ACADÉMIE D'AIX-MARSEILLE
UNIVERSITÉ D'AVIGNON ET DES PAYS DE VAUCLUSE

THÈSE

Techniques Hybrides de Propagation de Contraintes et de Programmation Mathématique.

SPÉCIALITÉ : INFORMATIQUE

par
Cristian Oliva

Composition du jury :

M	Philippe Michelon	Université d'Avignon	Directeur de thèse
M	Christian Artigues	Université d'Avignon	Co-Directeur de thèse
M	Jacques Ferland	Université de Montréal	Rapporteur
M	Victor Parada	Universidad de Santiago de Chile	Rapporteur
M	Michel Vasquez	Ecole des Mines d'Alès	Examinateur
M	Thierry Defaix	Centre Electronique de l'ARmement	Examinateur

---

Laboratoire Informatique d'Avignon

Remerciements

Table des matières

Table des figures

Liste des tableaux

Liste d'Algorithmes

Chapitre 1

Introduction Générale

Dans ce chapitre, nous introduisons brièvement l'Optimisation Combinatoire, la Programmation Par Contraintes et la Programmation Mixte en Nombres Entiers, dénommés respectivement OC, PPC et PMNE. Nous revenons, plus particulièrement, sur les concepts les plus utilisés dans la programmation linéaire et la programmation par contraintes à savoir la relaxation, l'inférence et la recherche. Nous poursuivons avec une revue de la littérature concernant les travaux liés, coopération programmation par contraintes et programmation linéaire, à cette thèse. Enfin, nous décrivons nos contributions.

1.1 Introduction

Les secteurs tels que les finances, les télécommunications et l'industrie sont confrontés, chaque jour, à des problèmes de décision. Ceux-ci peuvent être vus comme des problèmes de réalisabilité ou d'optimisation, dans lesquels la question est d'affecter des valeurs à différentes variables représentant un certain choix, comme une fréquence, une quantité ou un temps. Dans les finances, par exemple, nous voulons choisir des types de projets de manière à maximiser la rentabilité du portefeuille (gestion de portefeuille) ; dans les télécommunications, il peut s'agir d'affecter des fréquences à des trajets de façon à minimiser le niveau d'interférence dans les communications (affectation de fréquences) ; dans l'industrie, nous voulons savoir quelles machines utiliser pour exécuter un ensemble de tâches et à quelles dates ces tâches doivent être planifiées (projets à moyens limités) pour minimiser la durée totale du projet.

Dans l'étude de ces problèmes d'optimisation, l'optimisation combinatoire [86] traite les cas où les choix sont discrets. Un exemple de choix, auquel les entreprises sont souvent confrontées, est la sélection entre un projet A ou B pour mener à bien une idée.

Deux communautés s'occupent de la résolution de ces problèmes d'optimisation combinatoire : la communauté "Recherche Opérationnelle" (RO) et la communauté "Programmation Par Contraintes" (PPC). Dans la communauté "Recherche Opérationnelle", la résolution de problèmes d'optimisation combinatoire provient d'avances réalisées en programmation mathématique et est souvent basée sur la programmation linéaire (PL) [22]. Ces développements ont eu lieu dans la deuxième moitié du 20ème siècle, et ont permis de définir le champ connu sous le nom de programmation mixte en nombres entiers (PMNE)[1] et programmation linéaire en nombres entiers (PLNE)[2], (voir par exemple [126] et [86]). Un problème linéaire en nombres entiers est un cas spécial de problème mixte en nombres entiers dans lequel aucune variable n'est continue.

Par ailleurs, l'émergence de la Programmation Par Contraintes pour résoudre les problèmes d'optimisation date de la fin des années 80 [80] et vient de la découverte que la programmation logique peut se généraliser à des structures mathématiques arbitraires dotées d'un langage de contraintes représentant "le domaine du discours", comme par exemple, les domaines finis. Dans cette approche le processus de résolution est basé sur la génération de contraintes et la résolution de celles-ci à travers des méthodes spécifiques. Ces techniques sont étudiées dans le domaine de l'Intelligence Artificielle. Une des ces techniques, parmi les plus importantes, est la ***propagation de contraintes***. L'objectif

[1] En anglais Mixed-Integer Programming (MIP).
[2] En anglais Integer Programming (IP).

essentiel de la propagation de contraintes est de définir un problème équivalent au problème de départ, mais dont la nouvelle formulation présente un espace de recherche moins volumineux. La propagation de contraintes a pour but d'éliminer les valeurs des domaines des variables ne conduisant à aucune solution. Une caractéristique importante de la PPC est qu'elle considère généralement une seule contrainte à la fois : les algorithmes de filtrage sont exécutés successivement sur chaque contrainte et les informations obtenues sont propagées localement aux autres contraintes.

Il faut mentionner que la Programmation Par Contraintes résout des problèmes de Satisfaction de Contraintes (PSC), il ne s'agit pas de trouver une solution optimale mais plutôt une solution satisfaisant un ensemble de contraintes. Cependant, la PPC peut aussi être utilisée pour résoudre des problèmes d'optimisation. Il suffit d'ajouter une contrainte stipulant que la fonction objectif doit être plus grande (pour un problème de maximisation) que la valeur d'une solution précédemment trouvée.

Les principales avantages de la PPC sont la flexibilité et la lisibilité des formulations des problèmes. Ceci implique une mise en œuvre beaucoup plus facile et une maintenance plus simple. En revanche, une des principales faiblesses de la PPC est la difficulté à prévoir ses performances sur un problème d'optimisation donné.

Une troisième technique utilisée tant en RO qu'en PPC est la recherche locale (RL)[3] [4]. Cette technique est basée sur une exploration heuristique de l'espace de recherche par l'étude de solutions voisines de la solution courante. Cette classe d'algorithmes a été appliquée avec succès sur beaucoup de problèmes d'optimisation combinatoire. Cette n'assure cependant pas de trouver la meilleure solution ni surtout d'en prouver l'optimalité.

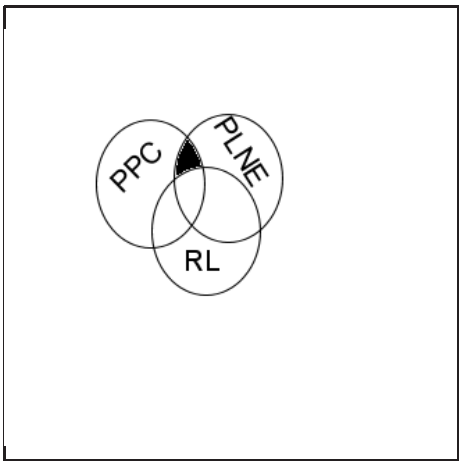

FIG. 1.1 – Combinaison d'approches de résolution de problèmes d'optimisation combinatoire.

[3]Local Search en anglais

La figure 1.1 schématise ces trois techniques et leurs intersections. À l'intersection noire, se situe le cadre de cette thèse : l'hybridation de la PPC et de la PLNE.

1.2 Concepts liés aux algorithmes d'optimisation combinatoire

Nous commençons par la description des concepts fondamentaux associés aux algorithmes d'optimisation combinatoire et nous les discutons par rapport à la PPC et à la PLNE. Ces concepts sont : la relaxation, la recherche et l'inférence [95].

1.2.1 La relaxation

Un problème de maximisation, par exemple, $P : \max\{f(x), x \in S_1\}$ est une relaxation du problème $Q : \max\{f(x)|x \in S_2\}$ si $S_1 \supseteq S_2$. En quelques mots, une relaxation d'un problème est une formulation qui contient un ensemble de solutions réalisables plus grand que celui du problème original. Dans ce cas de maximisation, le problème P donne une borne supérieure sur le problème Q. Dans le cas d'un problème de minimisation, la relaxation fournit une borne inférieure. Une bonne relaxation a une solution optimale proche de l'optimum du problème original. La relaxation la plus connue est la relaxation continue qui consiste à éliminer les contraintes d'intégralité. Le succès de cette relaxation vient du fait que les codes de programmation linéaire sont très efficaces et de plus, c'est une relaxation qui est systématiquement disponible pour autant que le problème initial possède une formulation linéaire.

De manière générale, la relaxation devrait satisfaire deux critères :

1. être plus facile à résoudre que le problème original, et
2. posséder une structure semblable à celle du problème original, de façon à ce qu'elle fournisse de bonnes bornes.

1.2.2 La recherche

La recherche est le processus d'exploration systématique d'un espace de solutions réalisables. Dans le contexte de l'optimisation, l'objectif de la recherche est d'affecter des valeurs aux variables de manière à satisfaire les contraintes associées au problème. Un algorithme de recherche est complet si, dans le cas où il existe au moins une solution réalisable, celui-ci prouve que la solution trouvée est optimale ou, dans le cas contraire, prouve

qu'il n'existe aucune solution qui satisfasse toutes les contraintes.

En l'optimisation combinatoire, la plupart des algorithmes complets sont basés sur des arbres de recherche, c'est-à-dire, une représentation de l'espace de recherche par un arbre où les nœuds internes sont les sous-espaces, les branches, la partition de l'espace et les feuilles, des affectations complètes des valeurs aux variables (solutions complètes) ou encore des solutions partielles ne pouvant mener à de meilleures solutions que la meilleure solution connue.

À la fois en PPC et RO, les arbres de recherche sont utilisés pour la résolution exacte de problèmes d'optimisation combinatoire mais de manière légèrement différente. En PPC, le branchement est combiné avec l'inférence de façon à réduire le nombre de choix possibles. Il existe plusieurs stratégies de branchement, comme par exemple choisir la variable qui a le plus petit domaine et lui affecter la plus petite valeur de son domaine, ou encore, lui affecter la valeur qui minimise le nombre de conflits avec les autres variables. Par contre, en RO, le branchement est effectué en fonction de la relaxation. Le but est d'éviter l'exploration de nœuds pour lesquels la relaxation n'est pas réalisable ou bien plus mauvaise que la meilleure solution connue jusqu'à présent. Il existe plusieurs stratégies d'exploration de l'arbre de recherche. Nous pouvons citer, entre autres, le "meilleur d'abord" où le prochain nœud à explorer est celui dont la valeur relaxée est la plus prometteuse.

1.2.3 L'inférence

Le but d'une méthode d'inférence est de réduire l'espace de recherche en essayant de dériver des implications à partir d'un ensemble de contraintes. En général, l'inférence renforce le problème original en ajoutant des contraintes valides.

En PPC, l'inférence se centre sur la notion de ***consistance***. La consistance est liée à la notion de réalisabilité d'une solution partielle. Une solution partielle est réalisable si et seulement si elle satisfait l'ensemble de contraintes du problème. Par exemple, les algorithmes de propagation de contraintes se développent sous cette notion de consistance en éliminant des domaines des variables les valeurs qui ne sont pas consistantes.

En RO, l'inférence joue un rôle tout aussi important. Par exemple, le prétraitement est une méthode d'inférence où on essaye d'éliminer des variables, en les fixant, dans le problème original [109]. D'autres techniques de programmation linéaire peuvent être vues comme des méthodes d'inférence, parmi-elles : les méthodes de décomposition (Benders, décomposition lagrangienne, Dantzig-Wolfe), les méthodes de renforcement de contraintes (lifting), les méthodes de coupes, etc.

Les méthodes d'inférence sont donc assez différentes selon qu'il s'agisse de PPC ou de RO.

1.3 Différences et similitudes entre la PPC et la PLNE

Étant donné que les champs de la PPC et de la RO ont été développés séparément, il est judicieux et nécessaire d'analyser les similitudes et différences entre ces deux champs d'étude. Cela peut aider à établir de bonnes méthodes de coopération entre ces deux approches.

Différence I : La Relaxation La notion de relaxation telle qu'elle existe en PLNE n'est pas la même en PPC. D'abord, en PPC la relaxation est sous-jacente au maintien de la consistance et nous pouvons la voir comme l'agrandissement du domaine au produit cartésien des domaines des variables. D'autre part, la fonction objectif est ramenée à une contrainte ordinaire ne servant qu'à éliminer des solutions moins bonnes et non, comme en RO, comme une mesure de la qualité de la solution relâchée (borne supérieure dans le cas d'un problème de maximisation). On espère, en RO, que cette solution relâchée satisfasse l'ensemble de contraintes du modèle original. Si la solution n'est pas réalisable alors une technique de séparation et évaluation est appliquée.

Différence II : Contraintes Redondantes Une des différences entre la PPC et PLNE est le traitement des contraintes redondantes. En effet, pour la communauté de PLNE les contraintes redondantes sont complètement ignorées et se font éliminer du problème en utilisant de techniques de réduction de contraintes. Au contraire, en PPC, elles servent à améliorer la propagation de l'information au travers des autres contraintes du problème [17]. Elles ont alors un rôle important.

Différence III : Fonction Objectif Une autre différence est l'utilisation de la fonction objectif. En effet, en RO, la fonction objectif guide la recherche vers la meilleure solution. Au contraire, en PPC, l'orientation de la recherche vers de "bonnes" solutions se fait par le biais d'heuristiques de branchement. Bien entendu, ces heuristiques peuvent être inspirées par la fonction objectif, mais cela reste des règles heuristiques. Ainsi, la fonction objectif, en PPC voit son rôle réduit à celui d'une contrainte ordinaire.

Différence IV : Modélisation La modélisation d'un problème en PPC est plus naturelle, lisible et flexible que la modélisation en terme de PLNE. De plus, les modèles contiennent souvent moins de variables.

Différence V : Type d'optimisation La PPC utilise un principe plus local dans le sens où chaque contrainte du modèle est regardée individuellement. Par contre, en RO, l'optimisation est faite de manière globale, dans le sens où les contraintes dans leur ensemble sont considérées en même temps pour obtenir une solution.

Du fait de ces différences, les approches PPC et PL sont clairement complémentaires : pourquoi ne pas utiliser un double raisonnement sur les contraintes pour la résolution d'une instance ? Pourquoi ne pas donner les deux rôles à la fonction objectif dans le même algorithme ?

Avec cette complémentarité, la coopération PPC/RO peut aussi être facilitée par les similitudes de ces deux techniques.

Similitude I : Stratégie de Branchement Le principe de branchement est commun aux deux approches et plusieurs stratégies de branchement sont employées à la fois en PPC et PLNE. En particulier, en PLNE, une stratégie consiste à sélectionner une variable fractionnelle selon la dégradation espérée de la valeur de la relaxation du problème. Cette stratégie se rapproche, en PPC, de la stratégie *Max-Regret* qui consiste à choisir la prochaine instantiation de façon à minimiser un risque estimé [17]. Les coûts réduits peuvent être utilisés pour estimer le "regret" des variables dans une approche PPC/PL, voir par exemple [31].

Similitude II : Prétraitement et probing versus disjonction constructive La disjonction constructive est une technique de PPC dans laquelle les informations communes à des disjonctions sont liftées et posées en tant que contraintes. Ceci permet, d'une part, d'améliorer la propagation de l'information et, d'autre part, d'élaguer l'espace de recherche [127]. En PLNE, l'association des techniques de prétraitement et de probing sont comparables à cette procédure [109].

