à la quatrième position.

De cette étude comparative, il s'en suit que CFT est la meilleure en tous points. Cependant, notre heuristique a la possibilité de la concurrencer, et même d'être plus rapide, pour peu que, au lieu de Cplex, une méthode plus rapide était trouvée pour approximer les deux problèmes scp1 and scp2.

	BC[10]	BJT[10]	CFT	CNS	LDW	YKI[10]	LH
G1	25776	2400	5000	1000	298.97	1800	187.52
G2	29024	2400	5000	1000	222.34	1800	190.48
G3	25212	2400	5000	1000	21.56	1800	196.88
G4	28031	2400	5000	1000	194.21	1800	194.85
G5	29571	2400	5000	1000	47.57	1800	197.29
H1	43067	2400	5000	1000	3917.08	1800	201.05
H2	46000	2400	5000	1000	238.45	1800	196.73
H3	45629	2400	5000	1000	783.20	1800	194.32
H4	44279	2400	5000	1000	1358.28	1800	199.83
H5	44881	2400	5000	1000	5.62	1800	199.24

TAB. 4.8: Temps de calcul ou temps limite des heuristiques à partir de la littérature

Heuristique	Machine utilisée	Vitesse relative
CFT	DEC station 5000/240	1
BC	SGI R4000, 100 MHz	2.5
CNS	IBM RS/6000 375, 62.5 MHz	4.8
BJT	Pentium 100 MHz	6.5
YKI	SUN Ultra SPARC II, 300 MHz	25
LDW	Pentium IV, 1.7 GHz	29.5
LH	Intel Pentium Dual Core, 2 GHz	145.8

TAB. 4.9: Machines utilisées

	BC	BJT	CFT	CNS	LDW	YKI	LH
G1	64440	15600	5000	4800	8820	45000	27340
G2	72560	15600	5000	4800	6559	45000	27772
G3	63030	15600	5000	4800	636	45000	28705
G4	70078	15600	5000	4800	5729	45000	28409
G5	73928	15600	5000	4800	1400	45000	28765
H1	107668	15600	5000	4800	115554	45000	29313
H2	115000	15600	5000	4800	7034	45000	28683
H3	114073	15600	5000	4800	23104	45000	28332
H4	110698	15600	5000	4800	40069	45000	29135
H5	112203	15600	5000	4800	166	45000	29049

TAB. 4.10: Temps normalisés (réf. DEC station 5000/240)

4.6 Conclusion

Les conséquences qui découlent de cette étude expérimentale sont les suivantes :

1. Si on veut approximer le PRE, on doit juste considérer le problème coreSCP. Celui-ci représente une petite partie du PRE original, mais comme l'a montré l'expérience, il contient toujours le meilleur recouvrement. De plus, toute heuristique efficace (telle que CFT par exemple) profiterait de cette réduction de la taille et agirait plus vite.
2. L'heuristique lagrangienne présentée est simple et facile à coder. Elle est basée sur deux nouvelles idées. La première consiste à incorporer le concept de regret dans l'heuristique gloutonne, et la seconde consiste à plonger un recouvrement dans un autre plus large pour en extraire le meilleur. L'expérimentation numérique montre que notre heuristique est efficace en identifiant des recouvrements de qualité. Elle pourrait être plus rapide si un bon moyen (en dehors de Cplex) était trouvé pour approximer les problèmes scp1 and scp2.
3. Il est attendu que l'idée sous-jacente à la définition du problème coreSCP, à partir de l'information extraite de la méthode sous-gradients, puisse s'appliquer potentiellement à tout problème d'optimisation combinatoire.
4. Les instances de PRE dites "unicoût" sont celles pour lesquelles les coûts sont identiques et les instances de type RAIL sont bien connues comme étant de très grande taille avec des coûts de deux valeurs possibles 1 ou 2. Ces instances proviennent d'un problème d'habillage posé à une compagnie italienne de chemins de fer (Ferrovie Dello Stato). Précisément à cause de leur structure des coûts, notre heuristique donne de médiocres résultats sur ces instances, puisque on n'a pu éliminer qu'une petite partie des sous-ensembles. Sans en éliminer une partie substantielle, le problème scp1 posé durant la phase de construction serait beaucoup trop grand pour être résolu avec Cplex.

Chapitre 5

Une heuristique polynomiale d'amélioration locale pour le PBC

5.1 Introduction

Les problèmes de consécutivité des 1 dans une matrice binaire sont bien connus de la communauté des chercheurs en algorithmique depuis les années 1950. Ils surviennent naturellement dans les problèmes de stockage et extraction [122, 129, 130, 198], de biologie calculatoire [124, 126, 128], d'archéologie [132], de reconnaissance de graphes d'intervalles et planaires [114], et de localisation de stations d'arrêt [18, 19]. Une majeure partie de la littérature traitant des problèmes de consécutivité des 1 est signalée dans la thèse de Dom [199].

Une matrice est dite satisfaisant la PC1s'il existe une permutation de ses colonnes de manière à ce que les 1 apparaissent consécutivement dans chaque ligne. Tucker caractérisa les matrices satisfaisant la PC1 en termes de matrices interdites [136]. Narayanaswamy and Subashini [137] proposèrent une nouvelle caractérisation en raffinant une autre due à Fulkerson et Gross [135]. Le premier algorithme polynomial de reconnaissance des matrices satisfaisant la PC1 fut découvert au milieu des années soixante par Fulkerson and Gross [135]. Plus tard, Booth et Lueker [114] donnèrent un algorithme linéaire (en la densité de la matrice binaire) en utilisant une structure spéciale, celle des PQ-arbres. Un algorithme de même complexité, mais plus simple, est proposé en [141]. Une caractéristique intéressante des matrices binaires ayant la PC1 est qu'elles sont

totalement unimodulaires. En conséquence, les problèmes d'optimisation combinatoires sont polynomialment résolubles lorsqu'ils sont restreints aux instances dont la matrice binaire des contraintes a la PC1 (voir [116]).

La plupart des matrices binaires ne satisfont pas la PC1, cependant. Lorsqu'on est confronté à de telles matrices, on doit diminuer notre prétention, soit en cherchant le nombre minimum de 0 qui doivent être remplacés par des 1 de manière à ce que la nouvelle matrice ait la PC1 ([147, 199]), soit en cherchant à supprimer un nombre minimum de colonnes de sorte que la matrice restant ait la PC1 ([137, 199]). Les deux problèmes sont $\mathbb{NP}$-durs [24]. De plus, ils ne peuvent être approximés avec garantie à moins que $\mathbb{NP} = \mathbb{P}$. Une troisième alternative est de chercher une permutation des colonnes de la matrice binaire qui minimise le nombre de B1C. C'est le problème PBC dont on a déjà parlé.

Etant donnée une matrice binaire B, soit B_π la matrice résultant d'une permutation π des colonnes de B. Formellement, le problème de décision associé à PBC est le suivant.

PBC-décision

Instance Entiers positifs m, n, k, et une $m \times n$-matrice binaire B.

Question Existe-t-il une permutation π de $(1, \cdots, n)$ telle que le nombre de coefficients de B_π vérifiant $B^i_{\pi(j)} = 1$, et soit $j = n$, soit $B^i_{\pi(j+1)} = 0$, soit au plus k ?

Il est facile de voir que le nombre de coefficients de B indiqués dans la question ci-dessus est égal au nombre de B1C. PBC-décision est $\mathbb{NP}$-complet [117]. Il est $\mathbb{NP}$-complet même lorsqu'il est restreint au matrices ayant deux 1 par ligne (voir [118]). Une bonne nouvelle est l'existence d'une heuristique polynomiale délivrant une permutation telle que le nombre de B1C ne diffère pas plus de 50% de l'optimum [119].

En vue de la difficulté de PBC, on a besoin de méthodes heuristiques, en particulier pour les grandes instances survenant en pratique. Malheureusement, en l'état actuel de nos connaissances, on ne connaît aucune telle méthode. Espérons que ce travail contribuera à combler cette lacune. Le chapitre est organisé comme suit. Dans la section suivante, on propose une heuristique polynomiale basée sur l'amélioration locale. Les résultats numériques de notre implémentation sont alors présentés à la section qui suit, et sont utilisés essentiellement pour analyser l'influence des paramètres du problème sur le temps de calcul. Quelques remarques sont consignées à la dernière section pour clore le chapitre.

5.2 L'heuristique d'amélioration locale

Observons d'abord que, s'agissant de PBC, on n'est confronté à aucun problème de "réalisabilité". On cherche une permutation, des colonnes de la matrice binaire qui minimise le nombre de B1C, et n'importe quelle permutation peut servir comme point de départ. Commençons par prouver quelques lemmes préliminaires. L'observation clé est qu'il est possible de calculer, en temps $O(m)$, le nombre de B1C créés en plaçant une colonne arbitraire α entre deux colonnes β et γ, notant que la colonne α est responsable de l'augmentation du nombre de B1C pour chaque ligne dans laquelle la séquence de caractères dans $\alpha\beta\gamma$ est 101 ou 010.

Lemma 5.1. *Soit B une $m \times 3$-matrice binaire. Le nombre d'occurences des séquences* 101 *ou* 010 *sur les lignes est*

$$\sum_{r=1}^{r=m} \left(B_2^r - B_1^r \times B_2^r - B_2^r \times B_3^r + B_1^r \times B_3^r\right)$$

où B_i^r indique le coefficient de B situé sur la ligne r et la colonne i.

Démonstration. Supposons qu'on a trois nombres binaires a, b, c, dans cet ordre, dans la ligne r. La table de calcul suivante

a	b	c	$(1-a)\,b\,(1-c) + a\,(1-b)\,c$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

nous indique que la valeur de $(1-a)\,b\,(1-c)+a\,(1-b)\,c$ est une règle de reconnaissance de 101 ou 010 dans la ligne r. L'expression est réécrite comme $b - ab - bc + ac$ pour diminuer le nombre de multiplications. En faisant la somme sur toutes les lignes de B on obtient le nombre désiré. □

Malheureusement, pour réaliser ces calculs, on ne peut tirer avantage, comme on peut le supposer, de la faible densité de la matrice binaire, puisque les 0 sont aussi importants que les 1 dans les calculs du lemme. La plupart des langages de programmation offrent, cependant, la possibilité de stocker un nombre binaire sur un "bit", de sorte qu'il suffit de $m \times n$ bits pour stocker une $m \times n$-matrice B. Par conséquent, le calcul de l'expression $(1-a)b(1-c)+a(1-b)c$ peut être réalisé comme !˜a & b & !˜c | a & !˜b & c, en utilisant la complémentation des bits (!˜), l'opérateur AND (&), et l'opérateur OR inclusif(|).

Lemma 5.2. *Si on insère une colonne j entre deux colonnes adjacentes (appelons-les i et k) dans une matrice binaire B on crée $\delta(i,j,k)$ nouveaux B1C où*

$$\delta(i,j,k) = \sum_{r=1}^{r=m} \left(B_j^r - B_i^r \times B_j^r - B_j^r \times B_k^r + B_i^r \times B_k^r\right)$$

Démonstration. Considérons n'importe quelle ligne r et les coefficients binaires successifs B_i^r, B_j^r, B_k^r. Un nouveau B1C est créé dans la ligne r si et seulement si soit $B_i^r = B_k^r = 0$ et $B_j^r = 1$, soit $B_i^r = B_k^r = 1$ et $B_j^r = 0$. Le lemme 5.1 donne le nombre de telles occurences dans les colonnes i, j, k. □

Notons que $\delta(i,j,k)$ est aussi le nombre de B1C enlevés quand la colonne j est enlevée laissant les colonnes i and k adjacentes. Maintenant, concaténons à la $m \times n$-matrice binaire B deux colonnes artificielles nulles, une en position 0 et l'autre en position $n+1$, pour obtenir une $m \times (n+2)$-matrice binaire $A = (\mathbf{0}|B|\mathbf{0})$. La position de ces deux colonnes artificielles ne change jamais. Leur seul but est de permettre des calculs intermédiaires comme on le verra plus loin. Clairement, A et B ont le même nombre minimum de B1C. Soit A_π la matrice obtenue à partir d'une permutation π des colonnes de A donnant un certain nombre de B1C, gardant à l'esprit que $\pi(0) = 0$ et $\pi(n+1) = n+1$ à jamais. On considère deux différentes façons d'améliorer π (en diminuant le nombre de B1C), soit en interchangeant deux colonnes distinctes, soit en déplaçant une colonne.

5.2.1 Amélioration par interchange de deux colonnes

Supposons que l'on inspecte la matrice A_π associée à une certaine permutation π telle que le nombre de B1C est σ. On considère le voisinage $\mathcal{N}(\pi)$, de taille $O\left(n^2\right)$, comme étant

celui de toutes les permutations qui résultent de π en interchangeant deux colonnes. On explore $\mathcal{N}(\pi)$ en cherchant une permutation donnant une valeur plus petite du nombre de B1C. Si une telle permutation n'existe pas, la procédure s'arrête. Quand une amélioration δ est obtenue via un interchange des colonnes $\pi(i)$ et $\pi(j), i \neq j$ (appelons cette nouvelle permutation π'), on met à jour $\sigma \longleftarrow \sigma - \delta, \pi'(i) \longleftarrow \pi(j), \pi'(j) \longleftarrow \pi(i)$, et on répète le processus avec π'.

Observons qu'il n'est nul besoin d'explorer le cas particulier où les colonnes $\pi(i)$ et $\pi(j)$ sont adjacentes (i.e. $j = i+1$), puisque interchanger ces deux colonnes revient à déplacer la colonne $\pi(i)$ à la position $\pi(i+1)$, et déplacer la colonne $\pi(i+1)$ une position vers la gauche. Considérons donc le cas où les colonnes $\pi(i)$ et $\pi(j)$ sont non adjacentes (i.e. il existe au moins une colonne distincte les séparant).

Proposition 5.3. *Soient*

$$\begin{aligned}
\delta_k^+ &= A^k_{\pi(i-1)} \times A^k_{\pi(j)} + A^k_{\pi(j)} \times A^k_{\pi(i+1)} + A^k_{\pi(j-1)} \times A^k_{\pi(i)} + A^k_{\pi(i)} \times A^k_{\pi(j+1)} \\
\delta_k^- &= A^k_{\pi(i-1)} \times A^k_{\pi(i)} + A^k_{\pi(i)} \times A^k_{\pi(i+1)} + A^k_{\pi(j-1)} \times A^k_{\pi(j)} + A^k_{\pi(j)} \times A^k_{\pi(j+1)} \\
\delta &= \sum_{k=1}^{k=m} \left(\delta_k^+ - \delta_k^-\right) \qquad (5.1)
\end{aligned}$$

Il existe une amélioration en interchangeant deux colonnes non adjacentes $\pi(i)$ et $\pi(j)$ si et seulement si $\delta > 0$. De plus, le nombre de B1C après interchange des deux colonnes est $\sigma - \delta$.

Démonstration. Interchanger les colonnes $\pi(i)$ et $\pi(j)$ signifie que : 1) La colonne $\pi(i)$ quitte sa position entre les colonnes $\pi(i-1)$ et $\pi(i+1)$; 2) la colonne $\pi(j)$ quitte sa position entre les colonnes $\pi(j-1)$ et $\pi(j+1)$; 3) la colonne $\pi(i)$ est insérée entre les colonnes $\pi(j-1)$ et $\pi(j+1)$; 4) et la colonne $\pi(j)$ est insérée entre les colonnes $\pi(i-1)$ et $\pi(i+1)$. En utilisant le lemme 5.2

$$\begin{aligned}
\delta_1^+ &= -\delta(\pi(i-1), \pi(i), \pi(i+1)) \\
\delta_2^+ &= -\delta(\pi(j-1), \pi(j), \pi(j+1)) \\
\delta_1^- &= \delta(\pi(j-1), \pi(i), \pi(j+1)) \\
\delta_2^- &= \delta(\pi(i-1), \pi(j), \pi(i+1)) \\
\delta &= \delta_1^+ + \delta_2^+ + \delta_1^- + \delta_2^-
\end{aligned}$$

les simplifications appropriées donnent la valeur de δ en (5.1). □

L'intérêt de l'introduction des deux colonnes artificielles aux positions 0 et $n+1$ apparaît claire quand on doit permuter la colonne en position 1 avec une autre colonne, ou quand on permute la colonne en position n avec une autre. En d'autres termes, si $i = 1$ (resp. $i = n$) prendre $\pi(i-1) = 0$ (resp. $\pi(i+1) = n+1$).

Proposition 5.4. *La complexité de la procédure d'interchange est* $O\left(mn^2(f-m)\right)$ *où* f *le nombre de 1 dans* A.

Démonstration. Explorer le voisinage $\mathcal{N}(\pi)$ prend un temps $O\left(n^2\right)$. Pour chaque voisin, les calculs en (5.1) prennent un temps $O(m)$. Puisque le nombre σ de B1C ne peut être plus petit que m, ni plus grand que f, la finitude de la procédure d'interchange est garantie, et le nombre d'améliorations successives est au plus $f-m$ au pire des cas. □

On donne le pseudo-code de la procédure d'interchange (rapellons que $A = (\mathbf{0}|B|\mathbf{0})$).

Procédure d'interchange
Input Entiers positifs m, n, σ, $m \times n$-matrice binaire B, π
Output π, σ
pour $i = 1, \cdots, n-2$
pour $j = i+2, \cdots, n$
  Comment Interchanger deux colonnes non adjacentes $\pi(i)$ et $\pi(j)$
  $\delta \longleftarrow -\delta(\pi(i-1), \pi(i), \pi(i+1)) - \delta(\pi(j-1), \pi(j), \pi(j+1))$
  $+\delta(\pi(j-1), \pi(i), \pi(j+1)) + \delta(\pi(i-1), \pi(j), \pi(i+1))$
  Si $\delta > 0$
    $\sigma \longleftarrow \sigma - \delta$
    Swaper les colonnes $\pi(i)$ et $\pi(j)$

5.2.2 Amélioration par déplacement d'une colonne

On considère le voisinage $\mathcal{N}'(\pi)$ comme étant l'ensemble de toutes les permutations que résultent de π en déplaçant une colonne, ce qui signifie que la colonne en question quitte sa position et est insérée ailleurs entre deux colonnes. La taille de $\mathcal{N}'(\pi)$ est également $O\left(n^2\right)$ car il y a n possibilités de choisir une colonne et, une fois celle-ci choisie, il reste $n-1$ places possibles pour la placer. $\mathcal{N}'(\pi)$ est scanné à la recherche d'une permutation amenant un plus petit nombre de B1C. Si une telle permutation n'existe pas, la procédure finit. Supposons qu'une amélioration δ est obtenue par déplacement de la colonne $\pi(i)$,

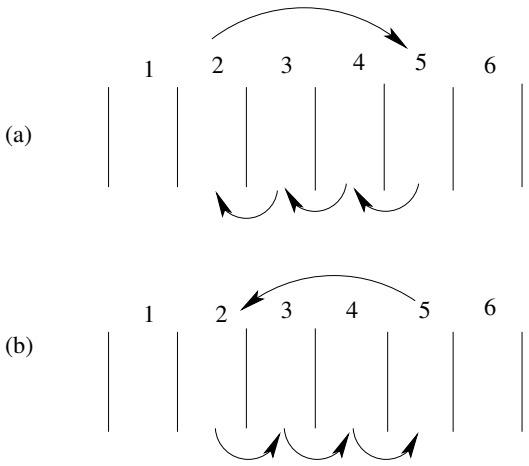

FIG. 5.1: Cas (a) Avant déplacement, $\pi = (123456)$, après déplacement de $\pi(2)$ à la position 5 $\pi' = (134526)$. Cas (b) Après déplacement de $\pi(5)$ à la position 2 $\pi' = (152346)$

et soit π' la permutation qui en résulte. Quand toutes les mises à jour sont faites, on répète la procédure en considérant π'.

Proposition 5.5. *Il existe une amélioration par déplacement de la colonne $\pi(i)$ entre les colonnes $\pi(j-1)$ et $\pi(j)$ si et seulement si $\delta > 0$ où*

$$\delta_r^+ = -A^r_{\pi(i-1)} \times A^r_{\pi(i)} - A^r_{\pi(i)} \times A^r_{\pi(i+1)} + A^r_{\pi(i-1)} \times A^r_{\pi(i+1)} \tag{5.2}$$

$$\delta_r^- = -A^r_{\pi(j-1)} \times A^r_{\pi(i)} - A^r_{\pi(i)} \times A^r_{\pi(j)} + A^r_{\pi(j-1)} \times A^r_{\pi(j)} \tag{5.3}$$

$$\delta = \sum_{r=1}^{r=m} \left(\delta_r^+ - \delta_r^-\right) \tag{5.4}$$

Démonstration. Analogue à la preuve de la proposition 5.3 en notant que

$$\delta = -\delta\left(\pi(i-1), \pi(i), \pi(i+1)\right) + \delta\left(\pi(j-1), \pi(i), \pi(j)\right)$$

et en utilisant le lemme 5.2. □

Les détails sont donnés dans un pseudo-code (voir la figure 5.1 pour une illustration). En invoquant des arguments similaires à ceux de la proposition 5.4, on prouve que la complexité de la procédure de déplacement est $O\left(mn^2(f-m)\right)$.

L'algorithme global est simple : Répéter la procédure de déplacement jusqu'à ce qu'il n'y ait plus d'amélioration possible (optimum local), puis répéter de la même manière la

Procédure de déplacement
Input Entiers positifs m, n, σ, $m \times n$-matrice binaire B, π
Output π, σ
for $i = 1, \cdots, n$
for $j = 1, \cdots, n$ $(j \neq i-1, j \neq i, j \neq i+1)$
  Comment Déplacer la colonne $\pi(i)$ entre les colonnes $\pi(j-1)$ et $\pi(j)$
  $\delta \longleftarrow -\delta(\pi(i-1), \pi(i), \pi(i+1)) + \delta(\pi(j-1), \pi(i), \pi(j))$
  Si $\delta > 0$
    $\sigma \longleftarrow \sigma - \delta$
    Si $i > j$ déplacer $\pi(i)$ à la position j dans π
    sinon déplacer $\pi(i)$ à la position $j-1$.

procédure d'interchange. Ce choix est dicté par les résultats obtenus durant l'expérimentation numérique, comme on va le voir.

5.3 Expérimentation numérique

L'algorithme d'amélioration locale est codé en C sur HP Compaq (fréquence de 2 GHz, RAM de 2 GB), et testé sur quarante-cinq instances générées aléatoirement, groupées en neuf ensembles (A - I), avec cinq instances dans chacun. Etant données la taille et la densité, les matrices binaires sont engendrées avec l'exigence que chaque colonne compte au moins un 1, et que chaque ligne contient au moins deux 1 (voir le tableau 5.2). L'ensemble C nous servira de référence. L'heuristique est aussi testée sur des données réelles ($\text{RCOV}_{1\text{km}}$- $\text{RCOV}_{10\text{km}}$) dont les tailles sont fournies au tableau 5.1. Ces dernières proviennent d'un problème de localisation de gares posé par une compagnie de chemins de fer allemande. Ils sont fournis par la compagnie, et nous ont été gracieusement envoyés par Niklaus Rüf, l'un des auteurs de [18]. Les matrices binaires associées à ces instances sont supposées avoir presque la PC1 (voir la même référence pour ce concept).

La première leçon tirée de l'expérimentation numérique est que la procédure de déplacement, quand elle appliquée seule, est plus efficace que la procédure d'interchange, qui est en retour légèrement plus rapide (voir le tableau 5.3, où les colonnes 2, 3, 4 et 5 refèrent respectivement au nombre initial et final de B1C, au nombre d'améliorations successives, et au temps de calcul en secondes). Cela est prévible néanmoins, puisque la taille du voisinage $\mathcal{N}'$ est double de la taille de $\mathcal{N}$. En effet, étant donnée une permutation π des colonnes, il y a $n(n-1)$ déplacements possibles et seulement $n(n-1)/2$ interchanges. On a testé plusieurs combinaisons de ces procédures et identifié une, qui semble la

Instance	$RCOV_{1km}$	$RCOV_{2km}$	$RCOV_{3km}$	$RCOV_{5km}$	$RCOV_{10km}$
Taille de la matrice	757×707	1196×889	1419×886	1123×593	275×165

TAB. 5.1: Taille des problèmes réels

	A	B	C	D	E	F	G	H	I
Densité(%)	2	5	10	2	5	10	2	5	10
Taille	100 x 200			100 x 500			100 x 1000		

TAB. 5.2: Caractéristiques des instances aléatoires

meilleure, et qui consiste à répéter chaque procédure seule, en commençant par la plus efficace (voir le tableau 5.4). L'algorithme résultant donne de meilleurs résultats que chacune des deux procédures en consommant un peu plus de temps.

Puisque l'algorithme d'amélioration locale débute avec une permutation arbitraire, une question intéressante est de savoir si le résultat obtenu est sensible à la permutation initiale. On a conduit une expérimentation sur l'ensemble C. Une méthode bien connue en statistiques d'ordre pour engendrer une permutation aléatoire sur $(1, 2, \cdots, n)$ est de générer n nombres alétoires uniformément distribués dans l'intervalle $[0, 1[$ et de les trier. Le tableau 5.5 montre les résultats obtenus par notre heuristique lorsqu'elle est exécutée 100 fois sur chaque instance, chaque fois avec une permutation initiale aléatoire. On peut voir que notre heuristique, comme toute heuristique d'amélioration locale, est sensible à la permutation initiale, de sorte qu'elle donne de meilleurs résultats lorsqu'elle est répétée avec différentes configurations initiales.

Les résultats sur les cinq données réelles sont présentés au tableau 5.6. On voit que la matrice binaire de $RCOV_{1km}$ satisfait la PC1 puique le nombre de B1C égale le nombre de lignes et est donc optimal. Pour les quatre autres instances, le nombre final de B1C est voisin de la borne inférieure triviale m. Les résultats sur les instances aléatoires sont montrés au tableau 5.7 où l'on donne seulement des valeurs moyennes. Il peut y être noté que le temps de calcul dépend légèrement de la densité (rappelons le nombre f) et du nombre m de lignes, mais fortement du nombre n de colonnes, confirmant ainsi la borne théorique $O\left(mn^2\left(f-m\right)\right)$.

		Déplacement seul			Interchange seul		
	Nb. init. de B1C	Nb. fin. de B1C	Nbre d'amél.	Temps de calcul (s)	Nb. fin. de B1C	Nbre d'amél.	Temps de calcul (s)
C1	1761	1341	330	0.65	1391	239	0.45
C2	1743	1341	310	0.63	1399	238	0.40
C3	1749	1341	351	0.63	1397	258	0.44
C4	1790	1366	326	0.64	1421	252	0.39
C5	1856	1418	321	0.77	1475	238	0.37

TAB. 5.3: Comparaison des deux procédures sur les instances de l'ensemble C

	Déplacement puis interchange			
	Nb. fin. de B1C	Nb. de déplac.	Nb. d'inter.	Temps de calcul (s)
C1	1336	330	5	0.84
C2	1338	310	2	0.76
C3	1337	351	3	0.78
C4	1364	326	2	0.76
C5	1418	321	0	0.82

TAB. 5.4: Résultats de l'algorithme global sur les instances du groupe C

	Meilleur	Moyen	Pire	Temps (s)
C1	1328	1346.94	1364	68.16
C2	1321	1341.14	1357	68.81
C3	1333	1348.50	1368	72.27
C4	1337	1359.23	1377	70.02
C5	1394	1416.06	1439	71.29

TAB. 5.5: Résultats de l'algorithme lorsqu'il est répété 100 fois avec une permutation initiale aléatoire

	Nb. ini. de B1C	Nb. fin. de B1C	Temps de calcul (s)
$RCOV_{1km}$	764	757	41.97
$RCOV_{2km}$	1359	1206	194.94
$RCOV_{3km}$	1813	1461	497.43
$RCOV_{5km}$	1597	1143	292.51
$RCOV_{10km}$	389	280	1.17

TAB. 5.6: Résultats sur les données réelles

	Nb. ini. de B1C	Nb. fin. de B1C	Temps de calcul (s)
A	416.0	253.0	0.45
B	955.6	695.8	0.62
C	1789.4	1358.6	0.79
D	1027.2	552.0	3.32
E	2370.2	1616.0	4.59
F	4529.2	3308.4	5.52
G	2078.8	1072.4	14.03
H	4778.4	3125.4	22.20
I	8998.8	6375.4	27.12

Tab. 5.7: Résultats sur les données aléatoires

5.4 Conclusion

On a présenté une heuristique polynomiale d'amélioration locale pour PBC où les voisinages considérés sont proposés pour la première fois. On l'a expérimenté sur un grand nombre d'instances réelles et aléatoires. Puisqu'on ne connait pas les valeurs optimales, et qu'on ne connait même pas une bonne borne inférieure, sans comparaison avec des méthodes connues, on ne peut porter un jugement ni sur la qualité des solutions obtenues, ni sur le temps de calcul. Cela étant précisé, puisque notre heuristique, comme toute heuristique d'amélioration locale, converge plus ou moins vite vers un optimum local, un point intéressant est que, en utilisant ces voisinages, une métaheuristique peut aider à s'échapper de l'optimum local et à continuer le processus d'amélioration. Par ailleurs, des voisinages plus larges peuvent être envisagés. Par exemple, en interchangeant trois colonnes, ou en en déplaçant deux. Mais, à cause de leurs tailles respectives ($O\left(n^3\right)$ and $O\left(n^4\right)$), ils doivent être explorés astucieusement. Cela constitue un domaine futur de recherche.