Similitude III : Coupes versus contraintes redondantes Les coupes en PLNE servent à réduire l'espace de recherche en renforçant la relaxation du problème. Les contraintes redondantes accomplissent le "même rôle", en renforçant la propagation de l'information de façon à réduire l'espace de recherche.

Similitude IV : Nogoods versus décomposition de Benders L'apprentissage [54] est une technique de la communauté PPC. Elle est basée sur la découverte de combinaisons de solutions ou branches irréalisables pendant la recherche et elle est enregistrée et utilisée, sous la forme d'une nouvelle contrainte "nogood", pour éviter d'examiner ultérieurement des sous-espaces similaires. Une idée similaire est utilisée en RO, avec la décomposition de Benders [86]. Cette décomposition est utilisée pour résoudre des problèmes complexes comme les problèmes mixtes de grande taille. La décomposition de Benders est une méthode de partition des variables : à chaque itération, un problème maître est résolu pour fixer la valeur d'un sous-ensemble des variables puis, un sous-problème complète cette affectation et produit une "coupe de Benders", correspondant aux "nogoods" en PPC, et qui est ajoutée au problème maître. Cette coupe est obtenue par la résolution du dual du sous-problème et assure la convergence de l'algorithme. De manière générale, une coupe en PLNE est déduite d'une solution irréalisable et peut s'afficher comme l'équivalent PLNE des "nogoods" de la PPC.

En résumé, nous pouvons dire qu'il existe une similitude quant au principe de base de résolution des problèmes en PPC et en PLNE. Par exemple, les deux techniques utilisent la recherche arborescente et la réduction de domaines mais les algorithmes et modes de résolution sont différentes. Nous pensons qu'une approche peut fournir de bons résultats en exploitant la complémentarité existant entre les deux approches.

1.4 Quelques définitions

Avant de poursuivre avec une revue de la littérature concernant les travaux liés à la combinaisons des approches PPC/PL, nous introduisons quelques définitions préliminaires.

Contrainte Globale Une faiblesse de la propagation de contraintes est la suivante : chaque contrainte est examinée indépendamment des autres. Parfois, la connaissance des autres contraintes peut améliorer substantiellement la réduction du domaine. C'est pour cette raison qu'ont été introduites les *contraintes globales*. Une contrainte globale est un sous-ensemble de contraintes, correspondant à une sous-structure du problème original, et auquel est associé un algorithme de filtrage spécialisé. Parmi ces contraintes globales, nous allons en expliciter trois : "alldifferent", "cumulative" et "element".

- $alldifferent(\{x_1, \ldots, x_n\})$ C'est une contrainte globale où chacune des variables $x_1, \ldots, x_n$ doit prendre une valeur différente, c'est-à-dire $x_1 \neq x_2 \neq \ldots \neq x_n$. L'algorithme de filtrage associé à cette contrainte globale repose sur le "maximal bipartite matching problem" [106].

- *cumulative*$([S_1, \ldots, S_m], [D_1, \ldots, D_m], [R_1, \ldots, R_m], L)$ Cette contrainte apparaît dans les problèmes d'ordonnancement et de placement. Elle sert à modéliser la situation suivante : On doit ordonnancer m tâches, $S_1, \ldots, S_m$ sont les dates de début associées à chaque tâche. La durée d'exécution de chaque tâche i est D_i. Chaque tâche i demande R_i unités de ressource. L unités de ressource sont disponibles à tout moment. Les algorithmes de filtrage pour la contrainte cumulative sont très complexes et variés. Pour plus d'information, nous renvoyons le lecteur aux ouvrages de Baptiste et al. [12] et Hooker [54].
- *element*$(Y, V_1, \ldots, V_k, X)$ La contrainte "element" simule le comportement d'un tableau. Y est une variable dont le domaine est $\{1, \ldots, k\}$, c'est-à-dire les indices du tableau et $\{V_1, \ldots, V_k\}$ est une liste de valeurs. La contrainte dit que la variable X doit prendre la Y-ème valeur du tableau, c'est-à-dire $X = V[Y]$.

Contrainte Globale Mixte C'est une contrainte globale où les variables peuvent prendre des valeurs continues.

Contrainte d'Optimisation Globale Une contrainte d'optimisation globale est une contrainte globale qui en plus d'un algorithme de filtrage, contient un algorithme d'optimisation. Ces deux algorithmes permettent la réduction de l'espace de recherche en supprimant les valeurs qui ne peuvent pas donner lieu à des meilleures solutions que la meilleure solution connue.

1.5 Travaux liés à cette thèse

Dans cette section, nous relatons l'étude de l'état de l'art des travaux liés à cette thèse. Nous nous consacrons, d'abord, à quelques études de comparaisons entre la PPC et la PLNE. Ensuite, nous poursuivons avec des approches hybrides PPC/PL.

1.5.1 Comparaison entre la PPC et la PLNE

Il existe, dans la littérature, plusieurs articles consacrés à la comparaison des deux approches PPC et PLNE.

Un des premiers, est celui de Darby-Dowman et al. [23] où sont étudiés plusieurs problèmes industriels de type affectation généralisée. Les auteurs proposent un modèle PPC en utilisant des variables de type $x_i \in D_i$, des contraintes "element" et un modèle

de PLNE en utilisant des variables du type $x_{ij} \in \{0,1\}$. Le nombre de variables du modèle PPC est plus petit que celui du modèle PLNE. L'article fait une description assez détaillée des différences entre les deux formulations. Pour cet ensemble de problèmes, la PPC s'est avérée beaucoup plus performante que la PLNE en temps CPU et en robustesse. Il faut mentionner que le solveur de PLNE atteint une solution entière optimale mais n'est pas capable de prouver l'optimalité. Finalement, l'article termine avec une discussion sur l'intégration de la PPC à l'intérieur du solveur de PLNE.

Le célèbre "progressive party problem" a été étudié par Smith et al. [113]. L'article présente deux modèles de PLNE et l'un d'entre eux est comparé à celui de la PPC. La formulation de PLNE présente deux désavantages : le modèle est volumineux en nombre de variables et la relaxation continue est très mauvaise. En comparaison, la formulation PPC est beaucoup plus compacte et les variables sont en relation beaucoup plus directe avec les décisions du problème original. Ceci permet de concevoir de meilleures stratégies de recherche et c'est pour ces raisons que la PPC est beaucoup plus performante que l'approche de PLNE. Cependant, il faut noter que le problème original est évalué grâce à la PLNE. Cette évaluation (borne inférieure) permet de déduire un sous-problème de réalisabilité. C'est sur ce sous-problème que, en réalité, la PPC est plus performante que la PLNE. Bien qu'une solution soit trouvée rapidement pour une valeur de la fonction objectif égale à 13 (avec l'aide des fixations arbitraires de variables), la PPC a beaucoup de difficultés pour prouver que cette solution est optimale. D'autre part, le simplexe prouve facilement l'irréalisabilité pour la valeur 12. Ceci illustre bien la complémentarité des deux approches. D'autres chercheurs ont étudié ce problème, parmi-eux : Hooker et Osorio [56], Kay [104], Kalvelagen [66] et Galinier et Hao [39].

1.5.2 Approches hybrides PPC/PL

Dans la suite nous entreprenons une brève revue sur les principaux apports dans le domaine de la coopération entre la PPC et la PLNE.

Une première tentative de coopération PPC/PL sous un même solveur est proposée par Hooker et Osorio [56]. Les auteurs proposent une extension du PMNE par ce qu'ils appellent "Mixed Logical/Linear Programming" (MLLP). La MLLP est une approche générale pour la modélisation et résolution des problèmes d'optimisation en variables entières et continues. Elle étend la PMNE en introduisant la modélisation logique et des stratégies de solution. La MLLP ne rejette la PLNE en aucune manière. Au contraire, elle incorpore les techniques associées à la programmation en nombres entiers. L'article discute les algorithmes de relaxation et de traitement logique (propagation de contraintes), et pré-

sente ensuite des résultats expérimentaux pour quatre ensembles de problèmes. Les auteurs concluent, d'après ces résultats, que certaines contraintes doivent s'exprimer de façon logique et que d'autres contraintes, par exemple les contraintes de type sac-à-dos, doivent s'exprimer par des inégalités 0-1. Enfin, l'approche de modélisation proposée peut permettre des formulations plus compactes sans sacrifier l'efficacité. Le plus grand répertoire des techniques de la PMNE peut accélérer la résolution ou même résoudre des problèmes qui sont intraitables par la PLNE. Ces techniques incluent des stratégies de branchement, des relaxations et des algorithmes de traitement logique qui ne sont pas, normalement, associés à la programmation en nombres entiers.

La relation entre la modélisation et les techniques de résolution (solveurs) a été étudiée par Hooker et al. [55], dans le but d'identifier comment le langage de modélisation peut aider à intégrer différents solveurs. La principale contribution réside dans les conditions générales, données par les auteurs, pour la complétude de la recherche lorsque cohabitent différents solveurs. Une intégration entre deux solveurs est proposée par Rodošek et al. [107]. L'article présente une intégration de la PMNE et de la PPC. Le système intégré fusionne les composantes du système PPC d'ECLiPSe et les composantes du système PMNE de CPLEX. Les contraintes sont gérées par les deux composantes. L'algorithme proposé réduit l'espace de recherche du problème progressivement en appelant la propagation de contraintes d'ECLiPSe et le simplexe dual de CPLEX. Les avantages de cette hybridation sont illustrées en résolvant les problèmes suivants : "2-Hoist Scheduling Problem" (2HSP), "Progressive Party Problem" (PPP), "Cabinet Assignment Problem" (CAP) et "Set Partitioning Problem" (SPP). Les résultats montrent que le problème CAP est facilement résolu par la PPC et difficilement résolu par la PMNE, tandis que le problème SPP est difficilement résolu par la PPC et facilement résolu par la PMNE. Les problèmes 2HSP et PPP sont difficiles à résoudre par la PPC et par la PMNE. Cependant, les résultats expérimentaux montrent que l'algorithme hybride obtient la solution optimale et prouve son optimalité pour chaque problème, ce qui est impossible par la PPC et la PMNE séparément. Hooker et al. [58] étudient un chemin pour la coopération des techniques de la PPC et de la PLNE en identifiant, les principes sous-jacents des deux approches (recherche, inférence, renforcement, relaxation) et en clarifiant leurs relations à partir de dualités basiques : recherche/inférence et renforcement/relaxation. L'article suggère, puisque la PPC et la PLNE utilisent des stratégies basées sur les mêmes dualités, leurs méthodes peuvent être combinées facilement. Pour illustrer ceci, les auteurs présentent des règles de réduction de domaines basées sur une relaxation linéaire pour des variables indexées (la contrainte "element"). Un autre article sur l'intégration PPC/PL est celui de Ottoson et al. [57]. Cet article se focalise sur la propagation de contraintes comme technique d'inférence, la programmation linéaire comme technique d'optimisation, et comment ces techniques peuvent

être combinées. Le lien proposé pour l'intégration entre l'inférence et l'optimisation est une contrainte conditionnelle. Cette structure conditionnelle est illustrée sur un problème d'ordonnancement multi-machine. Plus précisément, la communication PPC et PLNE est construite autour des contraintes conditionnelles et les techniques de résolution sont basées sur une procédure par séparation et évaluation. D'autre part, les auteurs présentent une relaxation linéaire pour des variables indexées continues, ainsi que l'intégration de "nogoods" obtenus à partir d'un sous-problème de programmation linéaire non-réalisable. Il faut mentionner qu'aucun test numérique n'est présenté.

Une des principales contributions de coopération entre la PPC et la PLNE repose sur "les contraintes globales" (voir section 1.4). Plusieurs chercheurs ont étudié ce sujet, parmieux : Ottosson et al.[97] et Focacci [31]. Ottosson et al. emploient des contraintes globales mixtes comme lien entre la PPC et la PL. Les auteurs présentent un schéma d'inférence bidirectionnel et de nouvelles stratégies de recherche pour les solveurs hybrides. Le processus d'inférence, appelé "back-propagation", identifie des valeurs pour les variables discrètes qui satisfont la solution de PL. Ceci facilite les choix de branchement. Une variable discrète pour laquelle la back-propagation ne peut pas trouver de valeur satisfaisante correspond à une inconsistance entre la solution de PL et les contraintes discrètes et indique une façon de séparer l'espace de recherche sur cette inconsistance. Ceci est similaire à la séparation effectuée par la PLNE sur les variables à valeurs fractionnelles. Les auteurs présentent des tests numériques pour le problème de planification de la production où le modèle hybride est constitué, d'une part par une fonction linéaire par morceaux et, d'autre part par des contraintes de type "element". Ce modèle hybride s'avère plus performant que celui du modèle PLNE. Focacci [31] propose une intégration de méthodes de RO au sein de ce qu'il appelle *contraintes d'optimisation globale*. Une contrainte d'optimisation globale est constituée, d'un ensemble de contraintes qui capture une sous-structure du problème original et d'un algorithme d'optimisation et de filtrage qui permet la réduction de l'espace de recherche en éliminant les valeurs qui ne peuvent pas donner à lieu à de meilleures solutions que la solution courante. La composante d'optimisation calcule une borne supérieure (si le problème est de maximisation) et une fonction gradient (coûts réduits pour la programmation linéaire) qui aide à obtenir plus d'information sur le problème. Finalement, l'auteur propose des solveurs pour cinq types de problèmes d'optimisation combinatoire et une étude numérique montre les avantages de cette approche par rapport à des approches de PPC pure. Les problèmes cibles sont : le problème du voyageur de commerce, le problème du voyageur de commerce avec fenêtres de temps, le problème d'ordonnancement avec temps de "setup" et ressources alternatives, le problème de planification d'horaires et le problème de minimisation de la somme des retards. Pour la résolution de ces problèmes, Focacci propose quatre types de contraintes d'optimisation globales, toutes basées sur une

fonction de gradient. Une autre étude se base sur l'utilisation des coûts réduits : Ottosson et Thorsteinsson [96]. Cet article présente et compare trois modèles pour le problèmes de configuration, un modèle PPC, un modèle PLNE et un autre hybride (PPC/PL). Pour ce dernier, les auteurs montrent comment une variante continue de la contrainte "element" peut être linéarisée et comment les coûts réduits obtenus, par programmation linéaire, peuvent être utilisés pour réduire les domaines des variables. Les tests comparatifs pour ces trois modèles ont été faits sur des instances du monde industriel. Les résultats montrent que le modèle PPC/PL est plus performant que les autres en nombre de nœuds et en temps d'exécution.

Une autre schéma de coopération que l'on trouve dans la littérature est la déduction de coupes linéaires à partir de la PPC. Demassey et al. [25] proposent cette méthode de coopération pour le calcul d'une borne inférieure pour le problème d'ordonnancement de projet à contraintes de ressources. Les bornes inférieures sont calculées à partir de relaxations linéaires d'un modèle de programmation en nombres entiers. À partir d'une solution réalisable avec une borne supérieure, des techniques de propagation par contraintes sont appliquées jusqu'à ce qu'il ne soit plus possible de faire des déductions. Ces déductions sont des deux types : réduction des domaines et coupes valides pour le problème linéaire. La méthode a été testée en utilisant les instances de Kolisch, Sprecher et Drexl [69] de 30 et 60 activités. Les résultats montrent que la technique proposée est plus performante que celle de Brucker et Knust [15] en terme du nombre d'instances prouvées à l'optimum avec les deux modèles de programmation en nombres entiers. Malgré cela, l'approche de Brucker et Knust est plus compétitive en terme de temps d'exécution.

1.6 Logiciels

Il existe principalement trois environnements qui permettent de modéliser un problème indépendamment du solveur à utiliser, nous pouvons citer les technologies : MOSEL de Dash, CONCERT et OPL Studio d'ILOG. Même si MOSEL peut appeler différents solveurs, en réalité MOSEL est dans une phase de développement et jusqu'à présent, ne peut qu'appeler le solveur de programmation linéaire XPRESS-MP de Dash. Plus ancienne, la technologie CONCERT a gagné beaucoup d'adeptes dans la nouvelle et croissante communauté PPC/PL dans le sens qu'on peut utiliser CONCERT pour modéliser des problèmes et les résoudre en appelant un solveur de programmation linéaire (CPLEX) et/ou un solveur de programmation par contraintes (SOLVER). D'autre part, un des produits d'ILOG, appelé HYBRID a été conçu pour intégrer les deux solveur de programmation par contraintes et programmation linéaire dans une seule librairie. Cependant, ce produit

n'a pas eu le succès espéré car il propose la résolution d'un même modèle par CPLEX et SOLVER alors que, en pratique, les modèles sont souvent différents.

1.7 Contributions

Les contributions de cette thèse peuvent être divisées en deux, selon le problème comme suit :

- Nous illustrons la difficulté de mettre en œuvre un algorithme hybride PPC/PL pour la résolution de problèmes difficiles (sac-à-dos multi-dimensionnel en variables binaires et le problème d'affection de fréquences).
- Nous étendons les événements de propagation des domaines, réalisées dans la littérature seulement pour les variables originales du problème, aux variables d'écarts permettant ainsi une meilleure propagation des informations obtenues à partir des coûts réduits.
- Nous proposons les contraintes de coûts réduits pour améliorer la propagation de l'information donnée par la programmation linéaire.
- Nous montrons le lien entre les contraintes de coûts réduits et les coupes mixtes de gomory.
- Nous proposons une contrainte de coûts réduits réversible en utilisant la relaxation surrogate du problème de sac-à-dos multidimensionnel en variables binaires.
- Nous proposons un nouveau modèle pour le problème d'affectation de fréquences dans un réseau de télécommunications par faisceaux hertziens.
- Nous proposons une extension des formulations classiques de PLNE pour prendre en compte les nouvelles contraintes du nouveau modèle.
- Nous montrons les difficultés rencontrées par la PLNE pour résoudre le problème d'affectation de fréquences.
- Nous proposons des algorithmes hybrides pour la résolution de nos deux problèmes objectif.
- Pris ensemble, nous avons avancé le champ des connaissances sur l'hybridation en clarifiant les difficultés que cette approche occasionne.

1.8 Structure de cette thèse

Ce mémoire est composé de deux parties. La première partie est consacrée à l'étude du problème de sac-à-dos multidimensionnel en variables binaires. Cette partie est divisée

en deux chapitres. Le chapitre 2 est dédié à une revue de la littérature sur l'état de l'art de ce problème tandis que le chapitre 3 est consacré à sa résolution. Celle-ci se fait à l'aide d'une procédure hybride PPC/PL où l'information obtenue par la PL, les coûts réduits, est utilisée afin de déduire de nouvelles informations de façon à améliorer la propagation. La deuxième partie est consacrée à l'étude du problème d'affectation de fréquences. Cette partie, est aussi divisée en deux chapitres. Le chapitre 4 fait un état de l'art sur ce problème tandis que le chapitre 5 est dédié à la modélisation et à la résolution de celui-ci. Nous proposons un modèle de programmation linéaire pour le problème d'affectation de fréquences ainsi qu'un modèle hybride pour sa résolution. Une conclusion fait ensuite le bilan des résultats obtenus et propose des perspectives.

Première partie

Le Problème du Sac à Dos Multidimensionnel 0-1

Chapitre 2

Formulation et méthodes de résolution pour le Problème de Sac à Dos Multidimensionnel

Ce chapitre présente une revue de l'état de l'art du Problème de Sac à Dos Multidimensionnel. Différentes relaxations du problème sont étudiées. Des techniques de prétraitement s'apparentant à de la propagation de contraintes basique sont présentées. Les principales méthodes exactes et heuristiques utilisées pour résoudre ce problème sont introduites.

2.1 Présentation du problème et des notations

Le Problème du Sac à Dos Multidimensionnel (PSDM) considère un ensemble d'éléments $N = \{1, \ldots, n\}$ et un ensemble de ressources $M = \{1, \ldots, m\}$ disponibles en quantités limitées. On note p_j le profit obtenu en sélectionnant l'élément $j \in N$. Chaque élément j nécessite, s'il est sélectionné, une quantité a_{ij} de la ressource i. On note ainsi A la matrice des consommations, A^j le vecteur colonne j et A_i le vecteur ligne i. On note b_i la quantité totale de ressource i disponible. Le PSDM revient à sélectionner un sous-ensemble d'éléments de façon à maximiser le profit total en respectant la disponibilité de chaque ressource.

La sélection d'un élément j peut être représentée par une variable binaire de décision x_j qui vaut 1 si l'élément est sélectionné et 0 sinon.

Pour un état de l'art récent sur le problème du sac à dos multidimensionnel, nous nous référons à [32].

2.2 Formulation

Le PSDM peut s'écrire comme le programme linéaire en nombres entiers (P) décrit ci dessous.

$$(P) \qquad z^*(P) = \max \quad \sum_{j=1}^{n} p_j x_j \tag{2.1}$$

$$s.a. \quad \sum_{j=1}^{n} a_{ij} x_j \leq b_i \quad \forall i \in M \tag{2.2}$$

$$x_j \in \{0,1\} \qquad \forall j \in N \tag{2.3}$$

avec $p_j \geq 0, \forall j \in N$, $a_{ij} \geq 0, \forall i \in M, \forall j \in N$, $b_i \geq 0, \forall i \in M$.

La fonction économique (2.1) décrit le profit total du modèle. Chacune des m contraintes en (2.2) est appelée contrainte du *sac à dos*. Le PSDM est ainsi appelé le *problème du sac à dos multidimensionnel*[1]. La plupart des recherches ont été réalisées sur le problème du sac à dos (à une seule contrainte). Ce problème n'est pas fortement NP-difficile et des algorithmes d'approximation efficaces ont été dévelopés pour obtenir des solutions proches de l'optimum. Une très bonne revue de l'état de l'art pour ce problème peut être trouvée dans [82] et plus récemment dans [81].

[1] D'autres noms on été donnés au problème, parmi eux : problème du sac à dos multirestreint, problème multi-sac à dos et problème sac à dos m-dimensionnel 0-1.

Si $z^*(P)$ est la valeur optimale de la fonction économique de P, on note $\bar{z}(P)$ une borne supérieure de la solution optimale de (P) et $\underline{z}(P)$ une borne inférieure.

2.3 Relaxations du PSDM

Plusieurs types de relaxations basées sur le PLNE de (P) ont été proposées pour résoudre le PSDM. Parmi celles-ci, on considère ci-après la relaxation de programmation linéaire RPL, la relaxation lagrangienne RL, le dual lagrangien DL, la relaxation surrogate RS et la relaxation mixte RC. On note $z_R^*(P)$ la valeur optimale de la relaxation $R \in \{RPL, RL, DL, RS, RC\}$.

Relaxation de Programmation Linéaire Soit le problème (P) de la section (2.2). La relaxation de Programmation Linéaire consiste à relacher les contraintes (2.3).

$$
\begin{aligned}
(\bar{P}) \qquad z^*_{RPL}(P) = \max \quad & \sum_{j=1}^{n} p_j x_j \\
s.a. \quad & \sum_{j=1}^{n} a_{ij} x_j \leq b_i \quad \forall i \in M \\
& 0 \leq x_j \leq 1 \quad \forall j \in N
\end{aligned}
$$

Relaxation Lagrangienne Soient $Q \subseteq M$, avec
$q =\mid Q \mid > 0$ et $\lambda \in \Re_+^q$, où $\Re_+ = \{y \in \Re / y \geq 0\}$. Soit

$$
\begin{aligned}
(P_{RL_Q}(\lambda)) \qquad z^*_{RL,Q}(\lambda)(P) = \max & \sum_{j=1}^{n} (p_j - \sum_{i \in Q} \lambda_i a_{ij}) x_j + \sum_{i \in Q} \lambda_i b_i \\
\text{s.a.} \quad & \sum_{j=1}^{n} a_{ij} x_j \leq b_i \qquad \forall i \in M \backslash Q \\
& x_j \in \{0, 1\} \qquad \forall j \in N
\end{aligned}
$$

$P_{RL_Q}(\lambda)$ est la Relaxation Lagrangienne de P relative à Q, avec une valeur Duale Lagrangienne donnée par

$$z^*_{DL,Q} = \min_{\lambda \geq 0} z^*_{RL,Q}(\lambda)$$

La relaxation lagrangienne s'est montrée efficace pour la résolution de problèmes de programmation en nombres entiers. En effet, pour n'importe quel ensemble $Q \subseteq M$,

$z^*_{DL,Q} \leq {z^*_{RPL}}^2$.

Pour le cas particulier du relâchement de toutes les contraintes, Everett [29] a montré comment les multiplicateurs de lagrange peuvent servir à la résolution de (P). Ces principes sont basés sur le théorème suivant.

Théorème 2.1 *Étant donné un vecteur non négatif de multiplicateurs* $\lambda = (\lambda_1, \ldots, \lambda_m)$ *et une solution* $x^* = (x^*_1, \ldots, x^*_n)$ *de*

$$\max_{x \in \{0,1\}^n} \{\sum_{j=1}^{n} p_j x_j - \sum_{i=1}^{m} \lambda_i \sum_{j=1}^{n} a_{ij} x_j\} \tag{2.4}$$

alors x^* *maximise*

$$Z(x) = \sum_{j=1}^{n} p_j x_j$$

s.à.

$$\sum_{j=1}^{n} a_{ij} x_j \leq \sum_{j=1}^{n} a_{ij} x^*_j = A_i x^* \qquad \forall i \in M$$
$$x_j = \{0,1\} \qquad \forall j \in N$$

Ce théorème peut être reformulé comme suit. Étant donné les multiplicateurs de lagrange, supposons que x^* soit la solution du problème de maximisation sans contraintes. De toutes les solutions qui utilisent au plus $A_i x^*$, la solution x^* donne le profit maximal. Or, le problème (2.4) est trivial puisqu'il revient à maximiser :

$$\max_{x \in \{0,1\}^n} \{\sum_{j=1}^{n} (p_j - \sum_{i=1}^{m} \lambda_i \sum_{j=1}^{n} a_{ij}) x_j\}$$

Une solution optimale est la suivante :

$$x^*_j = \begin{cases} 1 & \text{si } (p_j - \sum_{i=1}^{m} \lambda_i \sum_{j=1}^{n} a_{ij}) > 0 \\ 0 & \text{sinon} \end{cases} \tag{2.5}$$

La difficulté est de trouver des valeurs pour les multiplicateurs de lagrange tels que la solution x^* soit réalisable pour (P) et vérifie :

[2] $z_{RPL}(P)$ est la valeur optimale de la Relaxation de Programmation Linéaire.

$$\sum_{i=1}^{m} \lambda_i (b_i - \sum_{j=1}^{n} a_{ij} x_j^*) = 0$$

dans ce cas, x^* est une solution optimale de (P) selon le théorème 2.1.

Relaxation Surrogate Soit $\mu \in \Re_+^m$. La relaxation surrogate est donnée par

$$\begin{aligned} (P_{RS}(\mu)) \qquad z_{RS}^*(\mu)(P) = \qquad & \max \sum_{j=1}^{n} p_j x_j \\ \text{s.a.} \qquad & \sum_{j=1}^{n} (\sum_{i=1}^{m} \mu_i a_{ij}) x_j \leq \sum_{i=1}^{m} \mu_i b_i \qquad (2.6) \\ & x_j \in \{0,1\} \qquad \forall j \in N \end{aligned}$$

Le dual surrogate est alors donné par

$$z_{DS}^* = \min_{\mu \geq 0} z_{RS}^*(\mu)$$

La contrainte (2.6) est appelée *contrainte surrogate*. Une propriété du dual surrogate est que $z_{DS}^* \leq z_{DL,M}^*$. Ceci vient du fait que le dual lagrangien est une relaxation du dual surrogate pour n'importe quelle valeur des multiplicateurs.

Relaxation Mixte La relaxation mixte incorpore les aspects des relaxations lagrangienne et surrogate. Pour $\lambda, \mu \in \Re_+^m$, nous définissons

$$\begin{aligned} (P_{RC}(\lambda,\mu)) \qquad z_{RC}^*(\lambda,\mu)(P) = & \max \sum_{j=1}^{n} (p_j - \sum_{i \in Q} \lambda_i a_{ij}) x_j + \sum_{i \in Q} \lambda_i b_i \\ \text{s.a.} \qquad & \sum_{j=1}^{n} (\sum_{i=1}^{m} \mu_i a_{ij}) x_j \leq \sum_{i=1}^{m} \mu_i b_i \\ & x_j \in \{0,1\} \qquad \forall j \in N \end{aligned}$$

On obtient le dual mixte en calculant $z_{DC}^* = min_{\lambda,\mu \geq 0} z_{RC}^*(\lambda,\mu)$.

Crama et Mazzola [21] montrent que dans le cas des problèmes de sac à dos multidimensionnels, l'amélioration de la borne par ces différentes relaxations peut être limitée a priori. En particulier, ils démontrent que le gain obtenu par rapport à la borne de re-

laxation de programmation linéaire n'excède jamais la valeur du coefficient maximal de la fonction objectif, ni la moitié de la borne de programmation linéaire.

2.4 Techniques de prétraitement et de propagation de contraintes

Les techniques de prétraitement, ou de réduction, ont pour but de réduire la taille des problèmes. Elles sont très efficace surtout lorsque les problèmes s'avèrent difficiles à résoudre en permettant à l'algorithme de se concentrer sur le cœur du problème. Plusieurs chercheurs ont présenté des méthodes de réduction pour le PSDM. Mentions, parmi autres, Osorio et al [94], Fréville et Plateau [34], Trick [117] et Gavish et Pirkul [41]. En fait toutes ces techniques s'apparentent implicitement (ou explicitement pour la méthode de Trick) à des techniques de propagation de contraintes (voir chapitre 1). Le prétraitement d'un programme linéaire en nombres entiers par la propagation de contraintes constitue ainsi un premier niveau d'hybridation.

2.4.1 Techniques de base

Avant d'expliquer de tests plus sophistiqués pour la réduction de la taille du problème (P), les tests élémentaires suivants permettent d'obtenir un problème bien formé :

(Règle 1)	S'il existe un élément $j \in \{1, 2, \ldots, n\}$ et une ressource $i \in \{1, 2, \ldots, m\}$ vérifiant : $a_{ij} > b_i$ alors la variable x_j peut être fixée à 0
(Règle 2)	S'il existe une ressource $i \in \{1, 2, \ldots, m\}$ vérifiant : $\sum_{j=1}^{n} a_{ij} \leq b_i$ alors la contrainte i peut être éliminée
(Règle 3)	S'il existe un élément $j \in \{1, 2, \ldots, n\}$ tel que : $A^j = 0$ alors la variable x_j peut être fixée à 1

Tab. 2.1 – Techniques de Réduction de Base

Il faut noter que tous les algorithmes présentés ici utilisent ces tests de base.