Chapitre 6

Problème de localisation de stops ayant quasiment la propriété de consécutivité des 1 : une approche par décomposition de Benders

6.1 Introduction

Etant donnée une $m \times n$-matrice binaire A, un revouvrement est défini comme un sous-ensemble C de $\{1, 2, \cdots, n\}$ tel que $\sum_{j \in C} a_{ij} \geq 1$ pour tout i. On peut supposer, sans perte de généralité, que A n'a pas de ligne nulle, autrement un recouvrement ne peut exister. Soit $c_j > 0$ un coût associé à la colonne $j, j = 1, \cdots, n$. Le coût du recouvrement C est $\sum_{j \in C} c_j$. Le PRE est le problème de chercher un recouvrement de coût minimum, et a pour modèle mathématique le programme linéaire en variables binaires

$$\begin{aligned} & \max \sum_{j=1}^{n} c_j x_j \\ & \sum_{j=1}^{n} a_{ij} x_j \geq 1 \qquad i = 1, \cdots, m \\ & x_j \in \{0, 1\} \qquad j = 1, \cdots, n \end{aligned} \tag{6.1}$$

où les n variables sont définies par

$$x_j = \begin{cases} 1 & \text{si } j \text{ appartient au recouvrement} \\ 0 & \text{sinon} \end{cases}$$

Un bloc de 1 consécutifs (B1C) dans la matrice A est une séquence maximale de 1 situés consécutivement sur la même ligne. La matrice A est dite ayant la propriété PC1 s'il existe une permutation des colonnes de A de sorte à mettre les 1 consécutivement dans chaque ligne (en d'autres termes, A a m B1C). En utilisant les PQ-arbres [114] on peut facilement (en temps linéaire en la densité de A) reconnaître les matrices binaires ayant la PC1. De plus, l'algorithme de reconnaissance fournit la permutation qui fait apparaître les 1 consécutifs. Par conséquent, sans perte de généralité, on peut supposer qu'une matrice binaire ayant la PC1 a la propriété que les 1 sont déjà situés consécutivement sur chaque ligne. Un fait bien connu est qu'une telle matrice est totalement unimodulaire [116].

La plupart des matrices binaires ne satisfont pas la PC1. Cependant, certaines d'entre elles satisfont "presque" la PC1. Le concept de matrices satisfaisant presque la PC1 apparaît en [18], dans le contexte du problème de localisation de gares pour la compagnie allemande de chemins de fer Deutsche Bähn, où une matrice binaire est dite satisfaisant presque la PC1 si le nombre de B1C est très petit devant $m \times n$. Sans surprise, le PRE est $\mathbb{NP}$-dur même quand il est resteint aux matrices binaires ayant presque la PC1 (penser au problème de recouvrement des arêtes d'un graphe par les sommets qui est un PRE sur la matrice d'incidence arêtes-sommets qui n'a que deux 1 par ligne, i.e. chaque ligne a au plus deux B1C).

Etant donnée une matrice binaire A, un trou dans A est une séquence maximale de 0 entre deux 1 sur la même ligne. Evidemment, A satisfait la PC1 si et seulement si elle n'a pas de trou. Notre contribution dans ce chapitre est la présentation de deux méthodes de décomposition alternatives. La première convertit le PRE en un PLM dans lequel le nombre de variables entières égale le nombre de trous dans la matrice binaire du PRE. Ce nombre est donc relié à la permutation des colonnes de la matrice. Ce qui nous amène à considérer le PBC dont le but est de minimiser le nombre de trous par une permutation des colonnes de la matrice binaire. Puisque le PBC est $\mathbb{NP}$-dur, et puisque notre but ultime est de résoudre le PRE, on se contentera d'utiliser une simple heuristique pour le premier. Cette approche est destinée aux instances du PRE dont la matrice des

contraintes a un nombre de trous plus petit que n, telles que les instances réelles testées en [18]. La seconde transformation est similaire à la première. Elle concerne les instances satisfaisant presque la PC1 mais dont le nombre de trous de la matrice des contraintes est plus grand que n, telles que les benchmark utilisés en [18]. Deux algorithmes de coupe (BD1 et BD2) sont proposés, un pour chaque transformation. Ces algorithmes sont testés sur les instances provenant de [18] et de la bibliothèque OR-library [200].

Le chapitre est ainsi organisé. La section suivante rappelle quelques règles logiques pour réduire à priori la taille du PRE. La troisième section présente la première transformation et la quatrième montre comment minimiser le nombre de trous de la matrice binaire par une permutation appropriée de ses colonnes. La cinquième section propose une manière originale, fondée sur la dualité, de construire la décomposition de Benders pour un PLM. La sixième section suivante présente l'algorithme de coupes, appelé BD1, destiné à résoudre le PLM. Dans la septième section, inspirés par la première transformation de la section 7, on montre rapidement la conversion du PRE en un PBM. La huitième section concerne l'expérimentation numérique, où cinq instances réelles et quinze instances benchmark sont testées. La matrice binaire des ces instances satisfait presque le PC1. Les deux algorithmes de coupes sont clairement supérieurs à la méthode BaB de Rüf et Schöbel [18]. Puisque l'algorithme BD2 est performant, on l'a testé sur quarante-cinq instances provenant de la OR-library. La comparison de BD2 avec la méthode de recherche arborescente en [72] montre que notre algorithme met un peu plus de temps mais continue de générer peu de coupes. Quelques remarques finissent le chapitre à la dernière section.

6.2 Preprocessing (réduction de la taille du PRE)

Des tests logiques bien connus (voir [23]) peuvent être exécutés, à priori, pour réduire la taille du PRE. Trois d'entre eux sont particulièrement utiles.

1. Si une ligne i satisfait $\sum_{j=1}^{n} a_{ij} = a_{ir} = 1$, alors r doit appartenir à tout recouvrement. Donc, la ligne i, aussi bien que la colonne r, doivent être effacées de A. De plus, chaque ligne k telle que $a_{kr} = 1$ doit aussi être effacée, puique elle est déjà recouverte.

2. Une colonne r est dite dominée par une autre colonne t si $a_{ir} \leq a_{it}, i = 1, \cdots, m$ et $c_r \geq c_t$. Dans ce cas, la colonne r doit être effacée. Elle ne peut appartenir à un recouvrement optimum puisque la colonne t recouvre toutes les lignes recouvertes par la colonne r à moindre coût.
3. Cette règle est une généralisation de la précédente. S'il existe des colonnes $r, t_1, \cdots, t_s$ telles que $a_{ir} \leq \sum_{j=1}^{s} a_{it_j}$ pour tout i et $c_r \geq \Sigma_{j=1}^{s} c_{t_j}$ alors r est dominée et doit être effacée.

L'application des règles 1-3 résulte souvent en une importante réduction de la taille comme on le verra plus tard.

6.3 Conversion du PRE en un PLM

L'idée sous-jacente à la transformation du PRE en un PLM est simple. Si l'on remplit chaque trou de la matrice binaire avec des 1, on obtient une matrice binaire ayant la PC1, donc totalement unimodulaire. Néanmoins, le remplacement d'un 0 par un 1 résulte en l'addition d'une variable dans l'inégalité correspondante. Pour maintenir la réalisabilité, une valeur équivalente doit être soustraite. Avant de continuer, considérons un exemple.

$$\begin{array}{lccccccc} \min & 2x_1 & +6x_2 & +4x_3 & +3x_4 & +x_5 & & \\ & x_1 & * & +x_3 & * & +x_5 & \geq 1 & \\ & x_1 & * & +x_3 & +x_4 & & \geq 1 & \\ & & x_2 & * & +x_4 & +x_5 & \geq 1 & \\ & x_1, & x_2, & x_3, & x_4, & x_5 & \in \{0,1\} & \end{array} \tag{6.2}$$

Il y a quatre trous dans la matrice binaire du PRE (marqués par $*$). Si on remplit les deux trous de la première contrainte avec les variables x_2 et x_4, on doit soustraire deux valeurs (disons y_1 et y_2) avec l'addition au problème de deux nouvelles contraintes $y_1 = x_2$ et $y_2 = x_4$. En associant une variable entière à chaque trou, le PRE est converti

en le PLNE suivant :

$$
\begin{array}{lccccccccccc}
\min & 2x_1 & +6x_2 & +4x_3 & +3x_4 & +x_5 & & & & & & \\
 & x_1 & \boxed{+x_2} & +x_3 & \boxed{+x_4} & +x_5 & \boxed{-y_1} & \boxed{-y_2} & & & \geq 1 \\
 & x_1 & \boxed{+x_2} & +x_3 & +x_4 & & & & \boxed{-y_3} & & \geq 1 \\
 & & x_2 & \boxed{+x_3} & +x_4 & +x_5 & & & & \boxed{-y_4} & \geq 1 \\
 & & x_2 & & & & -y_1 & & & & = 0 \\
 & & & & x_4 & & & -y_2 & & & = 0 \\
 & & x_2 & & & & & & -y_3 & & = 0 \\
 & & & x_3 & & & & & & -y_4 & = 0 \\
 & x_1, & x_2, & x_3, & x_4, & x_5 & & & & & \in \{0,1\} \\
 & & & & & & y_1, & y_2, & y_3, & y_4 & \in \mathbb{Z}_+
\end{array}
$$

Supposons que la matrice binaire A contient h trous. La transformation proposée résulte en un programme avec n variables binaires et h variables entières :

$$\min \quad \mathbf{cx} \tag{6.3}$$

$$A_1\mathbf{x} - A_2\mathbf{y} \geq \mathbf{1} \tag{6.4}$$

$$A_3\mathbf{x} - I_h\mathbf{y} = \mathbf{0} \tag{6.5}$$

$$\mathbf{x} \in \{0,1\}^n \tag{6.6}$$

$$\mathbf{y} \in \mathbb{Z}_+^h \tag{6.7}$$

La matrice $A_1 \in \{0,1\}^{m\times n}$ est obtenue à partir de A en remplissant chaque trou avec des 1. $A_2 \in \{0,1\}^{m\times h}$ est la matrice en escalier dont la $k^{\text{ème}}$ colonne est associée à la variable y_k et dont la $i^{\text{ème}}$ ligne est associée aux variables y impliquées dans la ligne i. La matrice $A_3 \in \{0,1\}^{h\times n}$ a la propriété que chaque ligne i contient exactement les 1 insérés dans le $i^{\text{ème}}$ trou. I_h est la $h \times h$-matrice identité. Le PRE et le programme défini en (6.3-6.7) sont clairement équivalents. Dès lors, $(\mathbf{x}, \mathbf{y})$ est optimal pour le dernier si et seulement si $\mathbf{x}$ est un recouvrement optimal. Puisque le bloc

$$\begin{pmatrix} A_1 \\ A_3 \end{pmatrix}$$

satisfait la PC1, il est totalement unimodulaire, et en conséquence, les contraintes d'intégralité sur $\mathbf{x}$ peuvent être relâchées. La transformation proposée résulte donc en le

PLM

$$\begin{aligned} \min \quad & \mathbf{cx} \\ & A_1\mathbf{x} - A_2\mathbf{y} \geq \mathbf{1} \\ & A_3\mathbf{x} - I_h\mathbf{y} = 0 \\ & \mathbf{x} \leq \mathbf{1} \\ & \mathbf{x} \geq \mathbf{0} \\ & \mathbf{y} \in \mathbb{Z}_+^h \end{aligned}$$

Un point important est que le nombre h de trous est crucial dans la transformation. Puisque c'est le nombre de variables entières, on l'aimerait aussi petit que possible. Heureusement, cela est possible via une permutation appropriée des colonnes de A comme on le verra à la section suivante.

6.4 Préprocessing revisité (minimiser le nombre de trous)

PBC est le problème de trouver une permutation des n colonnes de A qui minimise le nombre de B1C. Clairement, il y a h trous dans A si et seulement si il y a $m + h$ B1C. Donc, résoudre PBC est équivalent à minimiser le nombre de trous. Haddadi et Layouni [119] convertèrent PBC en un problème du voyageur de commerce (TSP) satisfaisant l'inégalité triangulaire. Puique cette transformation polynomiale préserve les rapports d'approximation, alors l'heuristique de l'arbre de poids minimum constitute une 1.5-approximation pour PBC.

La transformation de PBC en TSP se fait ainsi : (i) Ajouter à A une colonne nulle numérotée 0 pour obtenir une $m \times (n+1)$-matrice binaire $A_0 = (\mathbf{0}|A)$. (ii) Calculer la $(n+1) \times (n+1)$-matrice symétrique des "distances" $D = A_0^T (U - A_0) + (U - A_0)^T A_0$ où U est la matrice dont les coefficients sont tous des 1. (iii) Considérer le graphe sur $n+1$ sommets correspondant aux colonnes de A_0 avec la matrice des distances D, et obtenir une tournée optimale T de longueur δ. (iv) supprimer le sommet 0 (correspondant à la colonne ajoutée) du circuit hamiltonien T pour obtenir une chaîne sur n sommets. Il est prouvé en [119] que cette chaîne résout PBC, i.e. elle fournit la permutation des colonnes de A qui minimise le nombre de B1C. De plus, δ est toujours pair et le nombre

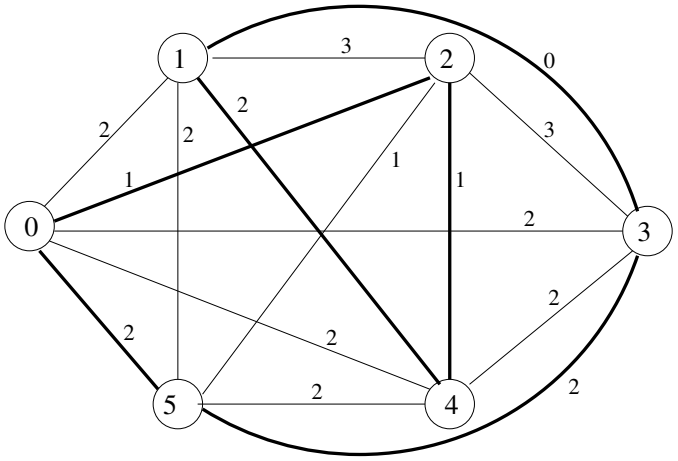

FIG. 6.1: Graphe associé à la matrice des distances D

minimum de B1C est $\delta/2$. Illustrons cette transformation sur l'exemple en (6.2). Etant donnée A_0, on calcule D.

$$A_0 = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} \quad D = \begin{pmatrix} 0 & 2 & 1 & 2 & 2 & 2 \\ & 0 & 3 & 0 & 2 & 2 \\ & & 0 & 3 & 1 & 1 \\ & & & 0 & 2 & 2 \\ & & & & 0 & 2 \\ & & & & & 0 \end{pmatrix}$$

Une tournée optimale (arêtes en gras) de longueur $\delta = 8$ est trouvée. La suppression du sommet 0 laisse une permutation (24135) des colonnes de A

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

aboutissant en un nombre minimum de $\delta/2 = 4$ B1C (ou 1 trou). Observons qu'on ne peut faire mieux puisque A contient la fameuse sous-matrice

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

En réarrangeant les variables, le PRE en (6.2) est réécrit

$$\begin{array}{lccccccc} \min & 6x_1 & +3x_2 & +2x_3 & +4x_4 & +x_5 & & \\ & & & x_3 & +x_4 & +x_5 & \geq 1 & \\ & & +x_2 & +x_3 & +x_4 & & \geq 1 & \\ & x_1 & +x_2 & & & +x_5 & \geq 1 & \\ & x_1, & x_2, & x_3, & x_4, & x_5 & \in \{0,1\} & \end{array} \tag{6.8}$$

et la transformation suggérée à la section 6.3, appliquée au problème (6.8), donne

$$\begin{array}{lccccccc} \min & 6x_1 & +3x_2 & +2x_3 & +4x_4 & +x_5 & & \\ & & & x_3 & +x_4 & +x_5 & & \geq 1 \\ & & +x_2 & +x_3 & +x_4 & & & \geq 1 \\ & x_1 & +x_2 & +x_3 & +x_4 & +x_5 & -y_1 & \geq 1 \\ & & & x_3 & +x_4 & & -y_1 & = 0 \\ & x_1, & x_2, & x_3, & x_4, & x_5 & & \leq 1 \\ & x_1, & x_2, & x_3, & x_4, & x_5 & & \geq 0 \\ & & & & & & y_1 & \in \mathbb{Z}_+ \end{array}$$

Comme on peut voir, il vaut la peine de résoudre PBC puique cela résulte en une diminution du nombre de variables entières. Dans notre exemple, au lieu de quatre, on n'a plus qu'une seule variable entière. Cependant, puisque PBC est $\mathbb{NP}$-dur, on utilise une simple heuristique du plus proche voisin dans notre implémentation.

Supposons, après la phase de préprocessing, que la matrice A des constraintes du PRE contient h trous. Sans perte de généralié, supposons que, après réarrangement des lignes, A a ses p (éventuellement nul) premières lignes sans trou, où

$$A = \begin{pmatrix} P \\ B \end{pmatrix}$$

avec $P \in \{0,1\}^{p \times n}$ et $B \in \{0,1\}^{(m-p) \times n}$. Il n'y a pas de trou dans P mais il y a h trous dans B. Le PRE réécrit

$$\begin{array}{lll} \min & \mathbf{cx} & \\ & P\mathbf{x} & \geq \mathbf{1} \\ & B\mathbf{x} & \geq \mathbf{1} \\ & \mathbf{x} & \in \{0,1\}^n \end{array}$$

est transformé en le PLM

$$(\text{MIP}) \begin{cases} \min & \mathbf{cx} & \\ & P\mathbf{x} & \geq \mathbf{1} \\ & B_1\mathbf{x} + B_2\mathbf{y} & \geq \mathbf{1} \\ & B_3\mathbf{x} - I_h\mathbf{y} & = 0 \\ & -\mathbf{x} & \geq -\mathbf{1} \\ & \mathbf{x} \geq \mathbf{0} & \\ & \mathbf{y} \leq \theta & \\ & \mathbf{y} \in \mathbb{Z}_+^h & \end{cases} \tag{6.9}$$

où $B_1 \in \{0,1\}^{(m-p)\times n}, B_2 \in \{0,1\}^{(m-p)\times h}, B_3 \in \{0,1\}^{h\times n}$ (notons que $\mathbf{y} \leq \theta = B_3\mathbf{1}$ est une contrainte valide).

Rappelons que cette transformation est convenable pour des instances du PRE telles que le nombre de trous est très petit. C'est exactement le cas des instances réelles qui, non seulement satisfont la P1C, mais contiennent un nombre de trous beaucoup plus petit que n.

6.5 Une autre façon de présenter la décomposition de Benders

La décomposition de Benders est particulièrement convenable pour résoudre (MIP), un problème de la forme

$$\begin{cases} \min & \mathbf{cx} & \\ & F\mathbf{x} + G\mathbf{y} & \geq \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} & \\ & \mathbf{y} \in \mathbb{Z}_+^h & \end{cases} \tag{6.10}$$

Notre but est d'éliminer le vecteur $\mathbf{x}$ de (6.10). Cela peut être fait en écrivant le problème ainsi

$$\begin{cases} \min & z & \\ & z \geq & \min \mathbf{cx} \\ & & F\mathbf{x} \geq \mathbf{b} - G\mathbf{y} \\ & & \mathbf{x} \geq \mathbf{0} \\ & \mathbf{y} \in \mathbb{Z}_+^h & \end{cases}$$

et en passant au dual du problème en $\mathbf{x}$.

$$\begin{cases} \min & z & \\ & z \geq & \max \mathbf{u} (\mathbf{b} - G\mathbf{y}) \\ & & \mathbf{u}F \leq \mathbf{c} \\ & & \mathbf{u} \geq \mathbf{0} \\ & \mathbf{y} \in \mathbb{Z}_+^h & \end{cases}$$

Supposons que $\mathbf{c}$ est positif (comme c'est le cas du PRE). Le polyèdre $\{\mathbf{u}F \leq \mathbf{c}, \mathbf{u} \geq \mathbf{0}\}$ est non vide et a un nombre fini de points extrêmes $\mathbf{u}^{(r)}, r \in R$. Par conséquent, la contrainte

$$z \geq \max \{\mathbf{u} (\mathbf{b} - G\mathbf{y}) \,|\mathbf{u}F \leq \mathbf{c}, \mathbf{u} \geq \mathbf{0}\}$$

qui est équivalente à

$$z \geq \mathbf{u} (\mathbf{b} - G\mathbf{y}) \;\; \forall \mathbf{u}, \mathbf{u}F \leq \mathbf{c}, \mathbf{u} \geq \mathbf{0}$$

peut être remplacée par

$$z \geq \mathbf{u}^{(r)} (\mathbf{b} - G\mathbf{y}) \,, r \in R$$

Il ne reste plus qu'à assurer la réalisabilité du programme linéaire

$$(\mathrm{P}) \begin{cases} \min \mathbf{cx} \\ F\mathbf{x} \geq \mathbf{b} - G\mathbf{y} \\ \mathbf{x} \geq \mathbf{0} \end{cases}$$

i.e. éviter les valeurs de $\mathbf{y}$ qui mènent à l'irréalisabilté. Observons que (P) est irréalisable si et seulement si le programme linéaire

$$\begin{aligned} \min \quad & \alpha \\ & F\mathbf{x} + \mathbf{1}\alpha \geq \mathbf{b} - G\mathbf{y} \\ & \mathbf{x} \geq \mathbf{0}, \alpha \geq 0 \end{aligned}$$

a une valeur optimale $\alpha > 0$. En passant au dual, (P) est irréalisable si et seulement si la valeur du programme linéaire

$$\begin{aligned} \max \quad & \mathbf{v}\,(\mathbf{b} - G\mathbf{y}) \\ & \mathbf{v}F \leq \mathbf{0} \\ & \mathbf{v}\mathbf{1} \leq 1 \\ & \mathbf{v} \geq \mathbf{0} \end{aligned}$$

est positive. Le cône polyédral $\{\mathbf{v}F \leq 0, \mathbf{1}\mathbf{v} \leq 1, \mathbf{v} \geq \mathbf{0}\}$ a un nombre fini de générateurs $\mathbf{v}^{(t)}, t \in T$. Donc (P) est réalisable si et seulement si

$$\max \{\mathbf{v}\,(\mathbf{b} - G\mathbf{y}) \,|\mathbf{v}F \leq \mathbf{0}, \mathbf{v}\mathbf{1} \leq 1, \mathbf{v} \geq \mathbf{0}\} \leq 0$$

Cette contrainte est équivalente à

$$\mathbf{v}\,(\mathbf{b} - G\mathbf{y}) \leq 0 \ \forall \mathbf{v}, \mathbf{v}F \leq \mathbf{0}, \mathbf{v}\mathbf{1} \leq 1, \mathbf{v} \geq \mathbf{0}$$

et peut être remplacée par

$$\mathbf{v}^{(t)}\,(\mathbf{b} - G\mathbf{y}) \leq 0, t \in T$$

Maintenant, le PLM défini en (6.10) est clairement équivalent à

$$(\text{MP}) \begin{cases} \min & z & & \\ & z - \mathbf{u}^{(r)}\,(\mathbf{b} - G\mathbf{y}) & \geq 0 & r \in R \\ & -\mathbf{v}^{(t)}\,(\mathbf{b} - G\mathbf{y}) & \geq 0 & t \in T \\ & \mathbf{y} \in \mathbb{Z}_+^h & & \end{cases}$$

L'inconvénient est que (MP) a typiquement un nombre astronomique de contraintes. Le résoudre serait une tâche impraticable. C'est là où l'algorithme itératif de coupes entre en jeu : Générer les contraintes itérativement dans un programme maître réduit avec

l'espoir que leur nombre sera raisonnablement petit.

6.6 L'algorithme de coupes BD1

La décomposition de Benders est particulièrement indiquée pour résoudre (MIP). Depuis le fameux article [151] (voir aussi [163]), cette décomposition a eu un énorme succès dans plusieurs domaines d'applications tels que la conception de réseaux [10, 155–158], l'affectation [159, 160], le routage et l'ordonnancement [36, 161]. On donne simplement un pseudo-code de l'algorithme.

Algorithme BD1

(0) Exécuter la phase de préprocessing (réduction de la taille, approximation de PBC)
$R \longleftarrow \varnothing, T \longleftarrow \varnothing, UB \longleftarrow \infty, i \longleftarrow 0$
Commencer avec $\mathbf{y} = \theta = B_3\mathbf{1}$

(1) Résoudre le programme linéaire

$$(\text{SP}) \begin{cases} \min & w = \mathbf{u}_1\mathbf{1} + \mathbf{u}_2\left(\mathbf{1} - B_2\mathbf{y}\right) + \mathbf{u}_3\mathbf{y} - \mathbf{u}_4\mathbf{1} & \\ & \mathbf{u}_1 P + \mathbf{u}_2 B_1 + \mathbf{u}_3 B_3 - \mathbf{u}_4 & \leq \mathbf{c} \\ & \mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_4 & \geq \mathbf{0} \end{cases}$$

Si (SP) a une solution optimale $(\mathbf{u}_1^*, \mathbf{u}_2^*, \mathbf{u}_3^*, \mathbf{u}_4^*)$ avec une valeur w^* alors
$R \longleftarrow R \cup \{(\mathbf{u}_1^*, \mathbf{u}_2^*, \mathbf{u}_3^*, \mathbf{u}_4^*)\}$
Si $UB > w^*$ alors $UB = w^*$
Aller à (2)
Si (SP) est non borné (soit $(\mathbf{v}_1^*, \mathbf{v}_2^*, \mathbf{v}_3^*, \mathbf{v}_4^*)$ un rayon extrémal) alors
$T \longleftarrow T \cup \{(\mathbf{v}_1^*, \mathbf{v}_2^*, \mathbf{v}_3^*, \mathbf{v}_4^*)\}$

(2) Résoudre le PLNE

$$(\text{RMP}) \begin{cases} \min & z & & \\ & z - \left(\mathbf{u}_2 B_2 + \mathbf{u}_3\right)\mathbf{y} & \geq \mathbf{u}_1\mathbf{1} + \mathbf{u}_2\mathbf{1} - \mathbf{u}_4\mathbf{1} & \mathbf{u} \in R \\ & -\left(\mathbf{v}_2 B_2 + \mathbf{v}_3\right)\mathbf{y} & \geq \mathbf{v}_1\mathbf{1} + \mathbf{v}_2\mathbf{1} - \mathbf{v}_4\mathbf{1} & \mathbf{v} \in T \\ & \mathbf{y} \leq \theta & & \\ & \mathbf{y} \in \mathbb{Z}_+^h & & \end{cases}$$

Soit $\mathbf{y}$ une solution optimale donnant une valeur z^*
$LB \longleftarrow z^*$
Si $LB = UB$ alors STOP sinon aller à (1)

Faisons quelques commentaires à propos de l'algorithme. D'abord notons que $\mathbf{u}_1 = \mathbf{0}, \mathbf{u}_2 = \mathbf{0}, \mathbf{u}_3 = \mathbf{0}, \mathbf{u}_4 = \mathbf{0}$ constitue une solution réalisable du problème (SP) puisque $c > 0$. Donc, soit (SP) a une solution optimale, soit il est non borné. D'un autre côté, en supposant qu'il n'y a pas de ligne nulle dans la matrice des contraintes, le PRE a toujours

un recouvrement optimal. En conséquence, le problème (PMR) ne peut être irréalisable. De plus, l'adjonction de la contrainte valide $\mathbf{y} \leq \theta$ garantit sa compacité. Donc il doit posséder une solution optimale à chaque itération, et la seule façon de s'arrêter est de satisfaire l'égalité entre les deux bornes. Puisque la taille du programme linéaire LP (SP) est $n \times (m + n + h)$, on voit un autre bénéfice de minimiser le nombre de trous h. Pour finir, notons que la séquence des bornes inférieures LB est non décroissante.