2.4.2 Algorithmes de fixation de variables

Les méthodes classiques de fixation de variables pour les problèmes en variables mixtes ont été utilisées et développées pour le PSDM. Elles reposent essentiellement sur le principe suivant. Soit $BI = \underline{z}(P)$ une borne inférieure de (P) et soit $\bar{z}(P|x_j = v)$ une borne supérieure de la solution optimale de (P) si la variable j est fixée à la valeur $v \in \{0, 1\}$. Si $\bar{z}(P|x_j = v) \leq BI$ et si BI n'est pas optimale alors x_j peut être fixée à la valeur $1 - v$. $\bar{z}(P|x_j = v)$ peut par exemple être obtenue immédiatement en utilisant le coût réduit de x_j dans la relaxation de programmation linéaire de (P). Dans le chapitre suivant nous utiliserons ce principe en étudiant ces contraintes "de coût réduit" dans un formalisme hybride programmation par contraintes / programmation linéaire.

L'algorithme de Fréville et Plateau [36] repose également sur ce principe. Il est basé sur des techniques d'élimination de variables et de contraintes, en trois phases. La première phase calcule une borne supérieure sur la valeur optimale en utilisant les relaxations lagrangienne et surrogate (section 2.3). Dans la deuxième phase, l'algorithme calcule une borne inférieure en utilisant les méthodes heuristiques AGNES [34]. Finalement, l'élimination de variables et de contraintes sont faites en utilisant la conjonction de tests simples et des principes énoncés ci-dessus, à partir de la borne inférieure et de la borne supérieure.

Gavish et Pirkul [41] utilisent également la relaxation surrogate pour fixer des variables. Soient x^* la solution de $P_{RS}(\mu)$ (voir section 2.3) et $\bar{P}_{RS}(\mu)$ la relaxation de programmation linéaire de $P_{RS}(\mu)$. L'approche propose de définir, pour chaque variable, le problème $P_{i_{RS}}(\mu|x_i = 1 - x_i^*)$ obtenu en fixant la variable x_i à $(1 - x_i^*)$ où x_i^* est la i-ème composante de x^*. En utilisant le principe de fixation présenté ci-dessus, nous pouvons fixer les variables x_i à x_i^* si la valeur optimale de $\bar{P}_{i_{RS}}(\mu/x_i = 1 - x_i^*)$ est plus petite que la meilleure solution réalisable connue. Gavish et Pirkul [41] montrent l'efficacité de leur méthode à travers deux expériences sur un ensemble d'instances générées aléatoirement. La première consiste à prendre comme borne inférieure la valeur de la solution optimale pour chaque problème afin de connaître les meilleurs résultats pouvant être obtenus par cette méthode. Le pourcentage de variables fixées est en moyenne de 90%. Les auteurs étudient ensuite l'impact d'une borne inférieure de qualité moindre sur les résultats. Enfin, ils constatent que la procédure proposée est moins efficace si le second membre b_i de la contrainte décroît par rapport à $\sum_j a_{ij}$. En effet, la diminution du second membre entraine un écart plus grand entre la borne et la valeur de la solution optimale.

2.4.3 Algorithmes de fixation de variables et de génération de coupes logiques

L'algorithme d'Osorio et al [94] combine l'analyse surrogate et le concept d'appariement de contraintes pour fixer certaines variables à 0 et pour séparer le reste de variables dans deux groupes : celles qui tendent à 0 et celles qui tendent à 1 dans une solution optimale entière. En utilisant une borne inférieure initiale (BI), l'algorithme génère des coupes logiques à la racine d'un arbre de recherche. Plus précisement, l'article propose un appariement de deux contraintes, la contrainte surrogate et la fonction objectif ($\sum_{j\in N} c_j x_j$). Afin d'obtenir les valeurs duales surrogates, les bornes $x_j \leq 1$ sont intégrées comme contraintes dans la relaxation linéaire du problème. La contrainte surrogate obtenue a alors la forme suivante :

$$\sum_{j\in N}(\nu_j + \sum_{i\in M}\mu_i a_{ij})x_j \leq \sum_{i\in M}\mu_i b_i + \sum_{j\in N}\nu_j \tag{2.7}$$

où ν_j est la valeur duale de la contrainte $x_j \leq 1$ et μ_i est la valeur duale de la contrainte $i \in M$ correspondante. On pose $BS = \sum_{i\in M}\mu_i b_i + \sum_{j\in N}\nu_j$. C'est en effet la valeur optimale de la relaxation de programmation linéaire et donc une borne supérieure du PSDM. On note $s_j = (\nu_j + \sum_{i\in M}\mu_i a_{ij})$. BI étant la meilleure borne inférieure connue, le système suivant de paires de contraintes est obtenu :

$$\sum_{j\in N} c_j x_j \geq BI \tag{2.8}$$

$$\sum_{j\in N}(\nu_j + \sum_{i\in M}\mu_i a_{ij})x_j \leq BS \tag{2.9}$$

En combinant (2.8) et (2.9) et la contrainte résultante devient :

$$\sum_{j\in N}(s_j - c_j)x_j \leq BS - BI \tag{2.10}$$

Les auteurs remarquent que si la valeur de la solution de programmation linéaire de la variable x_j est non nulle, alors $(\nu_j + \sum_{i\in M}\mu_i a_{ij}) = c_j$. Il faut ainsi noter que la contrainte (2.10) contient moins des variables que celle de la fonction objectif. Ceci vient du fait que pour les variables vérifiant $x_j > 0$ dans la solution optimale de programmation linéaire, $s_j = c_j$. Suivant la règle 1 présentée dans le tableau 2.1, on peut fixer à 0 toute variable x_j

de coefficient $s_j - c_j \geq BS - BI$. De plus, la contrainte (2.10) est une contrainte de sac-à-dos à coefficients positifs, ce que les auteurs utilisent pour générer des coupes logiques de type "covering inequalities". Dans le chapitre suivant, nous comparons la contrainte (2.10) avec la contrainte de coûts réduits proposée dans cette thèse.

L'algorithme proposé par les auteurs a été implanté à la racine d'un arbre de recherche de CPLEX 7.5 comme une procédure additionnelle au prétraitement utilisé par default. Les résultats montrent que l'approche est plus performante que celui de CPLEX 7.5 en nombre de nœuds mais moins performante en temps de calcul.

2.4.4 Programmation par contraintes et hyper-arc consistance

A notre connaissance, la seule approche de programmation par contraintes utilisée pour le problème de sac à dos multidimensionnel est celle de Trick [117].

Dans cette approche, où les variables considérées sont entières, chaque contrainte de sac à dos (y compris éventuellement la contrainte surrogate) est gérée comme une contrainte globale au sens de la programmation par contraintes (voir chapitre 1). L'auteur constate que sur les contraintes de sac à dos de type $L \leq A_i x \leq U$, seule une consistance aux bornes est en général effectuée, par exemple par les algorithmes de fixation présentés ci-dessus. Il propose à la place d'utiliser la programmation dynamique pour assurer l'hyper-arc consistance, c'est-à-dire d'assurer que pour toute valeur du domaine d'une variable (ici 0 ou 1), il existe une affectation des autres variables respectant la contrainte de sac-à-dos (borne inférieure et supérieure). Une structure de données permettant l'incrémentalité des mises à jour est proposée. Cette approche a un inconvénient assez important. En effet, si le second membre U de la contrainte sac-à-dos est suffisamment grand, le temps d'exécution et les besoins d'espace mémoire peuvent être prohibitifs. Les temps de calculs, même avec une mise à jour incrémentale peuvent être en théorie beaucoup plus longs que les approches basées sur la consistance aux bornes. Cependant, les résultats expérimentaux montrent une réduction considérable du nombre de branchements et des temps de calcul pour certaines instances difficiles d'un problème de sac à dos multidimensionnels voisin de celui étudié dans cette thèse.

2.5 Méthodes exactes

Plusieurs chercheurs ont présenté des algorithmes exacts pour le PSDM. Mentionnons, entre autres, Balas [10], Gilmore et Gomory [42], Weingartner et Ness [125], Cabot [16],

Shih [112], Gavish et Pirkul [41], Osorio et al [94]. Nous distinguons les approches d'énumération implicite pour les programmes en 0-1, les méthodes de programmation dynamique et les procédures de séparation et évaluations basées sur la PLNE. A part l'approche de Trick [117] présentée à la section précédente, nous n'avons trouvé aucune méthode exacte basée sur la programmation par contraintes.

2.5.1 Enumération implicite 0-1

L'algorithme de Balas [10] est une procédure systématique qui débute avec une solution nulle. L'algorithme affecte successivement certaines variables à la valeur 1 de telle façon qu'après avoir essayé une partie des 2^n combinaisons, on obtienne soit une solution optimale, soit une preuve d'irréalisabilité. Les résultats expérimentaux rapportés pour des implémentations plus récentes de ces méthodes ne sont pas competitifs [32]. Il ne dépassent pas en effet 45 variables et 30 contraintes.

2.5.2 Programmation dynamique

Les algorithmes de Gilmore et Gomory [42], Weingartner et Ness [125] appartiennent à cette classe. Le dernier algorithme a résolu deux problèmes de taille $(n = 28, m = 2)$ et $(n = 105, m = 2)$ dans une approche avant et arrière. Bien que Freville [32] conclue à l'inefficacité de ces méthodes en tant que telles, on peut noter les résultats intéressant obtenus par l'approche hybride programmation par contraintes / programmation dynamique de Trick [117] pour un problème voisin (voir section précédente).

2.5.3 Procédures de séparation et d'évaluation pour la PLNE

Les algorithmes énumératifs pour les problèmes de programmation en nombres entiers utilisent les bornes dérivées de relaxations bien connues, telles la relaxation lagrangienne, surrogate ou mixtes (voir section 2.3). Les bornes calculées de cette manière sont généralement meilleures que la borne obtenue par relaxation continue du problème.

L'algorithme de Shih [112] est un algorithme qui utilise la relaxation de programmation linéaire. A chaque nœud de l'arbre de recherche m problèmes de sac à dos sont résolus :

$$
\begin{array}{lll}
(\bar{P}_i) & z(\bar{P}_i) = \max & \sum_{j=1}^{n} p_j x_j \\
& s.a. & \sum_{j=1}^{n} a_{ij} x_j \leq b_i \\
& & 0 \leq x_j \leq 1 \qquad \forall j \in N
\end{array}
$$

où la borne supérieure est la valeur minimale des valeurs de chaque fonction objectif, i.e. $\bar{z}(P) = min_i\{z(\bar{P}_i)\}$. Cet algorithme a été testé sur un ensemble de problèmes générés aléatoirement avec une taille maximale égale à $m = 5$ et $n = 90$. Les résultats expérimentaux montrent que cet algorithme est plus performant que celui de Balas [10].

Gavish et Pirkul [41] proposent d'utiliser les diverses relaxations du problème mentionnées à la section 2.3 et précisent les relations théoriques entre ces relaxations. Des expériences détaillées ont été effectuées pour comparer les bornes produites par ces relaxations. Leurs résultats ont prouvé que la relaxation mixte obtient les meilleures bornes au prix d'un effort supplémentaire en temps de calcul. Pour obtenir des bornes surrogates, de nouveaux algorithmes ont été développés et testés. Des règles pour réduire la taille des problèmes ont été suggérées et se sont avérées pertinentes expérimentalement. Enfin, l'algorithme définitif a été testé sur un ensemble de problèmes générés aléatoirement avec une taille maximal de $m = 5$ et $n = 200$. Il a été comparé avec la méthode exacte de Shih [112] et s'est avéré plus rapide, au moins d'un ordre de magnitude. Il a été également comparé favorablement au solveur commercial Sciconic/VM.

2.6 Méthodes heuristiques

Nous entreprenons la deuxième partie de ce chapitre, à savoir la présentation de méthodes heuristiques et métaheuristiques.

2.6.1 Heuristiques

Nous décrivons ci-après quelques heuristiques spécifiques au PSDM.

Heuristiques Primales. Les heuristiques primales débutent avec un vecteur solution égal à 0. Selon une règle donnée, une succession de variables candidates sont affectées à la valeur 1 tant que la solution reste réalisable. Parmi ces méthodes nous citons celles de Kochenberger et al. [68], Toyoda [116] et Loulou et Michaelides [77]. L'algorithme de Loulou et Michaelides [77] choisit la variable ayant le plus grand *pseudo-ratio* parmi toutes les candidates de chaque étape. Le *pseudo-ratio* est défini par $u_j = p_j/v_j$ où v_j dépend des coefficients a_{ij} et peut être définie de plusieurs manières. Cette heuristique a été testée sur des problèmes générés aléatoirement. La déviation moyenne par rapport à l'optimum va de 0.26% à 1.08% pour les petits problèmes mais peut atteindre 14% pour les problèmes plus grands.

L'algorithme exact de Gavish et Pirkul [41] peut également être utilisé comme heuristique en interrompant la recherche. Les auteurs rapportent que cette méthode est supérieure à l'heuristique développée par Loulou et Michaelides [77].

Heuristiques Duales. Les heuristiques duales débutent avec un vecteur solution égal à 1. Selon une règle donnée, une succession de variables candidates est fixé à 0 jusqu'à l'obtention d'une solution réalisable. Parmi ces méthodes nous citons celle de Senju et Toyoda [110].

Heuristique basée sur la relaxation de programmation linéaire. [3] Parmi ces approches, on trouve l'algorithme de Hillier [52] et celui de Balas et Martin [11]. L'algorithme de Hiller utilise une stratégie de trois phases. Dans la première phase la procédure essaye d'identifier une région prometteuse en déterminant la solution optimale continue (relaxation de programmation linéaire) et un autre point voisin dont la solution arrondie entière ne viole pas les contraintes. Dans la deuxième phase, l'algorithme cherche un segment entre ces deux points afin d'identifier une meilleure solution réalisable. La troisième phase est une méthode d'amélioration de la solution réalisable obtenue à la phase 2, consistant à changer itérativement une ou deux variables à la fois.

Zanackis [128] a donné des résultats détaillés en comparant l'algorithme de Kochenberger et al [68], Senju et Toyoda [110] et celle de Hillier [52]. Aucune de ces heuristiques ne s'est avérée dominer les autres.