Appliquons l'algorithme BD1 à notre exemple d'accompagnement. Puisque $\mathbf{x} = (1, 1, 1, 1, 1)$ est un recouvrement évident, on commence avec $y_1 = B_3\mathbf{x} = 2$ ce qui assure une solution réalisable de

$$
\begin{array}{llrrrrrrrrrl}
 & \max & u_1 & +u_2 & +3u_3 & +2u_4 & -u_5 & -u_6 & -u_7 & -u_8 & -u_9 & \\
 & & & & u_3 & & -u_5 & & & & & \leq 6 \\
 & & & u_2 & +u_3 & & & -u_6 & & & & \leq 3 \\
\text{(SP)} & & u_1 & +u_2 & +u_3 & +u_4 & & & -u_7 & & & \leq 2 \\
 & & u_1 & +u_2 & +u_3 & +u_4 & & & & -u_8 & & \leq 4 \\
 & & u_1 & & +u_3 & & & & & & -u_9 & \leq 1 \\
 & & u_1, & u_2, & u_3, & & u_5, & u_6, & u_7, & u_8, & u_9 & \geq 0
\end{array}
$$

La solution optimale est $(0, 0, 1, 3, 0, 0, 2, 0, 0)$ avec $UB = 7$. Elle induit la coupe $-4y_1 + z \geq -1$. On résout

$$
\begin{array}{lrrr}
 & \min & z & \\
 & -4y_1 & +z & \geq -1 \\
\text{(RMP)} & y_1 & & \leq 2 \\
 & y_1 & & \in \mathbb{Z}_+
\end{array}
$$

et obtient la solution optimale $y_1 = 0, LB = 0$. Cette valeur de y_1 est utilisée pour calculer la nouvelle fonction objectif de (SP)

$$u_1 + u_2 + 2u_3 + u_4 - u_5 - u_6 - u_7 - u_8 - u_9$$

La solution de (SP) donne la solution optimale $(1, 3, 0, -2, 0, 0, 0, 0, 0)$ avec $UB = 3$. On ajoute à (RMP) la coupe $2y_1 + z \geq 4$ et on résout

$$
\begin{array}{lllll}
 & \min & & z & \\
 & & -4y_1 & +z & \geq -1 \\
\text{(RMP)} & & 2y_1 & +z & \geq 4 \\
 & & y_1 & & \leq 2 \\
 & & y_1 & & \in \mathbb{Z}_+
\end{array}
$$

La solution optimale est $y_1 = 1$ avec $LB = 3$. Ce qui finit l'algorithme.

6.7 Conversion du PRE en un PMB

Bien qu'une matrice binaire puisse satisfaire presque la PC1, lorsque le nombre de trous est plus grand que le nombre de colonnes n, la transformation présentée à la section 6.3 n'est plus adéquate. En effet, quel intérêt a-t-on à transformer un programme binaire avec n variables en un PLM avec plus de n variables entières ? Heureusement, une simple astuce permet de surmonter cette difficulté : Au lieu d'associer une variable entière à chaque trou, associons autant de variables binaires que de zéros dans le trou. Considérons l'exemple en (6.8). On associe à l'unique trou les deux variables binaires y_1 and y_2.

$$
\begin{array}{lllllllll}
\min & 6x_1 & +3x_2 & +2x_3 & +4x_4 & +x_5 & & & \\
 & & & x_3 & +x_4 & +x_5 & & & \geq 1 \\
 & & +x_2 & +x_3 & +x_4 & & & & \geq 1 \\
 & x_1 & +x_2 & +x_3 & +x_4 & +x_5 & -y_1 & -y_2 & \geq 1 \\
 & & & x_3 & +x_4 & & -y_1 & -y_2 & = 0 \\
 & x_1 & x_2 & x_3 & x_4 & x_5 & & & \leq 1 \\
 & x_1, & x_2, & x_3, & x_4, & x_5 & & & \geq 0 \\
 & & & & & & y_1, & y_2 & \in \{0, 1\}
\end{array}
$$

Bien que le nombre (appelons-le s) de variables binaires du PMB ne peut être prévu, il sera toujours plus petit que n. Comme il peut être observé, cette transformation est

analogue en tous points à celle de la section 6.3

$$(\text{PMB}) \begin{cases} \min & \mathbf{cx} & \\ & P\mathbf{x} & \geq \mathbf{1} \\ & B_1\mathbf{x} + B_2\mathbf{y} & \geq \mathbf{1} \\ & B_3\mathbf{x} - B_4\mathbf{y} & = 0 \\ & -\mathbf{x} & \geq -\mathbf{1} \\ & \mathbf{x} \geq \mathbf{0} & \\ & \mathbf{y} \in \{0,1\}^s & \end{cases}$$

où $B_1 \in \{0,1\}^{(m-p)\times n}$, $B_2 \in \{0,1\}^{(m-p)\times s}$, $B_3 \in \{0,1\}^{h\times n}$, et $B_4 \in \{0,1\}^{h\times s}$. L'algorithme BD2 est similaire à BD1, et peut être facilement décrit. Remarquons qu'il reste utile de minimiser le nombre de trous puisque la taille $n \times (m + n + h)$ du sous-problème (SP) en dépend.

6.8 Expérimentation numérique

Les deux algorithmes BD1 et BD2 sont codés en C sur HP Compaq (fréquence de 2.4 GHz, RAM de 2 GB), et testés sur soixante-cinq instances. Vingt d'entre elles satisfont presque la PC1 : Cinq instances réelles (R_{1km} à R_{10km}, fournis par la compagnie Deutsche Bähn) et quinze (A_1 à C_5) aléatoirement engendrés, tous fournis par N. Rüf, un des auteurs de [18]. Ces instances sont testés dans cette référence à l'aide d'un algorithme BaB. Puisque nos deux algorithmes donnent de très bons résultats sur ces instances, on décida de les tester sur quarante-cinq instances provenant de la OR-library (http ://-people.brunel.ac.uk/˜mastjjb/jeb/info.html). Les deux problèmes (SP) et (RMP) dans le pseudo-code sont résolus à l'aide d'un logiciel de commerce Cplex 10.1.

On commence par appliquer BD1 sur les instances réelles. Les résultats détaillés sont fournis au tableau 6.1. La colonne 1 donne le nom de l'instance, la colonne 2 la densité (en pourcentage) de la matrice binaire. Les colonnes 3 à 8 montrent les résultats de la phase de préprocessing. La colonne 3 donne la taille originale de l'instance et la colonne 4, la taille réduite par application des règles de la section 6.2. Les colonnes 5 et 6 indiquent le nombre de trous de la matrice binaire avant et après approximation de PBC. La colonne 7 donne le nombre p de lignes sans trou. La colonne 8 montre le temps

		Preprocessing						Algo. de coupes BD1		
		Réduction de la taille		Approx. PBC			Tps			Tps
	Dens.	Avant	Après	Avant	Après	p	pré.	Optim.	Cou.	total
R_{1km}	0.002	757x707	2x9	0	0	2	0.00	856943	0	0.01
R_{2km}	0.014	1196x889	46x123	22	2	44	0.03	1005524	2	0.04
R_{3km}	0.022	1419x886	102x226	91	7	95	0.06	901077	46	0.34
R_{5km}	0.043	1123x593	78x178	97	2	76	0.03	505052	8	0.07
R_{10km}	0.203	275x165	10x39	11	1	9	0.00	139370	2	0.00

TAB. 6.1: Résultats de l'algorithme BD1 sur les instances réelles

du préprocessing. Les trois dernières colonnes présentent les résultats proprement dits de l'algorithme de coupes. Elles rapportent respectivement la valeur optimale, le nombre de coupes produites, et le temps total de calcul.

Plusieurs observations peuvent être faites à la vue du tableau 6.1. (i) Les matrices binaires associées aux instances sont très creuses. Habituellement, la faible densité est un signe que l'instance est relativement facile à résoudre. (ii) La plupart des lignes de l'instance originale sont recouvertes par une seule colonne, de sorte que la phase de préprocessing permet de réduire substantillement la taille des instances. (iii) La simple heuristique du plus proche voisin obtient une diminution substantielle du nombre de trous (et donc du nombre de variables entières du PLM). Il apparaît que non seulement les instances réelles satisfont presque la PC1, mais elles ont un nombre de trous beaucoup plus petit que n. (iv) Seulement quelques coupes sont nécessaires pour trouver l'optimum en moins d'une demi-seconde.

La comparaison de l'algorithme BD1 avec l'algorithme BaB de Rüf et Schöbel est proposée au tableau 6.2 où les titres sont parlants à l'exception de la colonne intitulée Cou. qui donne le nombre de coupes générées. Bien que notre ordinateur soit beaucoup plus rapide que le vieux Pentium I de Rüf et Schöbel, clairement, la comparaison montre la supériorité de notre méthode. Il y a, en particulier, une instance (R_{3km}) qui a nécessité l'exploration d'un arbre de plus de 57000 noeuds, sans obtenir la solution optimale dans une limite prescrite de 3600 secondes.

Ensuite, l'algorithme BD2 est appliqué aux instances alétoires satisfaisant presque la PC1. Les résultats sont montrés au tableau 6.3, qui est similaire au tableau 6.1 exceptée la neuvième nouvelle colonne qui indique le nombre de variables binaires du PMB. Observons que la phase de préprocessing ne fournit pas des résultats aussi spectaculaires que ceux du tableau 6.1. Cependant, l'algorithme de coupes continue de trouver des

	Rüf and Schöbel Pentium I (200MHz)			Algorithme BD1 HP Compaq (2 GHz)		
	Arbre	Temps	Opt ?	Cou.	Temps	Opt ?
R_{1km}	1	0	oui	0	0.15	oui
R_{2km}	9,841	338	oui	2	0.11	oui
R_{3km}	57,681	Limit	non	46	0.21	oui
R_{5km}	4,860	94	oui	8	0.13	oui
R_{10km}	2	0	oui	2	0.11	oui

TAB. 6.2: Comparaison de l'algorithme BD1 avec l'algorithme BaB de Rüf et Schöbel sur les instances réelles

		Preprocessing						Algo. de coupes BD2			
		Réduction de la taille		Approx. PBC			Tps	Var.			Tps
	Dens.	Avant	Après	Avant	Après	p	pré.	bin.	Opt.	Cou.	total
A_1	0.99	100x95	95x61	199	175	14	0.00	59	320	28	0.15
A_2	0.95	100x92	65x52	103	93	23	0.00	49	552	26	0.11
A_3	0.83	100x92	100x69	193	180	18	0.01	66	411	37	0.21
A_4	0.22	100x88	66x56	111	105	18	0.00	53	540	27	0.13
A_5	0.11	100x98	73x62	119	115	19	0.00	58	532	26	0.11
B_1	Irréalisable										
B_2	Irréalisable										
B_3	Irréalisable										
B_4	Irréalisable										
B_5	Irréalisable										
C_1	0.14	100x100	77x57	188	136	7	0.00	54	939	41	0.19
C_2	0.62	100x100	89x73	291	207	6	0.00	71	782	42	0.20
C_3	0.86	100x99	84x66	221	156	3	0.00	64	824	57	0.26
C_4	Irréalisable										
C_5	0.47	100x100	90x73	269	188	5	0.00	71	898	50	0.29

TAB. 6.3: Résultats de l'algorithme BD2 sur les instances aléatoires

solutions optimales en moins d'une seconde, et sans générer plus de soixante coupes. Ce qui est surprenant est que les instances B_1-B_5 et C_4 sont irréalisables (i.e. au moins une ligne est nulle dans la matrice binaire des contraintes).

La comparaison de l'algorithme BD2 avec celui de Rüf et Schöbel est proposée au tableau 6.4. Si les deux algorithmes se comportent à peu près de la même façon sur les instances de type A, BD2 est clairement supérieur sur les instances plus difficiles du type C. Exceptée l'instance C1, l'algorithme BaB échoue toujours dans le temps precrit de 3600 secondes alors que BD2 finit toujours avec une solution optimale en moins d'une seconde.

Puisque l'algorithme BD2 obtint de bons résultats sur les instances ayant presque la

	Rüf and Schöbel Pentium I (200MHz)			Algorithme BD2 HP Compaq (2 GHz)		
	Arbre	Temps	Opt ?	Cou.	Temps	Opt ?
A_1	44	0	oui	28	0.15	oui
A_2	23	0	oui	26	0.11	oui
A_3	105	0	oui	37	0.21	oui
A_4	133	0	oui	27	0.13	oui
A_5	35	0	oui	26	0.11	oui
C_1	243,522	1571	oui	41	0.19	oui
C_2	580,303	Lim	non	42	0.20	oui
C_3	464,253	Lim	non	57	0.26	oui
C_5	390,353	Lim	non	50	0.29	oui

TAB. 6.4: Comparason de l'algorithme BD2 avec celui de Rüf et Schöbel sur les instances aléatoires (Lim = 3600 secondes)

PC1, et bien qu'il soit mis au point spécialement pour ces instances, on l'a testé sur des instances générales. Les résultats sont consignés aux tableaux 6.5 et 6.6 où il peut être observé que le nombre de trous est sensible à la densité de la matrice des contraintes. Dans le tableau 6.7, proposant la comparaison de BD2 avec l'algorithme amélioré de Beasley [72], on donne des valeurs moyennes qui sont plus explicites. Du point de vue du temps de calcul, l'algorithme BaB est plus rapide que l'algorithme de coupes BD2 sur les instances générales. Mais la taille de l'arbre de décision dans l'algorithme de Beasley augmente rapidement avec la taille du problème, tandis que le nombre de coupes de l'algorithme BD2 n'augmente que très peu. Cette observation suggère que l'algorithme de coupes est moins sensible à la taille du problème, ce qui est un fait plutôt prometteur.

6.9 Conclusion

On a présenté deux méthodes alternatives de décomposition pour le PRE. Ces approches sont destinées essentiellement aux instances satisfaisant presque la PC1. Les deux algorithmes BD1 et BD2 ont résolu toutes les instances très rapidement. Les deux algorithmes surclassent l'algorithme BaB de Rüf and Schöbel, et constituent de robustes méthodes pour de telles instances. En fait, notre approche (par matrices satisfaisant presque la PC1, et donc totalement unimodulaires) est plus générale, et peut être directement

		Preprocessing						Algo. de coupes BD2			
		Reducing size		Solving CBM			Tps	Var.			Tps
	Dens	Avant	Après	Avant	Après	p	pré.	bin.	Opt.	Cou.	total
4.1	2	200x1000	165x153	438	306	8	0.04	150	429	104	1.32
4.2			195x214	797	578	3	0.08	212	512	147	4.07
4.3			192x228	832	603	4	0.09	226	516	139	4.83
4.4			188x197	680	487	3	0.07	195	494	154	4.43
4.5			189x213	762	556	4	0.08	210	512	134	3.12
4.6			200x231	845	618	1	0.08	229	560	159	5.43
4.7			177x176	559	393	6	0.04	173	430	122	2.91
4.8			188x217	770	561	1	0.05	215	492	140	3.96
4.9			195x242	857	622	1	0.10	240	641	167	10.62
4.10			170x183	573	401	9	0.05	181	514	117	2.03
5.1	2	200x2000	198x222	854	623	1	0.08	220	253	147	7.17
5.2			195x263	1027	739	2	0.11	260	302	155	11.34
5.3			161x162	465	325	9	0.04	159	226	117	2.20
5.4			190x222	797	562	1	0.09	220	242	119	2.62
5.5			161x162	465	325	9	0.04	159	211	117	2.18
5.6			181x201	671	475	2	0.07	199	213	113	2.05
5.7			186x206	718	504	5	0.07	204	293	117	3.07
5.8			196x247	881	630	0	0.09	245	288	157	6.36
5.9			194x234	850	612	4	0.08	232	279	149	4.30
5.10			181x212	685	494	2	0.08	209	265	127	2.91
6.1	5	200x1000	200x232	2100	1681	0	0.09	230	138	134	16.28
6.2			200x276	2627	2080	0	0.09	274	146	135	18.21
6.3			200x264	2548	2041	0	0.09	262	145	114	9.08
6.4			200x224	2056	1648	0	0.06	222	131	118	6.26
6.5			200x279	2558	2026	0	0.13	277	161	137	20.61

TAB. 6.5: Résultats de l'algorithme BD2 sur les instances générales du type 4, 5, 6 (OR-library)

adaptée au PLNE

$$\begin{aligned} \min \quad & \mathbf{cx} \\ & A\mathbf{x} \geq \mathbf{b} \\ & \mathbf{x} \in \mathbb{Z}_+^n \end{aligned}$$

pour peu que la matrice A satisfasse presque la PC1.

		Preprocessing						Cutting-plane alg. BD2			
		Reducing size		Solving CBM			Pre.	Bin.			Tot.
	Dens	Before	After	Before	After	p	time	var.	Opt.	cuts	time
A1	2	300x3000	300x406	2503	1953	0	0.36	403	253	207	44.24
A2			300x411	2479	1940	0	0.38	409	252	212	61.59
A3			300x411	2502	1948	1	0.40	409	232	200	38.45
A4			300x394	2395	1865	0	0.36	392	234	187	22.78
A5			300x406	2480	1917	0	0.39	404	236	191	18.32
B1	5	300x3000	300x564	8189	6699	0	0.72	562	69	180	100.97
B2			300x572	8220	6710	0	0.74	570	76	214	342.31
B3			300x623	8950	7337	0	0.84	621	80	184	105.77
B4			300x576	8255	6797	0	0.73	574	79	194	263.78
B5			300x571	8172	6750	0	0.75	569	72	191	95.29
C1	2	400x4000	400x546	4387	3548	0	0.96	544	227	241	67.61
C2			400x578	4676	3772	0	1.10	576	219	272	141.25
C3			400x626	5130	4151	0	1.28	624	243	278	352.48
C4			400x579	4697	3802	0	1.08	577	219	268	131.15
C5			400x572	4605	3753	0	1.07	570	215	270	102.93
D1	5	400x4000	400x868	16681	14061	0	2.42	866	60	228	383.00
D2			400x914	17609	14780	0	2.68	912	66	215	214.07
D3			400x894	17138	14457	0	2.62	892	72	229	353.30
D4			400x859	16357	13793	0	2.42	857	62	249	877.99
D5			400x853	16351	13765	0	2.35	851	61	216	272.10

TAB. 6.6: Résultats de BD2 sur les instances générales du type A,B,C,D

	Algorithme de Beasley CRAY X-MP/28		Algorithme BD2 HP Compaq (2GHz)	
Ensemble	Taille de l'arbre	Temps de calcul	Nombre de coupes	Temps de calcul
4	1	1.4	138.3	4.27
5	1.4	2.4	131.8	4.42
6	54.0	6.4	127.6	14.09
A	226.8	18.2	199.4	37.07
B	1194.0	57.3	192.6	181.63
C	1727.2	105.0	265.8	159.08
D	5478.8	255.0	227.4	420.09

TAB. 6.7: Comparaison de l'algorithme BD2 avec celui de Beasley sur les instances générales

Chapitre 7

Conclusion

On a étudié trois problèmes d'optimisation combinatoire : le problème de localisation de stops dans un réseau de transport public (PLS), le problème de recouvrement d'ensemble (PRE) et celui de minimiser le nombre de blocs de 1 consécutifs dans une matrice binaire (PBC). Chacun des trois problèmes est NP-dur. Le premier est réduit au second au chapitre 2, où un état de l'art est proposé pour chacun des trois problèmes. On a ensuite présenté la panoplie des techniques utilisées dans notre étude au chapitre 3. On a proposé deux algorithmes pour le PRE, une heuristique lagrangienne et un algorithme exact de coupes. Chacune des deux méthodes a fait l'objet d'une expérimentation numérique rigoureuse, avec des comparaisons avec des méthodes existantes à l'issue desquelles nos méthodes ont montré leur supériorité. L'algorithme de coupes a fait l'objet d'un article soumis à « Journal of Combinatorial Optimization » publié par Springer. Une heuristique d'amélioration locale est proposée pour le PBC, fondée sur des bases théoriques, elle a fait l'objet d'une publication au journal « Information Processing Letters » publié par Elsevier.

Les axes futurs de recherche identifiés :

1. Montrer que l'heuristique de réduction de la taille du chapitre3, fondée sur l'information fournie par la méthode de sous-gradients, et appliquée au PRE, peut être appliquée à tout problème d'optimisation combinatoire.
2. Les voisinages de taille $O\left(n^2\right)$ décrits au chapitre 4 peuvent être élargis, et donc devenir plus coûteux en termes de temps d'exploration. On pense à l'idée d'interchanger trois colonnes ou de déplacer deux colonnes, mais on peut penser aussi à

un déplacement en chaîne. Ce dernier peut avoir une taille exponentielle, et doit être judicieusement exploré.

3. On pense aussi à utiliser ces voisinages avec une méta-heuristique qui puisse nous aider à nous échapper de l'optimum local pour continuer la procédure d'amélioration.
4. On essaiera de montrer que notre approche au chapitre 6 (par matrices satisfaisant presque la PC1, et donc totalement unimodulaires) est plus générale, et peut être adaptée au PLNE avec une matrice binaire des contraintes, pour peu que celle-ci satisfasse presque la PC1.

Bibliographie

[1] A Schöbel. *Optimization in Public Transportation : Stop Location, Delay Management and Tariff Zone Design in a Public Transportation Network*. Springer, 2005.

[2] M R Bussieck, T Winter, and U Zimmermann. Discrete optimization in public rail transport. *Mathematical Programming*, 79 :415–444, 1997.

[3] M J Demetsky, M Asce, and B B Lin. Bus stop location and design. *Transportation Engineering Journal*, 108 :313–327, 1982.

[4] T C Matisziw, A T Murray, and C Kim. Strategic route extension in transit networks. *Discrete Optimization*, 71 :661–673, 2006.

[5] A Ceder and N H M Wilson. Bus network design. *Transportation Research B*, 20 : 331–344, 1986.

[6] M Estrada, M Roca-Riu, H Badia, F Robusté, and C F Daganzo. Design and implementation of efficient transit networks : Procedure, case study and validity test. *Transportation Research Part A*, 45 :935–950, 2011.

[7] D R P Gross, H W Hamacher, S Horn, and A Schöbel. *Stop Location Design in Public Transportation Networks : Covering and Accessibility Objectives*. Springer, 2008.

[8] M Patriksson and M Labbé, editors. *Transportation Planning. State of the Art*, 2002. Kluwer.

[9] J M Su and C H Chang. The multimodal trip planning system of intercity transportation in taiwan. *Expert Systems with Applications*, 37 :6850–6861, 2010.

[10] V Guihaire and J K Hao. Transit network design and scheduling : A global review. *Transportation Research Part A*, 42 :1251–1273, 2008.

[11] R Borndörfer, M Neumann, and M E Pfetsch. Fare planning for public transport. Technical Report 05-20, ZIB Berlin, 2005.

[12] A Murray, R Davis, R Stimson, and L Ferreira. Public transportation access. *Transportation Research*, 3 :319–328, 1998.

[13] D R Poetranto. Stop location problem in public transportation network. Master's thesis, Technische Universität Kaiserslautern, 2004.

[14] A Schöbel, H W Hamacher, A Liebers, and D Wagner. The continuous stop location problem in public transportation networks. *Asia-Pacific Journal of Operational Research*, 26 :13–30, 2009.

[15] J Gleason. A set covering approach to bus stop location. *Omega*, 3 :605–608, 1975.

[16] R Borndörfer, M Grötschel, and M E Pfetsch. Models for line planning in public transport. Technical Report 04-10, ZIP Berlin, 2004.

[17] H W Hamacher, A Liebers, A Schöbel, D Wagner, and F Wagner. Locating new stops in a railway network. *Electronic Notes in Theoretical Computer Science*, 50 : 1–11, 2001.

[18] A Ruf and A Schöbel. Set covering problems with almost consecutive ones property. *Discrete Optimization*, 15 :215–228, 2004.

[19] N Ruf. *Locating Train Stations : Set Covering Problems with Almost Consecutive Ones Property.* PhD thesis, Department of Mathematics, Universität Kaiserslautern, 2002.

[20] Z Drezner and W H Hamacher, editors. *Facility Location - Applications and Theory*, 2002. Springer-Verlag.

[21] A Schöbel and M Schröder. Covering population areas by railway stops. In *Proceedings of OR 2002, Klagenfurt.* Springer, 2003.

[22] S Mecke, A Schöbel, and D Wagner. Station location - complexity and approximation. In *5th Workshop on Algorithmic Methods and Models for Optimization of Railways*, 2005.

[23] R S Garfinkel and G L Nemhauser. *Integer Programming*. Wiley, 1972.

[24] M R Garey and D S Johnson. *Computers and Intractability, a Guide to the Theory of NP-Completeness*. Freeman and Co., 1979.

[25] M Gondran. Des algorithmes linéaires pour les problèmes de partition, de recouvrement et de couplage dans les hypergraphes d'intervalle. *Rairo Recherche Opérationnelle*, 13 :13–21, 1979.

[26] M Minoux. A class of combinatorial problems with polynomially solvable large scale set covering/partitioning relaxations. *RAIRO Recherche Opérationnelle*, 21 : 105–136, 1987.

[27] V Chvátal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4 :233–235, 1979.

[28] D S Hauchbaum. Approximation algorithms for the set covering and vertex cover problems. *SIAM Journal on Computing*, 11 :555–556, 1982.

[29] U Feige. A threshold of logn for approximating set cover. *Journal of the ACM*, 45 :634–652, 1998.

[30] P Slavik. A tight analysis of the greedy algorithm for set cover. *Journal of Algorithms*, 25 :237–254, 1997.

[31] V Lifschitz and B Pittel. The worst and the most probable performance of a class of set-covering algorithms. *SIAM Journal on Computing*, 12 :329–346, 1983.

[32] D S Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer Sytems Science*, 9 :256–278, 1974.

[33] L Lovasz. On the ratio of optimal and fractional covers. *Discrete Mathematics*, 13 :383–390, 1974.

[34] V V Vazirani. *Approximation Algorithms*. Springer, 2001.

[35] M Ball and A Roberts. A graph partitioning approach to airline crew scheduling. *Transportation Science*, 19 :107–126, 1985.

[36] J Cordeau, G Stojkovic, and F Soumis. Benders decomposition for simultaneous aircraft routing and crew scheduling. *Transportation Science*, 35 :1415–1429, 2001.

[37] M Desrochers and F Soumis. A column generation approach to the urban transit crew scheduling problem. *Transportation Science*, 23 :1–13, 1989.

[38] B Gopalakrishnan and E L Johnson. Airline crew scheduling : State of the art. *Annals of Operations Research*, 140 :305–319, 2005.

[39] E Marchiori and A Steenbeck. *An Evolutionnary Algorithm for Large Scale Set Covering Problem with Application to Airline Crew Scheduling*, pages 367–375. 2000.

[40] R E Marsten and F Shepardson. Exact solution of crew scheduling problems using set partitioning model : Recent successful applications. *Networks*, 11 :165–177, 1981.

[41] H D Sherali, E K Bish, and X Zhu. Airline fleet assignment concepts, models, and algorithms. *European Journal of Operational Research*, 172 :1–12, 2006.

[42] M G Sohoni, E L Johnson, and T G Bailey. Operational airline reserve crew planning. *Journal of Scheduling*, 9 :203–212, 2006.

[43] S Yan and C H Tseung. A passenger demand model for airline flight scheduling and fleet routing. *Computers and Operations Research*, 29 :1559–1571, 2002.

[44] F M Zeghal and M Minoux. Modeling and solving a crew assignment problem in air transportation. *Discrete Optimization*, 175 :187–209, 2006.

[45] J Current and M O'Kelly. Locating emergency warning sirens. *Decision Sciences*, 23 :221–234, 1992.

[46] R Z Farahani, N Asgari, N Heidari, M Hosseininia, and M Goh. Covering problems in facility location : A review. *Journal of Computers and Industrial Engineering*, 62 :368–407, 2011.

[47] C Torregas, R Swain, C Revelle, and L Bergman. The location of emergency service facilities. *Operations Research*, 19 :1363–1373, 1971.

[48] W Walker. Application of the set covering to the assignment of ladder trucks to fire houses. *Operations Research*, 22 :275–277, 1974.

[49] T G Crainic and J Roy. Une approche de recouvrement d'ensemble pour l'établissement d'horaires de chauffeurs dans le transport routier de charges

partielles. *Revue française d'automatique, d'informatique et de recherche opérationnelle*, 24 :123–158, 1990.

[50] P Toth and D Vigo, editors. *The Vehicle Routing Problem*, 2002. SIAM Monographs on Discrete Mathematics and Applications.

[51] R S Garfinkel and G L Nemhauser. Optimal political districting by implicit enumeration techniques. *Management Science*, 16 :B495–B508, 1970.

[52] M Minoux. Optimal traffic assignment in a SS/TDMA frame : A new approach by set covering and column generation. *RAIRO Recherche opérationnelle*, 20 : 273–286, 1986.

[53] E Boros, P L Hammer, T Ibaraki, and A Kogan. Logical analysis of numerical data. *Mathematical Programming*, 79 :163–190, 1997.

[54] J M Borneman, M Chrobak, G D Vedova, A Figueroa, and T Jiang. Probe selection algorithms with applications in the analysis of microbial community. *Bioinformatics*, 17 :S39–S48, 2001.

[55] K Darby-Dowman and G Mitra. An extension of set partitioning with application to scheduling problems. *European Journal of Operational Research*, 21 :200–205, 1985.

[56] F J Vasko, F E Wolf, and K L Stott. Optimal selection of ingot sizes via set covering. *Operations Research*, 35 :346–353, 1987.

[57] M Aourid and B Kaminska. Neural networks for the set covering problem : An application to the test vector compaction. pages 4645–4649. IEEE International Conference on Neural Networks, 1994.

[58] N Bansal, A Caprara, and M Sviridenko. A new approximation method for set covering problems, with applications to multidimensional bin packing. *SIAM Journal on Computing*, 39 :1256–1278, 2009.