Heuristiques basées sur les Multiplicateurs de Lagrange Magazine et Oguz [78] ont présenté un algorithme qui combine les idées de Senju et Toyoda [110] avec l'approche de multiplicateurs de lagrange généralisés de Everett [29]. Leur méthode débute avec un vecteur de multiplicateurs de lagrange nul et toutes les variables égale à 1 de telle sorte que la contrainte (2.5) est satisfaite. En général, cette solution n'est pas réalisable puisque les contraintes 2.2 sont violées. À chaque itération, l'approche détermine la contrainte i qui a le rapport capacité restante de la ressource $(b_i - \sum_{j=1}^{n} a_{ij}x_j)$ sur capacité totale b_i le plus grand. Alors, la valeur du multiplicateur associé à cette contrainte est augmenté jusqu'à ce qu'une seule variable viole la condition (2.5). La valeur 0 est affectée à cette variable. Cette phase, est répétée jusqu'à l'obtention d'une solution réalisable. Enfin, dans la dernière phase la méthode applique une heuristique gloutonne afin d'améliorer la solution courante. L'heuristique, que nous utiliserons au chapitre suivant, est détaillée dans l'algorithme 1.

[3] Voir page 21.

Algorithme 1 Algorithme de Magazine et Oguz

1: Initialiser $\lambda_i = 0 \quad \forall i \in M$, et $x_j = 1 \quad \forall j \in N$.
Normaliser les coefficients $a'_{ij} = a_{ij}/b_i \quad \forall i \in M, \forall j \in N$ et $b_i = 1 \quad \forall i \in M$.
Calculer $y_i = \sum_{j=1}^{n} a_{ij} \quad \forall i \in M$.
2: Si $y_i \leq 1$ alors
3: Arrêter
4: fin si
5: Déterminer la contrainte i' la plus violée. $y_{i'} := \max_i y_i$
6: Calculer l'augmentation du multiplicateur de lagrange $\lambda_{i'}$. Pour tout $j \in N$ tel que $x_j = 1$

$$\theta_j = \begin{cases} (p_j - \sum_{i=1}^{m} \lambda_i \sum_{j=1}^{n} a_{ij})/a_{i'j} & \text{si } a_{i'j} > 0 \\ \infty & \text{sinon} \end{cases}$$

Soit $\theta_{j'} := \min_j \{\theta_j / x_j = 1\}$.
7: Augmentation des multiplicateurs et fixation des variables
$\lambda_{i'} := \lambda_{i'} + \theta_{j'}$
$x_{j'} := 0$ et $y_i := y_i - a_{ij'} \quad \forall i \in M$
Si $y_i \leq 1 \quad \forall i$ alors aller à 8 ; sinon aller à 5.
8: Amélioration de la solution. Vérifier si les variables qui sont nulles peuvent être fixées à la valeur 1 sans violer les contraintes $y_i \leq 1$. Si c'est le cas, alors choisir la variable avec le profit le plus grand. Répéter cette étape jusqu'à aucune variable de ce type ne puisse être trouvée.

Si x^* est une solution et si λ est le vecteur de multiplicateurs de lagrange à la fin de l'étape 7. Alors une borne supérieure pour le problème (P) est donné par :

$$BS_\lambda = Z(x^*) + \sum_{i=1}^{m} \lambda_i (b_i - \sum_{j=1}^{n} a_{ij} x_j^*)$$

Volgenant et Zoon [123] ont présenté une amélioration de l'algorithme ci-dessus en démontrant qu'on peut changer plus d'un multiplicateur à la fois, ce qui change les étape 5 et 7 par les suivantes :

5 : Déterminer les contraintes i' et i'' les plus violées. $y_{i'} := \max_i y_i$ et $y_{i''} := \max_i y_i : i \neq i'$

7 : $\lambda_{i'} := \lambda_{i'} + \theta_{j'} \times y_{i'}/y_{i''}$ et $\lambda_{i''} := \lambda_{i''} + \theta_{j'}$

Cependant les résultats expérimentaux n'améliorent pas significativement ceux de la méthode originale. En effet, en moyenne, la qualité de la solution et de la borne supérieure ne sont pas suffisamment bons au regard du temps de calcul requis.

Heuristique de Lee et Guignard [75]. Lee et Guignard [75] ont présenté une heuristique qui combine une méthode primale, la relaxation de programmation linéaire et une procédure, qui transforme les variables en leur complémentaires, de Balas et Martin [11].

Des résultats expérimentaux ont été présentés avec quelques instances de la littérature et d'autres générées aléatoirement de taille jusqu'à $m = 20$ et $n = 200$. Leur algorithme est supérieur, par rapport à la qualité de la solution, à ceux proposés par Toyoda [116] et Magazine et Oguz [78] mais inférieur à celui de Balas et Martin [11].

Heuristiques basées sur les Relaxations RL, RS, RC Les méthodes proposées par Fréville et Plateau [34] et Osorio et al [93] appartiennent à ce type d'heuristiques. Fréville et Plateau exploitent les idées déjà présentées pour la fixation des variables à la section 2.4.2, à savoir :

1. Trouver une borne supérieure de la valeur optimale en utilisant les relaxations lagrangienne et surrogate.
2. Trouver une borne inférieure en utilisant des méthodes heuristiques efficaces.
3. Éliminer des variables et des contraintes par le conjonction de tests simples.

Dans la phase 2, Fréville et Plateau proposent deux heuristiques appelées AGNES 1 et AGNES 2. Ces deux heuristiques sont basées sur la relaxation surrogate de (P). L'algorithme AGNES 1 génère, en plusieurs itérations d'un même processus, un ensemble de solutions réalisables pour le problème (P). Pour générer chaque solution, AGNES 1 utilise la solution optimale de la relaxation linéaire de la relaxation surrogate. Ils construisent à partir de l'ensemble des variables égales à 1 dans cette solution, un sous-ensemble de variables respectant les contraintes de (P). Ces variables sont alors fixées à 1 et le processus et itéré. À la différence de la méthode précédente, l'algorithme AGNES 2 fixe également à chaque étape des variables à 0.

Les résultats montrent que les heuristiques AGNES 1 et AGNES 2 sont supérieures aux algorithmes présentés jusqu'à ici.

L'algorithme d'Osorio et al [93] présenté dans le cadre du prétraitement (voir section 2.4.3) et utilisant la relaxation surrogate obtient de meilleures solutions approchées que CPLEX.

2.6.2 Métaheuristiques

Récuit simulé [67]

Le récuit simulé fut appliqué au PSDM par Drexl [26]. Il a proposé un mouvement aléatoire du type 2-exchange (échange de deux objets). Si cet échange n'est pas possible dû aux contraintes de ressources alors il propose d'exclure ou de supprimer une variable k

de la solution réalisable qui dans certain cas est contrôlée par la température. Les résultats expérimentaux sur 57 instances de la littérature montrent que l'algorithme obtient dans 23 cas la solution optimale. Les instances résolues ont pour taille $m = 2$ à $m = 30$ et de $n = 6$ à $n = 105$. Ces instances sont considerées petites et de facile solution.

2.6.3 Recherche Tabou [44], [45], [46]

Plusieurs auteurs ont appliqué la recherche tabou au PSDM, entre autres [48] et avec succès [120].

Hanafi et Fréville [48] proposent une approche de recherche tabou basée sur l'oscillation stratégique et l'information obtenue de la contrainte surrogate. Les auteurs rapportent des résultats numériques sur des problèmes de budget de capital de la littérature où la qualité des solutions obtenues avec leur méthode est au moins aussi bonne que les meilleures solutions obtenues avec les autres approches. Cependant, ils ne présentent pas de résultats numériques sur des problèmes de grande taille de la OR-library[4].

Vasquez et Hao [120] proposent une approche qui combine la programmation linéaire et la recherche tabou. L'idée esentielle est la recherche autour de la solution optimale continue $\bar{x}$ d'une relaxation du PSDM. Leur hypothèse est que l'espace de recherche autour de $\bar{x}$ contient des solutions de haute qualité. Le vecteur $\bar{x}$ contient l'information globale qui guidera l'algorithme tabou tout en contrôlant le mécanisme de visite du voisinage d'une configuration courante x. Pour ce faire, les auteurs limitent la recherche locale aux seuls points $x^{'}$ tels que : $distance(x^{'}, \bar{x}) \leq \sum_{j \in N} |x_j^{'} - x_j|$. D'autre part, les auteurs font la remarque suivante : toute solution du PSDM vérifie une égalité du type :

$$\sum_{j=1}^{n} x_j = k \text{ avec } k \in \mathbb{N}$$

Ils définissent ainsi une série de problèmes du type :

$$z^*(P) = \max \quad \sum_{j=1}^{n} p_j x_j \tag{2.11}$$

$$s.a. \quad \sum_{j=1}^{n} a_{ij} x_j \leq b_i \quad \forall i \in M \tag{2.12}$$

$$\sigma(x) = k \tag{2.13}$$

$$x_j \in \{0, 1\} \qquad \forall j \in N \tag{2.14}$$

[4]Librairie de problèmes tests d'évaluation en http ://mscmga.ms.ic.ac.uk/jeb/orlib/mknapinfo.html

où $\sigma(x)$ est la somme des composantes du vecteur x. La méthode dispose alors de plusieurs points $\bar{x}_k$ pour explorer l'espace de recherche. En effet, la contrainte $\sigma(x) = k$ est utilisée comme un mécanisme stratégique de diversification pour la recherche tabou. Pour résumer, l'algorithme proposé par Vazquez et Hao se décompose en trois étapes :

1. détermination des valeurs de k intéressantes ;
2. utilisation du simplexe pour calculer les $\bar{x}_k$ correspondants ;
3. exécution de la recherche locale tabou autour de ces points.

Les auteurs raportent que leur algorithme a amélioré significativement la borne inférieure meilleure connue de 150 sur 270 instances d'évaluation.

2.6.4 Algorithmes génétiques [47]

Plusieurs auteurs ont appliqué avec succès les algorithmes génétiques au [PSDM], parmi eux celui de Chu et Balas [28]. Les auteurs ont modifié le schéma classique des algorithme génétiques en incorporant la connaissance spécifique du problème. Ils utilisent la représentation classique des individus, c'est-à-dire la représentation binaire. Puisqu'une solution peut être non-réalisable pour le PSDM, ils utilisent un opérateur de réparation qui garantit sa transformation en une solution réalisable. Cette opérateur est une simple heuristique gloutonne pour le Problème de Sac à Dos Unidimensionnel. Un tel problème est facile à obtenir en utilisant la relaxation surrogate du PSDM (voir page 23). Les multiplicateurs que les auteurs ont choisi pour construire la contrainte surrogate, sont simplement les valeurs duales obtenues en résolvant la relaxation de programmation linéaire du problème original.

Pour les petits problèmes de la littérature, c'est-à-dire $m = 2$ à $m = 30$ et $n = 6$ à $n = 105$ la solution optimale est toujours trouvée avec un temps moyen d'exécution égal à 8 fois le temps d'exécution de CPLEX MIP Solver. Pour les problèmes de grande taille, c'est-à-dire $m = 5$ à $m = 30$ et $n = 100$ à $n = 500$ le temps moyen d'exécution est égal à $0,4$ fois le temps d'exécution de CPLEX MIP-Solver et les résultats numériques présentés par les auteurs donnent un *gap* plus petit que CPLEX, en moyenne. Enfin, la comparaison avec les heuristiques de Magazine et Oguz [78], Volgenant et Zoon [123] et Pirkul [105] donnent la suprématie à l'algorithme génétique.

Cependant, l'algorithme développé par Vasquez et Hao [120] est apparemment meilleur que l'algorithme présenté ci-dessus. En effet, les résultats numériques obtenus sur 30 problèmes de taille $m = 30$ et $n = 500$ sont toujours supérieurs, en borne et en temps cpu, que ceux obtenus par Chu et Balas [28].

Chapitre 3

Expérimentation de l'utilisation des coûts réduits pour résoudre le Problème du Sac à dos Multidimensionnel 0-1.

L'objectif général de ce chapitre est de présenter une approche de coopération entre la Programmation linéaire et la Programmation Par Contrainte pour le PSDM. Pour cela, nous proposons une nouvelle idée qui consiste à utiliser les coûts réduits des variables du problème pour générer un ensemble de contraintes logiques parmi lesquelles au moins une doit être satisfaite. Finalement, nous présentons des résultats numériques d'une procédure de séparation et évaluation basée sur ces principes.

Les travaux présentés dans ce chapitre ont été en partie présentés dans [89, 87, 90].

3.1 Programmation par Contraintes basée sur les coûts réduits

L'idée de base de notre approche est de propager l'information induite par les coûts réduits. Néanmoins, cette information peut seulement être exploitée si une solution entière réalisable, si possible de bonne qualité, est connue. Par conséquent, dans une étape préliminaire, notre méthode calcule d'abord une solution réalisable au moyen d'une heuristique présentée dans la section 3.1.1, la méthode étant décrite en 3.1.2.

3.1.1 Calcul d'une solution réalisable de départ

Afin d'obtenir une solution réalisable de bonne qualité, nous avons implanté une version modifiée de l'algorithme de Volgenant et Zoon [123] . À l'étape 8 (voir la page 31), nous pouvons améliorer la solution en résolvant exactement un PSDM réduit. En effet, le nombre de variables qui sont à 0 au début de l'étape 8 est minimal et le côté droit des contraintes est petit. Supposons que les premières k variables sont à 0 dans la solution, alors la meilleure amélioration est donnée par :

$$\begin{aligned} \max \quad & \sum_{j=1}^{k} p_j x_j & & \\ s.a. \quad & \sum_{j=1}^{k} a_{ij} x_j \leq b_i - \sum_{j=k+1}^{n} a_{ij} & & \forall i \in M \\ & x_j \in \{0,1\} & & j = 1, \ldots, k \end{aligned}$$

C'est cette modification que nous proposons à l'algorithme de Volgenant et Zoon [123]. Il faut noter que même si l'article de Volgenant et Zoon mentionne que nous pouvons modifier plusieurs multiplicateurs à la fois, des résultats expérimentaux sont fournis pour la modification d'au plus deux multiplicateurs. Nous avons étendu l'implementation de l'algorithme pour la modification dynamique de plusieurs multiplicateurs à la fois. Nos résultats montrent que cet algorithme est supérieur à celui de Volgenant et Zoon [123]. La procédure est très simple mais les résultats expérimentaux montrent que les solutions sont assez proches des solutions optimales ou des meilleures solutions connues avec un temps d'exécution négligeable, voir la section 3.4.

3.1.2 Idée de base : la contrainte de coûts réduits

Les coûts réduits sont couramment utilisés en PMNE pour réduire les bornes des variables. De telles réductions de domaines peuvent également être accomplies efficacement par des techniques de PPC utilisées en conjonction avec la relaxation continue du problème (voir [31]). Nous proposons d'aller plus loin dans cette direction en étendant les événements de réduction de domaines aux variables d'écart.