[59] Z Blázsik and B Imreh. A note on connection between PNS and set covering problems. *Acta Cybernetica*, 12 :309–312, 1996.

[60] L Chen and J Crampton. Set covering problems in role-based access control. In *14th European Symposium on Research in Computer Security (ESORICS09)*, pages 689–704, 2009.

[61] P Hansen, M Labbé, and D Schindl. Set covering and packing formulations of graph coloring : Algorithms and first polyhedral results. *Discrete Optimization*, 6 :135–147, 2009.

[62] B Nepal, G Lassan, B Drow, and K Chelst. A set-covering model for optimizing selection of portfolio of microcontrollers in an automotive supplier company. *European Journal of Operational Research*, 193 :272–281, 2009.

[63] M Seda. Heuristic set-covering-based postprocessing for improving the quine-McCluskey method. *Wold Academy of Science, Engineering and Technology*, 29 : 256–260, 2007.

[64] A Caprara, P Toth, and M Fischetti. Algorithms for the set covering. *Annals of Operations Research*, 98 :353–371, 2000.

[65] S Ceria, P Nobili, and A Sassano. Set covering problem. In M Dell'Amico, F Mafioli, and P Toth, editors, *Annotated Bibliographies in Combinatorial Optimization*, pages 415–428, New York, 1997. John Wiley and Sons.

[66] N Christofides and S Korman. A computational survey of methods for the set covering problem. *Management Science*, 21 :591–599, 1975.

[67] F C Gomes, C N Meneses, P M Pardalos, and G V R Viana. Experimental analysis of approximation algorithms for the vertex cover and set covering problems. *Computers and Operations Research*, 33 :3520–3534, 2006.

[68] T Grossman and A Wool. Computational experience with approximation algorithms for the set covering problem. *European Journal Of Operational Research*, 101 :81–92, 1997.

[69] E Balas and M C Carrera. A dynamic subgradient-based branch-and-bound procedure for set covering. *European Journal of Operational Research*, 44 :875–890, 1996.

[70] E Balas and A Ho. Set covering algorithms using cutting planes, heuristics, and subgradient optimization : A computational study. *Mathematical Programming Study*, 12 :37–60, 1980.

[71] J E Beasley. An algorithm for set covering problems. *European Journal of Operational Research*, 31 :85–93, 1987.

[72] J E Beasley and J Jörnsten. Enhancing an algorithm for set covering problems. *European Journal of Operational Research*, 58 :293–300, 1992.

[73] M L Fisher and P Kedia. Optimal solution of set covering/ partitioning problems using dual heuristics. *Management Science*, 36 :674–688, 1990.

[74] S Haddadi. Une approche par partitionnement des contraintes du problème de couverture d'ensemble. *Revista Ibero-Latino Americana de Investigacion Operativa*, 5 :99–116, 1996.

[75] S Haddadi and A Benchettah. Un algorithme pour le problème de couverture d'Ensemble. *Maghreb Mathematical review*, 11 :66–81, 2002.

[76] C Mannino and A Sassano. Solving hard set covering problems. *Operations Research*, 18 :1–5, 1994.

[77] L K Hoffman and M Padberg. Solving airline crew scheduling problems by branch-and-cut. *Management Science*, 39 :657–682, 1993.

[78] D Y Yeh. A dynamic programming approach to the complete set partitionning problem. *BIT*, 26 :467–474, 1986.

[79] P Galinier and A Hertz. Solution techniques for the large set covering problem. *Discrete Applied Mathematics*, 155 :312 –326, 2007.

[80] M Afif, M Hifi, V T Paschos, and V Zissimopoulos. A new efficient heuristic for the minimum set covering problem. *Journal of the Operational Research Society*, 46 :1260–1268, 1995.

[81] L Jacobs and M Brusco. A local-search heuristic for large set-covering problems. *Naval Research Logistics*, 42 :1129–1140, 1995.

[82] N Musliu. Local search algorithm for unicost set covering problem. In *The 19th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems (IEA/AIES06)*, 2006.

[83] J E Beasley. A lagrangian heuristic for set-covering problems. *Naval Research Logistics*, 37 :151–164, 1990.

[84] A Caprara, M Fischetti, and P Toth. A heuristic method for the set covering problem. *Operations Research*, 47 :730–743, 1999.

[85] S Ceria, P Nobili, and A Sassano. A lagrangian-based heuristic for large-scale set covering problems. *Mathematical Programming*, 81 :215–228, 1998.

[86] S Haddadi. Simple lagrangian heuristic for the set covering problem. *European Journal of Operational Research*, 97 :200–204, 1997.

[87] M Alminana and J T Pastor. An adaptation of SH heuristic to the location set covering problem. *European Journal of Operational Research*, 100 :586–593, 1996.

[88] M J Brusco, L W Jacobs, and G M Thompson. A morphing procedure to supplement a simulated annealing heuristic for cost-and-coverage-correlated set-covering problems. *Annals of Operations Research*, 86 :611–627, 1999.

[89] K S Al-Sultan, M F Hussain, and J S Nizami. A genetic algorithm for the set covering problem. *Journal of the Operational Research Society*, 47 :702–709, 1996.

[90] J E Beasley and P C Chu. A genetic algorithm for the set covering problem. *European Journal of Operational Research*, 94 :392–404, 1996.

[91] M Solar, V Parada, and R Urrutia. A parallel genetic algorithm to solve the set covering problem. *Computers and Operations Research*, 29 :1221–1235, 2002.

[92] Z M Naji-Azimi, P Toth, and L Galli. An electromagnetism metaheuristic for the unicost set covering problem. *European Journal of Operational Research*, 205 : 290–300, 2010.

[93] J Bautista and J Pereira. A GRASP algorithm to solve the unicost set covering problem. *Computers and Operations Research*, 34 :3162–3173, 2007.

[94] B Crawford, R Soto, E Monfroy, F Paredes, and W Palma. A hybrid ant algorithm for the set covering problem. *International Journal of the Physical Sciences*, 6 : 4667–4673, 2011.

[95] T A Feo and M G C Resende. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8 :67–71, 1989.

[96] G Lan, G W DePuy, and G E Whitehouse. An effective and simple heuristic for the set covering problem. *European Journal Of Operational Research*, 176 :1387–1403, 2007.

[97] S R Balachandar and K Kannan. A metaheuristic algorithm for set covering problem based on gravity. *International Journal of Computational and Mathematical Sciences*, 1 :502–507, 2010.

[98] L Lessing, I Dumitrescu, and T Stützle. A comparison between ACO algorithms for the set covering problem. *Lecture Notes in Computer Science*, 3172 :1–12, 2004.

[99] M Ohlsson, C Peterson, and B Soderberg. An efficient mean field approach to the set covering problem. *European Journal Of Operational Research*, 133 :583–595, 2001.

[100] F J Vasko and F E Wolf. A heuristic concentration approach for weighted set covering problems. *ePublication of Location Analysis*, 2 :1–14, 2001.

[101] M Yagiura, M Kishida, and T Ibaraki. A 3-flip neighborhood local search for the set covering problem. *European Journal of Operational Research*, 172 :472–499, 2006.

[102] D Gouwanda and S G Ponnambalam. Evolutionary search techniques to solve set covering problems. *World Academy of Science, Engineering and Technology*, 39 : 20–25, 2008.

[103] S Sundar and A Singh. A hybrid heuristic for the set covering problem. *European Journal of Operational Research*, 1 :21–21, 2010.

[104] D Bertsimas and R Vohra. Rounding algorithms for covering problems. *Mathematical programming*, 80 :63–89, 1998.

[105] G Cornuejols and A Sassano. On the 0-1 facets of the set covering polytope. *Mathematical Programming*, 43 :45–56, 1989.

[106] P L Hammer and Jr D J Rader. Maximally disjoint solutions of the set covering problem. Technical Report MS TR 98-03, Department of Mathematics, Rose-Hulman Institute of Technology, Terre-Haute, 1998.

[107] S G Kolliopoulos and N E Young. Approximation algorithms for covering/packing integer programs. *Journal of Computer and System Sciences*, 71 :495–505, 2005.

[108] M Mouthuy, Y Deville, and G Dooms. Contrainte globale pour le problème de recouvrement d'ensemble. In *Journées Francophones de Programmation Par Contraintes (JFPC'07)*, pages 255–264, Rocquencourt, France, 2007.

[109] P Beraldi and A Ruszczynski. The probabilistic set covering problem. Technical Report RRR 8-99, RUTCOR (Rutgers Center for Operations Research), 1999.

[110] P Chen and G Ding. A greedy heuristic for a generalized set covering problem. Technical Report RRR 16-2008, RUTCOR (Rutgers Center for Operations Research), 2008.

[111] C I Chiang, M J Hwang, and Y H Liu. An alternative formulation for certain fuzzy set-covering problems. *Mathematical and Computer Modelling*, 42 :363–365, 2005.

[112] S Mecke and D Wagner. Solving geometric covering problems by data reduction. In *12th Annual European Symposium on Algorithms (ESA'04)*, pages 760–771. Springer, 2004.

[113] J Yang and J Y T Leung. A generalization of the weighted set covering problem. *Wiley Periodicals*, pages 142–149, 2003.

[114] K S Booth and G S Lueker. Testing for the consecutive ones property, interval graphs and planarity using PQ-tree algorithms. *Journal of Computing Systems Science*, 13 :335–379, 1976.

[115] B Korte and J Vygen. *Combinatorial Optimization – Theory and Algorithms*. Springer, 2008.

[116] A Schrijver. *Theory of Linear and Integer Programming*. John Wiley and Sons, 1986.

[117] L Kou. Polynomial complete consecutive information retrieval problems. *SIAM Journal on Computing*, 6 :67–75, 1977.

[118] S Haddadi. A note on the NP-hardness of the consecutive block minimization problem. *International Transactions in Operational Research*, 9 :775–778, 2002.

[119] S Haddadi and Z Layouni. Consecutive block minimization is 1.5-approximable. *Information Processing Letters*, 108 :132–135, 2008.

[120] E L Lawler, J K Lenstra, A H G Rinnooy Kan, and D B Shmoys, editors. *The Traveling Salesman Problem – a Guided Tour of Combinatorial Optimization*, 1985. John Wiley and Sons.

[121] N Christofides. Worst-case analysis of a new heuristic for the traveling salesman problem. Technical Report 388, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburg, 1976.

[122] J S Deogun and K Gopalakrishnan. Consecutive retrieval property – revisited. *Informations Processing Letters*, 69 :15–20, 1999.

[123] J J Bartholdi III, J B Orlin, and H D Ratliff. Cyclic scheduling via integer programs with circular ones. *Operations Research*, 28 :1074–1085, 1980.

[124] J E Atkins and M Middendorf. On physical mapping and the consecutive ones property for sparse matrices. *Discrete Applied Mathematics*, 71 :23–40, 1996.

[125] W F Lu and W L Hsu. A test for the consecutive ones property on noisy data – application to physical mapping and sequence assembly. *Journal of Computational Biology*, 10 :709–735, 2003.

[126] J T Atkins, E G Boman, and B Hendrickson. A spectral algorithm for seriation and the consecutive ones problem. *SIAM Journal on Computing*, 28 :297–310, 1998.

[127] D Baatar, N Boland, S Brand, and P J Stuckey. Minimum cardinality matrix decomposition into consecutive-ones matrices : CP and IP approaches. In P Van Hentenryck and L Wolsey, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, volume 4510 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2007.

[128] T M Christof, M Oswald, and G Reinelt. Consecutives ones and a betweenness problem in computational biology. Technical report, University of Heidelberg, 1998.

[129] S P Ghosh. File organization : The consecutive retrieval property. *Communications of the ACM*, 15 :802–808, 1972.

[130] A Jennings. A compact storage scheme for the solution of simultaneously equations. *Computer Journal*, 9 :281–285, 1966.

[131] D Johnson, S Krishnan, J Chhugani, S Kumar, and S Venkatasubramanian. Compressing large boolean matrices using reordering techniques. pages 13–23. Proceedings of the 30th international conference on very large databases, 2004.

[132] D G Kendall. Incidence matrices, interval graphs and seriation in archaeology. *Pacific Journal of Mathematics*, 28 :565–570, 1969.

[133] G Lancia, V Bafna, S Istrail, R Lippert, and R Schwartz. SNPs problems, complexity, and algorithms. In *Algorithms – ESA 2001*, volume 2161 of *Lecture Notes in Computer Science*, pages 182–193. Springer, 2001.

[134] F Bendali-Amor and A Quilliot. Compatibilité entre structures d'intervalles et relations d'ordre. pages 1–22, Montréal, 1989. Colloque J. Cartier.

[135] D R Fulkerson and O A Gross. Incidence matrices and interval graphs. *Pacific Journal of Mathematics*, 15 :835–855, 1965.

[136] A Tucker. A structure theorem for the consecutive ones property. *Journal of Combinatorial Theory*, 12 :153–162, 1972.

[137] N S Narayanaswamy and R Subashini. A new characterization of matrices with the consecutive ones property. *Discrete applied mathematics*, 157 :3721–3727, 2009.

[138] M Oswald and G Reinelt. Some relations between consecutive ones and betweenness polytopes. In P Chamoni, R Leisten, A Martin, J Minnemann, and H Stadler, editors, *Operations Research 2001 : Selected Papers of the International Conference on Operations Research 2001(OR'01)*, pages 277–283. Springer, 2002.

[139] M Dom. Algorithmic aspects of the consecutive-ones property. *Bulletin of the European Association for Theoretical Computer Science*, 98 :27–59, 2009.

[140] M Habib, R McConnell, C Paul, and L Viennot. Lex-BFS and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing. *Theoretical Computer Science*, 234 :59–84, 2000.

[141] W L Hsu. A simple test for the consecutive ones property. *Journal of Algorithms*, 43 :1–16, 2002.

[142] J Meidanis, O Porto, and G P Telles. On the consecutive ones property. *Discrete Applied Mathematics*, 88 :325–354, 1998.

[143] W L Hsu and R M McConnell. PC trees and circular-ones arrangements. *Theoretical Computer Science*, 293 :99–116, 2003.

[144] G P Telles and J Meidanis. Building PQR trees in almost-linear time. Technical Report IC-03-026, Instituto de Computação, UNiversidade Estadual de Campinas, 2003.

[145] M T Hajiaghayi and Y Ganjali. A note on the consecutive ones submatrix problem. *Information Processing Letters*, 83 :163–166, 2002.

[146] M Dom, J Guo, and R Niedermeier. Approximability and parametrized complexity of consecutive ones submatrix. In *Proc. 4th TAMC*, volume 4484 of *LNCS*, pages 680–691. Springer, 2007.

[147] M Veldhorst. Approximation of the consecutive ones matrix augmentation problem. *SIAM Journal on Computing*, 14 :709–729, 1985.

[148] R Wang and F C M Lau. Consecutive ones block for symmetric matrices. Technical Report CSIS TR-2003-09, Hong-Kong University, 2003.

[149] M Oswald. *Weighted Consecutive Ones Problems*. PhD thesis, Ruprecht-Karls-Universität, Heidelberg, 2003.

[150] M Oswald and G Reinelt. The simultaneous consecutive ones problem. *Theoretical Computer Science*, 410 :1986–1992, 2009.

[151] J F Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4 :238–252, 1962.

[152] A J Conejo, E Castillo, R Minguez, and R Garcia-Bertrand. *Decomposition Techniques in Mathematical Programming*. Springer, 2006.

[153] O E Flippo and A H G Rinnooy Kan. Decomposition in general mathematical programming. *Mathematical Programming*, 60 :361–382, 1993.

[154] P Mahey. Méthodes de décomposition et décentralisation en programmation linéaire. *RAIRO Recherche Opérationnelle*, 20 :287–306, 1986.

[155] S Binato, M V F Pereira, and S Granville. A new benders decomposition approach to solve power transmission network design problem. *IEEE Transactions on Power Systems*, 16 :235–240, 2001.

[156] A M Costa. A survey on benders decomposition applied to fixed-charge network design problems. *Computers and Operations Research*, 32 :1429–1450, 2005.

[157] T L Magnanti, P Mireault, and R T Wong. Tailoring benders decomposition for uncapacitated network design. *Mathematical Programming Study*, 26 :112–154, 1986.

[158] J Naoum-Sawaya. New benders' decomposition aproaches for w-CDMA telecommunication network design. Master's thesis, University of Waterloo, 2007.

[159] S Haddadi and O Slimani. Alternative decomposition based approaches for assigning disjunctive tasks. *Algorithmic Operations Research*, 2 :129–136, 2007.

[160] J F Cordeau, F Soumis, and J Desrosiers. A benders decomposition approach for the locomotive and car assignment problem. *Transportation Science*, 34 :133–149, 2000.

[161] A Mercier, J F Cordeau, and F Soumis. A computational study of benders decomposition for the integrated aircraft routing and crew scheduling problem. *Computers and Operations Research*, 32 :1451–1476, 2005.

[162] R Rouhani, L Lasdon, W Lebow, and A D Warren. A generalized benders decomposition approach to reactive source planning in power systems. *Mathematical Programming Study*, 25 :62–75, 1985.

[163] L S Lasdon. *Optimization Theory for Large Systems*. McMillan, 1970.

[164] J Beasley. Lagrangian relaxation. In C Reeves, editor, *Modern Heuristic Techniques for Combinatorial Optimization*. Wiley, 1993.

[165] M L Fisher. The lagrangian relaxation method for solving integer programming problems. *Management Science*, 27 :1–18, 1981.

[166] M L Fisher. An applications oriented guide to lagrangian relaxation. *Interfaces*, 15 :10–21, 1985.

[167] A M Geoffrion. Lagrangian relaxation for integer programming. *Mathematical Programming Study*, 2 :82–114, 1974.

[168] G L Nemhauser. The age of optimization : Solving large-scale real-world problems. *Operations Research*, 42 :5–14, 1994.

[169] M Held, P Wolfe, and H P Crowder. Validation of subgradient optimization. *Mathematical Programming*, 6 :62–88, 1974.

[170] S Haddadi. *Algèbre Linéaire Dans Rn : Théorie, Algorithmes et Complexité.* Hermes Science/Lavoisier, Paris, 2012.

[171] A Turing. On computable numbers with application to the entscheidungsproblem. In *Proc. London Math. Soc. Ser.*, pages 230–265 and 544–546, 1936.

[172] R G Jeroslow. There cannot be any algorithm for integer programming with quadratic constraints. *Operations Research*, 21 :221–224, 1972.

[173] J P Delahaye. *Logique, Informatique et Paradoxes.* Pour La Science, Paris, 1993.

[174] A S Fraenkel and D Lichtenstein. Computing a perfect strategy for nxn chess requires exponential time. *Journal of Combinatorial Theory*, A31 :199–214, 1981.

[175] A.R. Meyer and L.J. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential time. *13th annual conference on switching and automata theory*, pages 125–129, 1972.

[176] L J Stockmeyer and A K Chandra. Provably difficult combinatorial games. *SIAM Journal on Computing*, 8 :151–174, 1979.

[177] J Edmonds. Minimum partition of a matroïd in independent subsets. *Journal of Research of the National Bureau of Standards*, 69B :67–72, 1965.

[178] A A Cook. The complexity of theorem proving procedures. In *Proc. 3rd ACM Symposium on Theory of Computing*, pages 151–158, 1971.

[179] L J Stockmeyer. Classifying the computational complexity of problems. *Journal of Symbolic Logic*, 52 :1–43, 1987.

[180] R E Ladner. On the structure of polynomial time reducibility. *Journal of the ACM*, 22 :155–171, 1975.

[181] L J Stockmeyer. Computational complexity. In E. G. Coffman, J. K. Lenstra, and A. H. G. RinnooyKan, editors, *Computing*, volume 3 of *Handbooks in Operations Research and Management Science*, chapter 9, pages 455–513. Elsevier, 1992.

[182] J A M Schreuder. Application of a local model to fire stations in rotterdam. *European Journal of Operational Research*, 6 :212–219, 1981.

[183] M A Breuer. Simplification of the covering problem with application to boolean expressions. *Journal of the ACM*, 17 :166–181, 1970.

[184] F J Vasko, F E Wolf, and K L Stott. A set covering approach to metallurgical grade assignment. *European Journal of Operational Research*, 38 :27–34, 1989.

[185] F J Vasko, F E Wolf, K L Stott, and J M Scheirer. Selecting optimal ingot sizes for bethlehem steel. *Interfaces*, 19 :68–84, 1989.

[186] C C Ribeiro, M Minoux, and M C Penna. An optimal column-generation-with-ranking algorithm for very large-scale set partitioning problems in traffic assignment. *European Journal of Operational Research*, 41 :232–239, 1989.

[187] M Bellmore and H D Ratliff. Optimal defence of multi-commodity networks. *Management Science*, 18 :B174–B185, 1971.

[188] G G Brown, G W Graves, and D Ronen. Scheduling ocean transportation of crude oil. *Management Science*, 33 :333–346, 1987.

[189] F Shepardson and R E Marsten. A lagrangian relaxation algorithm for the two duty period scheduling problem. *Management Science*, 26 :274–281, 1980.

[190] E Balas and S M Ng. On the set covering polytope : I. all the facets with coefficients in 0,1,2. *Mathematical programming*, 43 :57–69, 1989.

[191] E Balas and S M Ng. On the set covering polytope : II. lifting the facets with coefficients in 0,1,2. *Mathematical programming*, 45 :1–20, 1989.

[192] A Sassano. On the facial structure of the set covering polytope. *Mathematical ptogramming*, 44 :181–202, 1989.

[193] U Aickelin. An indirect genetic algorithm for set covering problems. *Journal of the Operational Research Society*, 53 :1118–1126, 2001.

[194] S Umetani and M Yagiura. Relaxation heuristics for the set covering problem. *Journal of the Operations Research Society of Japan*, 50 :350–375, 2007.

[195] R Hassin and A Keinan. Greedy heuristics with regret, with application to the cheapest insertion algorithm for the TSP. *Operations Research Letters*, 36 :243–246, 2008.

[196] S Martello and P Toth. An algorithm for the generalized assignment problem. Amsterdam, North-Holland, 1981. Operational Research'81.

[197] J J Dongarra. Performance of various computers using standard linear equations software. Technical Report CS-89-85, University of manchester, 2013.

[198] K P Easwaran. Placement of records in a file and file allocation in a computer network. pages 304–307. IFIP Congress, 1974.

[199] M Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* PhD thesis, Friedrich-Schiller-Universität, Jena, Germany, 2008.

[200] J E Beasley. OR-library : Distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41 :1069–1072, 1990.

[201] V Chvátal. *Linear Programming.* W.H. Freeman and Co., 1983.

[202] T H Cormen, C E Leiserson, R L Rivest, and C Stein. *Introduction to Algorithms.* The MIT Press, 2009.

[203] M F Mammana, S Mecke, and D Wagner. The station location problem on two intersecting lines. *Electronic Notes in Theoretical Computer Science*, 92 :52–64, 2004.

[204] G L Nemhauser and L A Wolsey. *Integer and Combinatorial Optimization.* Wiley-Interscience, 1988.

[205] C H Papadimitriou. *Computational Complexity.* Prentice-Hall, 1994.

[206] J F Pierce. Application of combinatorial programming to a class of all-zero-one integer programming problems. *Management Science*, 15 :191–209, 1968.

[207] R Sedgewick and K Wayne. *Algorithms.* Addison-Wesley, 2011.

[208] E Horowitz and S Sahni. *Fundamentals of Computer Algorithms.* Computer Science Press Inc., 1978.

[209] G L Nemhauser and G M Weber. Optimal set partitionning, matching and lagrangian duality. *Naval Research Logistics Quarterly*, 26 :553–563, 1979.

[210] L A Wolsey. *Integer Programming.* Wiley, 1998.

[211] G Ausiello, P Crescenzi, G Gambosi, V Kann, A Marchetti-Spaccamela, and M Protasi. *Complexity and Approximation.* Springer, 1999.

[212] D S Johnson. A theoretician's guide to the experimental analysis of algorithms, 2001. http ://www. research.att.com/ dsj/.

[213] P Van Hentenrick and L Wolsey, editors. *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization*, 2007. Springer.

[214] M Jünger, T Liebling, D Naddef, G Nemhauser, W Pulleyblanck, G Reinelt, G Rinaldi, and L Wolsey, editors. *50 Years of Integer Programming 1958-2008*, 2010. Springer.

[215] A Dress, Y Xu, and B Zhu, editors. *Combinatorial Optimization and Applications*, 2007. Springer.

[216] R E Bixby, E A Boyd, and R Z Rios-Mercado, editors. *Integer Programming and Combinatorial Optimization*, 1998. Springer-Verlag.

[217] A A Assad. Models for rail transportation. *Transportation Research A*, 14 :205–220, 1980.

[218] P Brucker and J Hurink. A railway scheduling problem. *Zeitschrift für Operations Research*, 30 :223–227, 1986.

[219] M R Bussieck, P Kreuzer, and U T Zimmermann. Optimal lines for railway systems. *European Journal of Operational Research*, 96 :54–63, 1996.

[220] J Goossens, C P M VanHoesel, and L G Kroon. A branch and cut approach for solving line planning problems. *Transportation Science*, 38 :379–393, 2004.

[221] G Laporte, J A Mesa, and F A Ortega. Maximizing trip coverage in the location of a single rapid transit alignment. *Annals of Operations Research*, 136 :49–63, 2005.

[222] G Laporte, J A Mesa, and F A Ortega. Locating stations on rapid transit lines. *Computers and Operations Research*, 29 :741–759, 2002.

[223] D Pacciarelli, A Mascis, and M Pranzo. Scheduling models for shortterm railway traffic optimization. In *Proceedings of 9th Meeting on Computer-Aided Scheduling of Public Transport(CASPT 2004)*, 2004.

[224] B Gerards, editor. *Algorithmic Methods and Models for Optimization of Railways*, 2004. Electronic Notes in Theoretical Computer Science.

[225] D P Bovet and P Crescenzi. *Introduction to the Theory of Complexity*. Creative Commons, Free Electronic Edition, 2006.

[226] H R Lewis and C H Papadimitriou. *Elements of the Theory of Computation*. Prentice Hall, Upper Saddle River, 1981.

[227] N Hamidane. Un algorithme fondé sur une décomposition de benders pour le problème de couverture d'ensemble. Master's thesis, Université Badji Mokhtar, Annaba, 2001.

[228] C Barnhart, P Belobaba, and A R Odoni. Applications of operations research in the air transport industry. *Transportation Science*, 37 :368–380, 2003.

[229] C Barnhart and A Cohn. Airline schedule planning : Accomplishments and opportunities. *Manufacturing and Service Operations Management*, 6 :3–10, 2004.

[230] R Anbil, F Barahona, L Ladanyi, R Rushmeyer, and J Snowdown. Airline optimization. *OR/MS Today*, 26 :26–45, 1999.

[231] F Geraets, L Kroon, A Schöbel, D Wagner, and C D Zaroliagis, editors. *Algorithmic Methods for Railway Optimization*, 2007. International Dagstuhl Workshop, ATMOS 2004, Springer.

[232] R W Hall, editor. *Handbook of Transportation Science*, International Series in Operations Research and Management Science, 2003. Kluwer Academic Pub.

[233] R Borndörfer, M Grötschel, and M E Pfetsch. A path-based model for line planning in public transport. Technical Report 05-18, ZIP Berlin, 2005.

[234] M Gatto, R Jacob, L Peeters, B Weber, and P Widmayer. Theory on the tracks : A selection of railway optimization problems. *Bulletin of EATCS*, 84 :41–70, 2004.

[235] M A Odijk. A constraint generation algorithm for the construction of periodic railway timetables. *Transportation Science*, 30B :455–464, 1996.

[236] A Schöbel. Locating stops along bus or railway lines – a bicritetial problem. *Annals of Operations Research*, 136 :211–227, 2005.

[237] D Wagner. *Algorithms and Models for Railway Optimization*, volume 2748 of *Lecture Notes in Computer Science*. Springer, 2003.

[238] S Haddadi and S Chenche. Simple heuristic for the set covering problem. In *International Conference on Metaheuristics and Nature Inspired Computing (META'12)*, Port El-Kantaoui, Tunisia, 28-31 octobre, 2012.

[239] E K Baker, L D Bodin, W F Finegan, and R J Ponder. Efficient heuristics solutions to an airline crew scheduling problem. *AIIE Transactions*, 11 :79–85, 1979.

[240] K Hoffman and M Padberg. Set covering, packing and partitioning problems. *Encyclopedia of Optimization*, pages 3482–3486, 2009.

[241] S Haddadi, S Chenche, M Cheraitia, and F Guessoum. Polynomial-time local improvement heuristic for consecutive block minimization. In *International Conference on Metaheuristics and Nature Inspired Computing (META'14)*, Marrakech 27-31 octobre, 2014.

[242] S Haddadi, S Chenche, M Cheraitia, and F Guessoum. Polynomial-time local improvement algorithm for consecutive block minimization. *Information Processing Letters*, 2015. http ://dx.doi.org/10.1016/j.ipl.2015.02.010.

[243] S Haddadi and S Chenche. Benders decomposition for the set covering problem. Submitted.