Reconsidérons le PSDM décrit à la section 2.2.

$$\begin{aligned} (P) \qquad \max \quad & \sum_{j=1}^{n} p_j x_j \\ s.a. \quad & \sum_{j=1}^{n} a_{ij} x_j \leq b_i \quad i = 1, \ldots, m \\ & x \in \{0,1\}^n \end{aligned}$$

Supposons que nous connaissons une borne inférieure BI[1] à sa valeur optimale. Considérons également sa relaxation continue

$$\begin{aligned} (\bar{P}) \qquad \max \quad & \sum_{j=1}^{n} p_j x_j \\ s.a. \quad & \sum_{j=1}^{n} a_{ij} x_j \leq b_i \quad i = 1, \ldots, m \\ & 0 \leq x_j \leq 1 \quad j = 1, \ldots, n \end{aligned}$$

qui sous la forme standard devient (s_i étant la variable d'écart de la i-ème contrainte) :

$$\begin{aligned} \max \quad & \sum_{j=1}^{n} p_j x_j \\ s.a. \quad & \sum_{j=1}^{n} a_{ij} x_j + s_i = b_i \qquad i = 1, \ldots, m \\ & 0 \leq x_j \leq 1 \quad j = 1, \ldots, n, s_i \geq 0 \quad i = 1, \ldots, m \end{aligned}$$

Soit $(\bar{x}, \bar{s})$ une solution optimale de ce problème relâché et BS sa valeur optimale. Soit le vecteur de coûts réduits $(\bar{c}, \bar{u})$ correspondant aux variables (x, s) pour la solution de base $(\bar{x}, \bar{s})$. Le problème $\bar{P}$ peut s'écrire en fait, de manière totalement équivalente, comme suit :

$$\begin{aligned} (\bar{P_{eq}}) \quad \max \quad & BS + \sum_{j \in N^-} \bar{c}_j x_j - \sum_{j \in N^+} \bar{c}_j (1 - x_j) + \sum_{i \in M} \bar{u}_i s_i \\ s.a. \quad & \bar{A}x + \bar{S}s = \bar{b} \\ & 0 \leq x \leq 1, s \geq 0 \end{aligned}$$

[1] Pour en obtenir une, nous pouvons utiliser une des heuristiques présentées à la section (2.6). Nous avons choisi l'heuristique proposée à la section 3.1.1.

où $\bar{A}, \bar{S}, \bar{b}$ dépendent de la base associée à $(\bar{x}, \bar{s})$ et

$$N^- = \{\ j \in N : x_j \text{ est une variable hors base à sa borne inférieure}\}$$

$$N^+ = \{\ j \in N : x_j \text{ est une variable hors base à sa borne supérieure}\}$$

De plus, par définition de l'optimalité, nous savons que les coûts réduits satisfont les conditions suivantes :

$$\bar{x_j} = 1 \Rightarrow \bar{c_j} \geq 0$$

$$0 < \bar{x_j} < 1 \Rightarrow \bar{c_j} = 0$$

$$\bar{x_j} = 0 \Rightarrow \bar{c_j} \leq 0$$

$$\bar{s_i} > 0 \Rightarrow \bar{u_i} = 0$$

$$\bar{s_i} = 0 \Rightarrow \bar{u_i} \leq 0$$

En fait, nous pouvons même dire que le problème suivant (P_{eq}) est parfaitement équivalent à (P).

$$\begin{array}{lll} (P_{eq}) & \max & BS + \sum_{j \in N^-} \bar{c}_j x_j - \sum_{j \in N^+} \bar{c}_j (1 - x_j) + \sum_{i \in M} \bar{u}_i s_i \\ & s.a. & \bar{A}x + \bar{S}s = \bar{b} \\ & & x \in \{0,1\}^n, s \geq 0 \end{array}$$

Si A et b sont entiers, alors $s \in \mathbb{N}^0$. Par la suite, le problème

$$\begin{array}{lll} (\tilde{P_{eq}}) & \max & BS + \sum_{j \in N^-} \bar{c}_j x_j - \sum_{j \in N^+} \bar{c}_j (1 - x_j) + \sum_{i \in M} \bar{u}_i s_i \\ & & x \in \{0,1\}^n, s \in \mathbb{N}^0 \end{array}$$

constitue une relaxation de (P), notre problème original. Nous pouvons affirmer que $(\bar{x}, \bar{s})$ constitue une solution optimale de la relaxation continue et que la valeur optimale de $(\tilde{P_{eq}})$ est BS. En plus, la solution courante BI satisfait la relation suivante :

$$BS + \sum_{j \in N^-} \bar{c}_j x_j - \sum_{j \in N^+} \bar{c}_j (1 - x_j) + \sum_{i \in M} \bar{u}_i s_i \quad \geq BI$$

ou

$$- \sum_{j \in N^-} \bar{c}_j x_j + \sum_{j \in N^+} \bar{c}_j (1 - x_j) - \sum_{i \in M} \bar{u}_i s_i \quad \leq BS - BI \tag{3.1}$$

Néanmoins, puisque nous connaissons une solution entière, nous souhaitons obtenir des solutions meilleures que la de valeur BI, c'est-à-dire à trouver des solutions réalisables qui satisfassent la contrainte suivante que nous appelons *"contrainte de coûts réduits"*.

$$BS + \sum_{j \in N^-} \bar{c}_j x_j - \sum_{j \in N^+} \bar{c}_j (1 - x_j) + \sum_{i \in M} \bar{u}_i s_i \quad \geq BI + \epsilon$$

ou

$$- \sum_{j \in N^-} \bar{c}_j x_j + \sum_{j \in N^+} \bar{c}_j (1 - x_j) - \sum_{i \in M} \bar{u}_i s_i \quad \leq BS - (BI + \epsilon) \tag{3.2}$$

ϵ étant un nombre strictement positif. Dans le cas où la fonction objectif ne prend que des valeurs entières, alors ϵ peut être choisi égale à 1.

3.1.3 Propagation des coûts réduits

La programmation linéaire nous donne l'information nécessaire pour construire la contrainte (3.2) que nous utilisons dans une approche de programmation par contraintes. Cette contrainte est propagée en utilisant la consistance de borne. Nous illustrons cette procédure en utilisant l'exemple 1, issu de [94]. La table 3.1 donne les coefficients pour un PSDM de $n = 15$ variables et $m = 4$ contraintes.

j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
p_j	36	83	59	71	43	67	23	52	93	25	67	89	60	47	64	b_i
a_{1j}	7	19	30	22	30	44	11	21	35	14	29	18	3	36	42	87
a_{2j}	3	5	7	35	24	31	25	37	35	25	40	21	7	17	22	75
a_{3j}	20	33	17	45	12	21	20	2	7	17	21	11	11	9	21	65
a_{4j}	15	17	9	11	5	5	12	21	17	10	5	13	9	7	13	55

TAB. 3.1 – Exemple 1, PSDM issu de [94].

Exemple 1 Après avoir résolu la relaxation de PL, les coûts réduits correspondants sont :

$$\bar{c} = \{-24.4, 0, 0, -20.5, -10.7, -5.11, -43.2, -40.9, 0, -35.7, 0, 23.1, 22.4, -10.6, -24.4\}$$
$$\bar{u} = \{-0.66, -0.52, -0.62, -2.78\}$$

Ce problème a une solution optimale de valeur $BS = 335.6$. Supposons que nous connaissions une borne inférieure pour ce problème de valeur $BI = 301$. On a $BS-(BI+1) = 33.6$ et la contrainte de coûts réduits (3.2) donne :

$$\begin{aligned} &24.2x_1 + 20.5x_4 + 10.7x_5 + 5.11x_6 + 43.2x_7 + 40.9x_8 + 35.7x_{10} + 23.1(1 - x_{12}) + \\ &22.4(1 - x_{13}) + 10.6x_{14} + 24.4x_{15} + 0.66s_1 + 0.52s_2 + 0.62s_3 + 2.78s_4 \qquad \leq 33.6 \end{aligned}$$

La consistance de borne réduit les domaines des variables $\{x_7, x_8, x_{10}, s_1, s_2, s_3, s_4\}$, dont le domaine initial est $\{[0..1], [0..1], [0..1], [0..87], [0..75], [0..65], [0..55]\}$, comme suit :

$$x_7 \leq \lfloor \frac{33.6}{43.2} \rfloor \rightarrow x_7 \leq 0 \rightarrow x_7 = 0$$

$$x_8 \leq \lfloor \frac{33.6}{40.9} \rfloor \rightarrow x_8 \leq 0 \rightarrow x_8 = 0$$

$$x_{10} \leq \lfloor \frac{33.6}{35.7} \rceil \rightarrow x_7 \leq 0 \rightarrow x_{10} = 0$$

$$s_1 \leq \lfloor \frac{33.6}{0.66} \rfloor \rightarrow s_1 \leq 50$$

$$s_2 \leq \lfloor \frac{33.6}{0.52} \rfloor \rightarrow s_1 \leq 64$$

$$s_3 \leq \lfloor \frac{33.6}{0.62} \rfloor \rightarrow s_1 \leq 54$$

$$s_4 \leq \lfloor \frac{33.6}{2.78} \rfloor \rightarrow s_1 \leq 12$$

Nous pouvons constater que les coûts réduits du problème nous ont aidé à réduire les domaines de certaines variables. Ces informations peuvent être propagées aux autres contraintes du modèle afin d'obtenir plus de renseignements sur la consistance de la solution partielle obtenue. **Ce type de propagation est la stratégie de base** de coopération entre la PL et la PPC que nous proposons, en étendant la réduction de domaine aux variables d'écart. Cette procédure est décrite par l'algorithme 2.

Avant de poursuivre notre discussion, nous voulons faire quelques remarques sur l'approche proposée par rapport à celles d'Osorio et al. [94] et Focacci [31]. D'abord, il est intéressant d'observer que l'appariement de contraintes proposé par Osorio et al. [94] (voir section 2.4.3, page 26) revient à écrire la contrainte de coûts réduits (3.2) mais sans les variables d'écart. De plus, seuls les coûts réduits négatifs sont utilisés pour fixer les variables correspondantes à zéro. Enfin, cette information est utilisée pour générer des coupes logiques mais uniquement à la racine d'un arbre de recherche. L'approche originale de Focacci [31] consiste en l'utilisation des coûts réduits des variables originales comme une contrainte globale dans une approche de PPC. Nous proposons une approche plus pous-

Algorithme 2 Algorithme de filtrage par les coûts réduits

Soit y le vecteur $n+m$ des variables x_i et s_i.
Soit cr_{y_i} le coût réduit de la variable y_i.
Soit d_i (f_i) la borne inférieure (respectivement supérieure) de y_i.
Soit $div = (BS - (BI + \epsilon))/cr_{y_i}$.

```
1: pour i = 1 à n + m faire
2:   si d_i < f_i alors
3:     si cr_{y_i} > 0 alors
4:       d_i = max(d_i, ⌈f_i − div⌉)
5:     sinon si cr_{y_i} < 0 alors
6:       f_i = min(f_i, ⌊d_i + div⌋)
7:     Fin si
8:   Fin si
9: Fin pour
```

sée, en étendant la réduction des domaines aux variables d'écart et en fixant non seulement certaines variables à zéro mais aussi en fixant certaines variables à un. Notre approche est donc fondée sur l'algorithme 2

3.1.4 Méthode de recherche hybride PPC/PL

Dans le cadre d'une méthode de recherche arborescente, on peut simplement lancer l'algorithme 2 à chaque nœud avec l'inconvénient que les coûts réduits des nœuds précédents ne sont plus exploités. En fait, l'algorithme se contente de borner les variables à chaque nœud de l'arbre de séparation et évaluation, en fonction uniquement des coûts réduits courants. Or, les coûts réduits sont une information valide pour n'importe quel sous-problème et peuvent couper une solution si nous les combinons avec d'autres contraintes. Pour cette raison une première amélioration que nous proposons est de mettre explicitement la contrainte de coûts réduits (contrainte 3.2) dans le modèle PPC.

La figure 3.1 clarifie davantage le schéma de coopération PPC/PL et les modèles pris en compte par la PPC et la PL.

Enfin, nous proposons une procédure de séparation et évaluation, décrite comme suit : à chaque nœud de l'arbre de séparation et évaluation, la relaxation linéaire courante du problème est résolue. Si la borne supérieure est plus mauvaise que la valeur de la meilleure solution connue, c'est-à-dire, si le domaine de la fonction objectif devient vide, alors un retour en arrière (backtrack) est déclenché. Autrement, la contrainte de coûts réduits est générée et ajoutée au sous-problème courant dans le modèle PPC et propagée. Il peut alors se produire que les réductions de domaines induites par la propagation coupent la solution optimale continue de la relaxation linéaire. Dans ce cas, la relaxation linéaire, qui

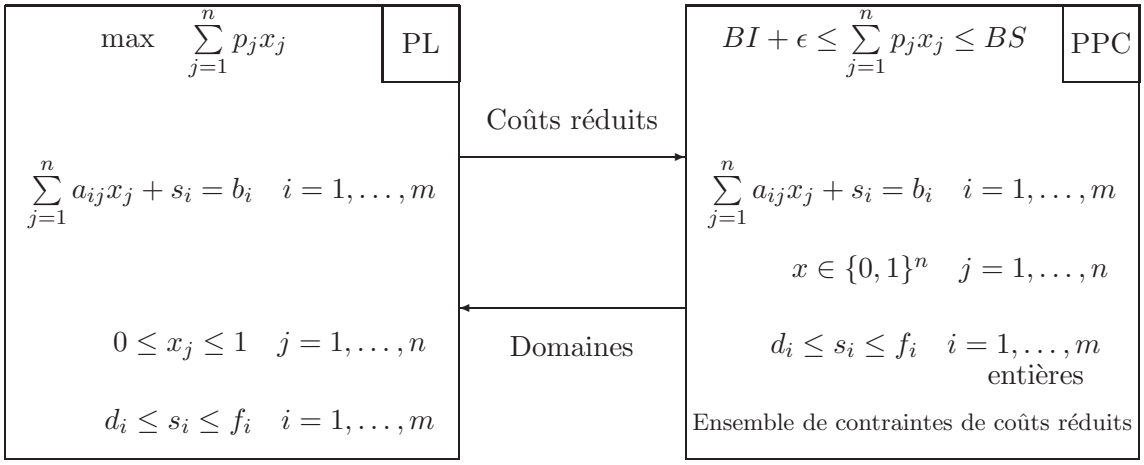

FIG. 3.1 – Schéma de coopération à un nœud de l'arbre de recherche

prend en compte les dernières réductions, est résolue une fois de plus et le processus peut se répéter jusqu'à ce qu'une solution entière soit trouvée ou bien que la solution optimale de la relaxation linéaire soit encore valide après la réduction des domaines.

Il est pas nécessaire d'ajouter la contrainte de coûts réduits au problème linéaire relâché puisqu'elle est par définition toujours vérifiée par la solution optimale continue courante.

Une fois que toutes les propagations et réductions de domaines ont été exécutées, nous devons sélectionner une variable pour brancher. Nous ne branchons que sur les variables binaires. Pour sélectionner la variable à brancher, plusieurs stratégies sont bien sur possibles, mais nous avons choisi de brancher sur la variable dont la valeur, dans la solution optimale continue est plus proche de 0,5. Cette procédure est illustrée dans l'algorithme 3.

Algorithme 3 Algorithme de recherche hybride PPC/PL.

1: Résoudre le Programme Linéaire PL. Si le problème linéaire est irréalisable ou si la solution est entière (mettre à jour la borne inférieure BI) alors faire un retour en arrière (backtrack).
2: Générer la contrainte de coûts réduits et l'ajouter au modèle PPC. Propager la contrainte de coûts réduits aux autres contraintes du modèle PPC. Si une solution unique est trouvée (mettre à jour la borne inférieure BI) ou si échec alors faire un retour en arrière (backtrack)
3: Si la solution de la PL est coupée alors mettre à jour le modèle PL avec les nouveaux domaines et révenir à 1.
4: Brancher sur la variable $x_i, i = 1, \ldots, m$ la plus fractionnaire

Il faut noter que le comportement par défaut du solveur HYBRID d'Ilog [61], voir section 1.6 n'utilise pas les variables d'écart pour la réduction de domaines et la contrainte de coûts réduits n'est pas prise en compte dans la pile de contraintes de SOLVER d'ILOG

[62]. Plus précisément, le solveur d'ILOG exploite les coûts réduits pour resserrer les bornes des variables originales. La méthode proposée dans cette section devrait donc être capable d'effectuer des réductions plus fortes.

Intuitivement, notre approche a deux effets immédiats sur la résolution : la première est que le nombre de contraintes du modèle PPC augmente, ce qui peut signifier une augmentation du temps de calcul. La deuxième est que la combinaison des contraintes de coûts réduits peut s'avérer intéressante pour obtenir des réductions de domaines plus fortes. Nous étudierons le comportement de la méthode sur des résultats expérimentaux dans la section 3.4.

3.2 Améliorations

Dans le schéma de recherche hybride de la section 3.1.4 nous proposons différentes améliorations ou variantes de la contrainte de coûts réduits.

3.2.1 Les coupes de Gomory

Nous montrons dans cette section que la contrainte de coûts réduits est étroitement liée à la coupe de Gomory [86] qui peut être dérivée de la fonction objectif.

Nous avons déjà vu, à la page 37, que la fonction objectif de (P) peut être écrite comme :

$$z = BS + \sum_{j \in N^-} \bar{c}_j x_j - \sum_{j \in N^+} \bar{c}_j (1 - x_j) + \sum_{i \in M} \bar{u}_i s_i$$

ou

$$z - \sum_{j \in N^-} \bar{c}_j x_j + \sum_{j \in N^+} \bar{c}_j (1 - x_j) - \sum_{i \in M} \bar{u}_i s_i = BS$$

si $f_j = -\bar{c}_j - \lfloor \bar{c}_j \rfloor \quad \forall j \in N^-$, $f_j = \bar{c}_j - \lfloor \bar{c}_j \rfloor \quad \forall j \in N^+$ et $f_0 = BS - \lfloor BS \rfloor$ alors la coupe de Gomory correspondant à cette égalité est :

$$z + \sum_{f_j \leq f_0, j \in N^-} \lfloor -\bar{c}_j \rfloor x_j + \sum_{f_j \leq f_0, j \in N^+} \lfloor \bar{c}_j \rfloor (1 - x_j) + \sum_{f_i \leq f_0, i \in M} \lfloor -\bar{u}_i \rfloor s_i +$$

$$\sum_{f_j>f_0, j\in N^-} (\lfloor -\bar{c_j} \rfloor + \frac{f_j - f_0}{1-f_0}) x_j + \sum_{f_j>f_0, j\in N^+} (\lfloor \bar{c_j} \rfloor + \frac{f_j - f_0}{1-f_0})(1-x_j) + \sum_{f_j>f_0, i\in M} (\lfloor -\bar{u_i} \rfloor + \frac{f_i - f_0}{1-f_0}) s_i \quad \leq \lfloor BS \rfloor$$

En retournant à l'exemple 1, de la section 3.1.2, la coupe de gomory devient :

$$z + 24x_1 + 20x_4 + 10.088x_5 + 5x_6 + 43x_7 + 40.72x_8 + 35.3x_{10} + 23(1-x_{11}) + 22(1-x_{12}) + 10x_{14} + 24x_{15} + 0.091s_1 + 0.007s_3 + 2.432s_4 \leq 335$$

Puisque z est la fonction objectif alors l'algorithme de la borne-consistance substituera z par la valeur de son domaine la plus petite, c'est-à-dire, $BI + \epsilon$. Maintenant, puisque $z \geq 302$, nous pouvons déduire que :

$$24x_1 + 20x_4 + 10.088x_5 + 5x_6 + 43x_7 + 40.72x_8 + 35.3x_{10} + 23(1-x_{11}) + 22(1-x_{12}) + 10x_{14} + 24x_{15} + 0.091s_1 + 0.007s_3 + 2.432s_4 \leq 33$$

Cette contrainte est proche de la contrainte de coûts réduits. Cependant, il faut souligner qu'elle permet de fixer que les variables x_7, x_8, x_{10} et de déduire que $s_4 \leq 12$ (c'est-à-dire qu'aucune déduction ne peut être faite sur s_1, s_2 et s_3).

Le principal avantage de générer et propager cette coupe, plutôt que la contrainte de coûts réduits, est qu'elle peut être ajoutée à la relaxation de programmation linéaire, puisqu'elle coupe la solution optimale courante tant que la borne supérieure courante n'est pas entière. Donc, cette coupe devrait entraîner des réductions de domaines plus profondes que la contrainte de coûts réduits. En outre, elle améliore aussi la borne supérieure. D'autre part, puisqu'une contrainte (ou plus) de ce type peut être ajoutée à chaque nœud, le problème linéaire à résoudre possède un grand nombre de contraintes en bas de l'arbre de séparation et évaluation. Dans le cadre de la procédure présentée dans l'algorithme 3, nous pouvons ajouter la contrainte de gomory à l'étape 2 et à l'étape 3 dans la PL.

3.2.2 La contrainte reverse de coûts réduits

Dans l'exemple 1, nous pouvons observer que la contrainte de coûts réduits permet de fixer certaines variables à leurs valeurs optimales courantes du problème relâché. C'est le cas pour $\{x_7, x_8, x_{10}\}$ qui sont fixées à $\{0, 0, 0\}$. Toutefois, la solution optimale relâchée peut être très différente de la solution optimale entière. Considérons l'exemple 2, où s_1 et s_2 sont variables d'écart.

Exemple 2

$$\begin{aligned} \text{Max} \quad & 43x_1 + 10x_2 + 18x_3 + 12x_4 + 36x_5 + 22x_6 \\ \text{s.à.} \quad & 12x_1 + 2x_2 + 3x_3 + 2x_4 + 4x_5 + 3x_6 + s_1 = 20 \\ & 3x_1 + 8x_2 + 12x_3 + 13x_4 + 20x_5 + 14x_6 + s_2 = 36 \\ & x \in \{0,1\}^6 \end{aligned}$$

Le problème relâché mène à la solution optimale continue $\bar{x} = (1, 0, 0, 0, 1, 0.93)$ et $\bar{s} = (1.21, 0)$ avec une valeur de la fonction objectif égale à $z(\bar{P}) = 99.43$ alors que la solution optimale entière est $x = (1, 0, 1, 0, 1, 0)$ et $s = (1, 1)$. Nous observons que x_3 dans la solution relâchée est égale à sa borne inférieure tandis que dans la solution optimale entière elle est égale à sa borne supérieure. Il est alors légitime de penser que la maximisation de la fonction objectif guide vers des régions du domaine réalisable où il n'y a pas de solutions entières. Pour limiter ces "erreurs" de guidage, il est en fait possible d'utiliser les coûts réduits comme nous l'avons fait à la section précédente, mais en utilisant une inégalité inverse, c'est-à-dire en bornant l'expression de la fonction objectif, en terme de coûts réduits, par une borne BS_s meilleure que celle fournie par la programmation linéaire. On pourra alors propager cette contrainte "reverse" de coûts réduits pour en déduire de nouvelles réductions de domaine. Ainsi, la contrainte reverse de coûts réduits s'écrit :

$$BS + \sum_{j \in N^-} \bar{c}_j x_j - \sum_{j \in N^+} \bar{c}_j (1 - x_j) + \sum_{i \in M} \bar{u}_i s_i \quad \leq BS_s \tag{3.3}$$

ou

$$BS - BS_s \leq \quad - \sum_{j \in N^-} \bar{c}_j x_j + \sum_{j \in N^+} \bar{c}_j (1 - x_j) - \sum_{i \in M} \bar{u}_i s_i \tag{3.4}$$

En pratique, une borne possible est celle de la borne surrogate [94], voir section 2.3,

page 23.

Supposons que pour l'exemple 2, nous connaissions une solution réalisable de valeur $BI = 97$. Toujours pour cet exemple, la borne supérieure surrogate est égale à $BS_s = 97$ (la borne de programmation linéaire est égale à $BS = 99.43$). Cependant, pour illustrer l'intérêt de la contrainte de coûts réduits réversible, supposons que nous ayons une borne supérieure moins bonne égale à $BS'_s = 98$. Les coûts réduits sont donnés au tableau 3.2.

j	1	2	3	4	5	6	s_1	s_2
$\bar{c_j}$	38.29	-2.57	-0.86	-8.43	4.57	0	0	-1.57

TAB. 3.2 – Coûts réduits associés aux variables de l'Example 2

La contrainte reverse de coûts réduits, comme définie dans (3.4), est,

$$99.43 - 98 \leq 38.29(1 - x_1) + 2.57x_2 + 0.86x_3 + 8.43x_4 + 4.57(1 - x_5) + 1.57s_2$$

et la contrainte de coûts réduits est (pour $BI = 97$) :

$$38.29(1 - x_1) + 2.57x_2 + 0.86x_3 + 8.43x_4 + 4.57(1 - x_5) + 1.57s_2 \leq 99.43 - 97$$

En propageant la contrainte de coûts réduits nous obtenons $\{(x_1 = 1), (x_2 = 0), (x_3 = [0..1]), (x_4 = 0), (x_5 = 1), (s_2 = [0..1])\}$. La contrainte reverse de coûts réduits devient alors :

$$1.43 \leq 0.86x_3 + 1.57s_2$$

Nous pouvons en déduire que $s_2 = 1$ (alors que dans la solution optimale relâchée, nous avons $s_2 = 0$). Ceci implique que la solution optimale continue courante est coupée, alors nous relançons la PL avec cette nouvelle information. Enfin, dans le cadre de la procédure présentée dans l'algorithme 3 nous générons et ajoutons les contraintes de coûts réduits et la contrainte reverse de coûts réduits dans le modèle PPC à l'étape 2.

3.2.3 Ensemble de Contraintes Logiques

Une autre variante à la stratégie de base que nous proposons est fondée sur le fait que la contrainte de coût réduit nous aide à identifier un ensemble minimal de contraintes logiques où au moins une doit être satisfaite. Par souci de clarté, ci-dessous nous récrivons la contrainte de coûts réduits.

$$-\sum_{j\in N^-}\bar{c}_j x_j + \sum_{j\in N^+}\bar{c}_j(1-x_j) - \sum_{i\in M}\bar{u}_i s_i \quad \leq BS-(BI+\epsilon) \tag{3.5}$$

Supposons que les variables soient triées selon l'ordre décroissant des valeurs absolues des coûts réduits. Il est alors possible d'identifier une ensemble de taille minimale de contraintes dont une, au moins, doit être active à l'optimum. Supposons par exemple avoir la contrainte de coûts réduits :

$$2(1-x_1)+2s+x_2+x_3+x_4+x_5 \leq 3 \tag{3.6}$$

Il est clair qu'il est impossible avoir $(x_1 = 0 \wedge s \geq 1)$, sans violer la contrainte. Donc, une solution meilleure que la solution courante doit respecter $\neg(x_1 = 0 \wedge s \geq 1)$. Ceci est totalement équivalent à $(x_1 = 1 \vee s = 0)$ qui est, finalement, l'ensemble minimal de contraintes logiques où au moins une doit être satisfaite. Cet ensemble de contraintes peut être propagé de différentes manières. Nous avons choisi de propager séparément chacune des contraintes de l'ensemble et de tester la consistance de celle-ci.

Dans le cadre de la procédure présentée dans l'algorithme 3, cette variante introduit une nouvelle étape, disons 3b, décrite dans l'algorithme 4

Algorithme 4 Étape 3b de l'algorithme 3

La procédure Valide(C) ajoute une contrainte C au modèle PPC et la propage en testant s'il y a un échec. Elle retire C du modèle et restaure les domaines des variables. Elle envoie comme résultat VRAI s'il n'y a pas eu d'échec et FAUX dans le cas contraire.

1: Générer un ensemble E de contraintes logiques de taille K.
2: **pour** chaque contrainte $E_k \in E$ **faire**
3: **si** !Valide(E_k) **alors**
4: ajouter $\neg E_k$ au modèle PPC et au modèle PL. Revenir à l'étape 1 de la procédure proposée dans l'algorithme 3.
5: **Fin si**
6: **Fin pour**

Par souci de clarté, nous illustrons cette stratégie à l'aide de l'exemple 3.

Exemple 3 Soit le Sac à Dos unidimensionnel suivant :

$$\begin{aligned} (P) \qquad Max \quad & 20x_1 + 16x_2 + 11x_3 + 9x_4 + 7x_5 + x_6 \\ \text{s.à} \quad & 9x_1 + 8x_2 + 6x_3 + 5x_4 + 4x_5 + x_6 \leq 12 \\ & x \in \{0,1\}^6 \end{aligned}$$

qui sous la forme standard devient :

$$\begin{aligned} (P) \qquad Max \quad & 20x_1 + 16x_2 + 11x_3 + 9x_4 + 7x_5 + x_6 \\ \text{s.à} \quad & 9x_1 + 8x_2 + 6x_3 + 5x_4 + 4x_5 + x_6 + s = 12 \\ & x \in \{0,1\}^6 \\ & s \in [0,12] \end{aligned}$$

Les modèles initiaux correspondants sont illustrés dans la figure 3.2.

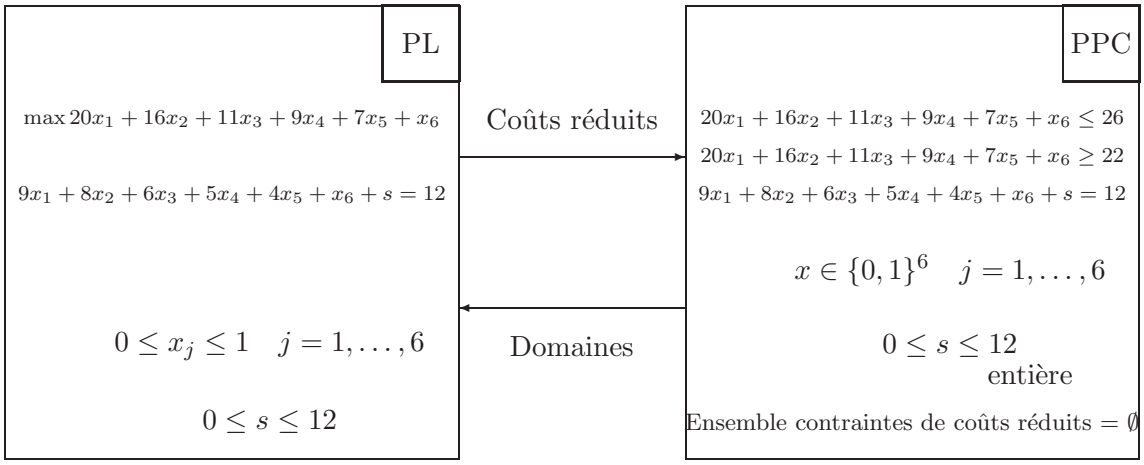

FIG. 3.2 – Modèles PL et PPC pour l'exemple 3

Une solution réalisable pour ce problème est $x = (1, 0, 0, 0, 0, 1)$ avec une valeur de la fonction objectif égale à $BI = 21$. Nous souhaitons trouver des solutions meilleures, c'est-à-dire qui aient une valeur de la fonction objectif plus grande que 21, disons $BI + \epsilon = 22$. En suivant la procédure décrite dans l'algorithme 3 auquel on a ajouté l'étape 3b dans l'algoritme 4 on obtient :

Étape 1

En résolvant le problème relâché de (P) , nous obtenons le vecteur des coûts réduits $\bar{c}$ associé aux variables x. $\bar{c} = (2, 0, -1, -1, -1, -1)$ et le coût réduit associé à la variable d'écart s $\bar{u} = (-2)$. La valeur optimale du problème relâché est $BS = 26$ et c'est aussi

une borne supérieure pour le problème original. La solution optimale continue est $\bar{x} = (1, 3/8, 0, 0, 0, 0)$ et $\bar{s} = 0$.

Étape 2 Nous générons la contrainte de coûts réduits qui est illustrée en (3.7).

$$2(1 - x_1) + 0x_2 + x_3 + x_4 + x_5 + x_6 + 2s \leq 4 \tag{3.7}$$

Ensuite, nous la propageons en obtenant uniquement la réduction de domaine de la variable d'écart $s \leq 2$. Ceci est totalement équivalent à :

$$10 \leq 9x_1 + 8x_2 + 6x_3 + 5x_4 + 4x_5 + x_6 \leq 12$$

Étape 3

La solution optimale continue n'est pas coupée, alors on continue avec l'étape 3b.

Étape 3b

À partir de (3.7) nous pouvons déduire un ensemble minimal de contraintes. Par exemple, $x_1 = 0 \wedge s \geq 2$ n'est pas possible. Alors, au moins une des contraintes $x_1 = 1 \vee s \leq 1$ doit être satisfaite.

Soit $E = \{x_1 = 1, s \leq 1\}$.

Nous utilisons l'algorithme 4 pour valider ces contraintes dans le modèle PPC.

Valide($x_1 = 1$) implique dans le modèle PPC

$$16x_2 + 11x_3 + 9x_4 + 7x_5 + x_6 \leq 6 \tag{3.8}$$

$$16x_2 + 11x_3 + 9x_4 + 7x_5 + x_6 \geq 2 \tag{3.9}$$

que les domaines des variables x_2, x_3, x_4, x_5 deviennent bornés à la valeur 0 de la contrainte (3.8). Cependant le domaine associé à la variable x_6 devient vide, aucune valeur du domaine de x_6 (0,1) satisfait la contrainte (3.9). par conséquence, la valeur renvoyée par

Valide($x_1 = 1$) est FAUX. Donc, la valeur de x_1 doit être zéro pour toute solution plus grande ou égale a 22. Alors, nous posons $x_1 = 0$, $D_{x_1} = \{0\}$ au modèle de PL et au modèle PPC, respectivement. Nous revenons à l'étape 1 de l'algorithme 3.

Étape 1

En résolvant le problème relâché de (P) (avec la nouvelle contrainte valide $x_1 = 0$, en utilisant une technique de prétraitement nous l'éliminons du problème), nous obtenons le vecteur des coûts réduits $\bar{c}$ associé aux variables x. $\bar{c} = (-, -4/3, 0, 1/6, 1/3, 5/6)$ et le coût réduit associé à la variable d'écart s $\bar{u} = (11/6)$. La valeur optimale du problème relâché est $BS = 23.3$ et c'est aussi une borne supérieure pour le problème original. La solution optimale continue est $\bar{x} = (0, 1, 2/3, 0, 0, 0)$ et $\bar{s} = 0$.

Étape 2 Nous générons la contrainte de coûts réduits qui est illustrée en (3.10).

$$\tfrac{4}{3}(1 - x_2) + \tfrac{1}{6}x_4 + \tfrac{1}{3}x_5 + \tfrac{5}{6}x_6 + \tfrac{11}{6}s \leq \tfrac{4}{3} \tag{3.10}$$

Ensuite, nous la propageons en obtenant uniquement la réduction de domaine de la variable d'écart $s \leq 0$.

Ceci est totalement équivalent à :

$$8x_2 + 6x_3 + 5x_4 + 4x_5 + x_6 = 12 \tag{3.11}$$

Étape 3

La solution optimale continue n'est pas coupée, alors on continue avec l'étape 3b.

Étape 3b

À partir de (3.10) nous pouvons déduire un ensemble minimal de contraintes. Par exemple, $x_2 = 0 \wedge x_6 = 1$ n'est pas possible. Alors, au moins une des contraintes $x_2 = 1 \vee x_6 = 0$ doit être satisfaite.

Soit $E = \{x_2 = 1, x_6 = 0\}$.

Nous utilisons l'algorithme 4 pour valider ces contraintes dans le modèle PPC.

Valide($x_2 = 1$) implique dans le modèle PPC

$$11x_3 + 9x_4 + 7x_5 + x_6 \leq (23 - 16) = 7 \tag{3.12}$$

$$11x_3 + 9x_4 + 7x_5 + x_6 \geq (22 - 16) = 6 \tag{3.13}$$

En propageant la contrainte (3.12) on obtient une réduction des domaines des variables x_3, x_4. En fait, ces variables sont affectées aux valeurs $(0, 0)$, respectivement. En propageant ces informations à la contrainte (3.13) on obtient une nouvelle réduction de domaine, cette fois sur la variable x_5 dont sa valeur est 1. Ces valeurs (réductions des domaines) sont propagées à la contrainte (3.11), d'où on obtient une nouvelle information, la réduction du domaine de la variable x_6 dont sa valeur est 0. En résumé, nous avons trouvé une nouvelle solution entière $x = (0, 1, 0, 0, 1, 0)$ dont la valeur de la fonction objectif est $16 + 7 = 23$. On met à jour la borne inférieure $BI = 23$. Comme la borne supérieure $BS = 23$ est égale à la borne inférieure $BI = 23$, alors l'algorithme s'arrêt puisque nous avons trouvé la solution optimale $x = (0, 1, 0, 0, 1, 0)$ de valeur 23. Il faut noter que nous avons trouvé la solution optimale à la racine de l'arbre de recherche (mais en résolvant plusieurs PL).

3.2.4 Amélioration de la stratégie de branchement et renforcement de contraintes

Pour résoudre le PSDM, certains auteurs (notamment Vasquez et Hao [120]) diminuent la borne supérieure obtenue par programmation linéaire en considérant différents sous-problèmes selon le nombre de variables x_j devant être à 1. Chacun de ces sous-problèmes est déduit du problème initial en lui ajoutant une contrainte de la forme

$$\sum_{j=1}^{n} x_j = k \text{ avec } k \in \mathbb{N}$$

Les valeurs de k intéressantes sont calculées en résolvant les deux modèles linéaires suivants :

$$\begin{aligned} (Max_k) \qquad \max \quad & \sum_{j=1}^{n} x_j \\ \text{s.à.} \quad & \sum_{j=1}^{n} a_{ij} x_j \leq b_i \quad i = 1, \ldots, m \\ & 0 \leq x_j \leq 1 \quad j = 1, \ldots, n \end{aligned}$$

$$
\begin{aligned}
(Min_k) \qquad \min \quad & \sum_{j=1}^{n} x_j \\
\text{s.à.} \quad & \sum_{j=1}^{n} a_{ij} x_j \leq b_i \quad i = 1, \ldots, m \\
& \sum_{j=1}^{n} c_j xj \geq BI \\
& 0 \leq x_j \leq 1 \qquad j = 1, \ldots, n
\end{aligned}
$$

où la borne supérieure de k est égale à $\gamma = \lfloor Max_k \rfloor$ et la borne inférieure de k est égale à $\kappa = \lceil Min_k \rceil$. Donc, le nombre de sous-problèmes à résoudre est égale à $\gamma - \kappa + 1$.

Pour illustrer l'avantage d'utiliser la contrainte (3.14) nous reprenons l'exemple 1 issu de [94], voir table 3.1.

Pour ce problème $\gamma = \kappa = 4$. Par conséquence, il faut juste résoudre un seul problème avec la contrainte (3.14). En résolvant le modèle linéaire avec la contrainte :

$$\sum_{j=1}^{15} x_j = 4 \tag{3.14}$$

Après avoir résolu la relaxation de PL, les coûts réduits correspondants sont :

$\bar{c} = \{-24.2, 20.7, 1.8, 0, -21.2, 0, -42.4, -14.0, 26.8, -40.1, -2.1, 25.7, 0, -15.1, -0.9\}$

$\bar{u} = \{0, -0.23, -0.13, 0\}$

Ce problème a une solution optimale de valeur $BS = 327.1$. Nous connaissions une borne inférieure pour ce problème de valeur $BI = 301$. On a $BS - (BI + 1) = 25.1$ et la contrainte de coûts réduits (3.2) donne :

$$24.2x_1 + 20.7(1 - x_2) + 1.8x_3 + 21.2x_5 + 42.4x_7 + 14.0x_8 + 26.8(1 - x_9) + 40.1x_{10} + 2.1x_{11} + 25.7(1 - x_{12}) + 15.1x_{14} + 0.9x_{15} + 0.23s_2 + 0.13s_3 \leq 25.1$$

En propageant cette contrainte on obtient, en plus des réductions de domaines illustrées à la section 3.1.3, $x_9 = 1$ et $x_{12} = 1$. Ceci est très important parce qu'on a pu fixer des variables à 1, ce qui est, en fait, le plus difficile.

3.3 Schéma d'hybridation PSDM

Dans les expérimentations numériques de la section 3.4.3 et selon les résultats préliminaires montrés dans la section 3.4.2, nous avons adopté la stratégie décrite dans l'algorithme 3 avec les variante décrites dans l'algorithme 4 et dans la section 3.2.4. Ainsi, dans le premier niveau de l'arbre de séparation et évaluation nous avons $\gamma - \kappa + 1$ nœuds à traiter. Nous résolvons la relaxation de programmation linéaire de chaque sous-problème en obtenant de chacun d'eux une borne supérieure égale à, disons BS_k, $k = \kappa, \ldots, \gamma$. Ces bornes sont triées selon l'ordre décroissant. Nous choisisons le nœud à traiter comme celui qui a la borne supérieure la plus grande. La justification de ce choix provient du fait que, si une solution entière est trouvée il y a de forte chances que la valeur de cette solution entière est plus grande que la borne supérieure d'un sous-problème (nœud) pas encore traité. Alors, la branche correspondante à ce nœud est élaguée.

Dans la section 3.4.2 nous avons testé séparement les stratégies de : contraintes de coûts réduits, coupes de gomory, contraintes de coûts réduits et contrainte reverse de coûts réduits, et contrainte de coûts réduits avec la variante d'ensemble de contraintes logiques.

Dans ce qui suit nous présentons les résultats obtenus en utilisant les diverses stratégies d'hybridation.

3.4 Résultats Expérimentaux.

Nous avons testé notre approche en utilisant deux types d'expériences. Pour la première expérience nous avons utilisé de petits problèmes de la vie réel déjà publiés dans la littérature et disponibles dans "OR-Library[2]". Pour la deuxième expérience, nous avons utilisé l'ensemble de 270 problèmes de grande taille développé par Beasley[28] et aussi disponible dans "OR-library". Ces instances sont organisées comme suit :

- 90 instances avec 100 variables : 30 instances avec $m = 5$ contraintes, 30 avec $m = 10$ et 30 avec $m = 30$.
- 90 instances avec 250 variables : 30 instances avec $m = 5$ contraintes, 30 avec $m = 10$ et 30 avec $m = 30$.
- 90 instances avec 500 variables : 30 instances avec $m = 5$ contraintes, 30 avec $m = 10$ et 30 avec $m = 30$.

[2]http ://mscmga.ms.ic.ac.uk/

Chaque ensemble de 30 instances est divisé en 3 séries avec $\alpha = b_i / \sum_{j=1}^{n} a_{ij} = 1/4$, $\alpha = 1/2$ et $\alpha = 3/4$.

Toutes les procédures ont été développée sur un PentiumII cadencé à 350MHz et possédant 256 Mo de RAM. Le système d'exploitation est Windows 98 DE, le langage orienté objet est le C++, compilé avec Microsoft Visual C++ 6.0. Pour la modélisation du PSDM, nous avons utilisé la librairie d'ILOG CONCERT version 1.1 qui sert à la modélisation des programmes linéaires et fait office d'interface avec HYBRID version 1.1 d'ILOG pour l'optimisation de programmes linéaires en nombres entiers, mixtes ou continus et qui utilise les solveurs CPLEX version 7.1 et SOLVER version 5.1 mais sans pouvoir profiter de toutes les fonctionnalités de deux solveurs, par exemple les coupes de CPLEX.

3.4.1 Bornes Inférieures Initiales.

Les tableaux 3.3 et 3.4 illustrent les résultats obtenus par l'heuristique de Magazine et Oguz [78] (colonne $v1^1$) et ses différentes versions $v2^2, \ldots, v6^6$ pour les 270 instances décrites ci-dessus. Le tableau 3.3 illustre les résultats sans résoudre exactement le sous-problème qui reste et le tableau 3.4 montre les résultats en résolvant le sous-problème qui reste (voir section 3.1.1. Plus particulièrement, $vi^i, i = 1, \ldots, 5$ désigne les résultats obtenus en utilisant une heuristique qui change i multiplicateurs à la fois où $v2^2$ représente l'heuristique de Volgenant et Zoon [123]. $v6^6$ représente une méthode qui change dynamiquement les multiplicateurs de lagrange à la fois. La colonne (7) illustre les résultats obtenus pour chaque instance des 270. La ligne nommée "Nbresf" illustre le nombre de fois où l'heuristique est la meilleure parmi les autres. La ligne "Écart Moyen" montre écarts moyens entre la meilleure solution connue et la solution obtenue par l'heuristique vi^i. La ligne "Écart Max" ("Écart Min") illustre l'écart maximal (minimal) pour les 270 instances entre la meilleure solution connue et celle de l'heuristique, respectivement.

Nous voyons que l'heuristique représentée par $v6E^6$ domine les autres variantes. Finalement, dans notre algorithme exacte l'heuristique choisie pour nous fournir une "bonne" solution est la variante représentée par la colonne (6) du tableau 3.4, c'est-à-dire un changement dynamique de multiplicateurs avec une résolution exacte du sous-problème qui reste à résoudre plutôt que d'insérer de variables d'une manière gloutonne.

	v1$^{(1)}$	v2$^{(2)}$	v3$^{(3)}$	v4$^{(4)}$	v5$^{(5)}$	v6$^{(6)}$	$\max_{i=1,\ldots,6} V_i^{(7)}$
Nbresf	4	41	31	35	35	124	
Écart Moyen	5637.3	3723.3	3605.2	3979.0	2811.3	2436.4	1913.1
Écart Max	14917	11440	11866	17222	9048	8009	7422
Écart Min	601	116	29	322	221	221	29

(1)Heuristique de Magazine et Oguz [78]
(2)Heuristique de Volgenant et Zoon [123]
(3),(4),(5) Des variations de l'heuristique originale avec 3, 4, 5 multiplicateurs qui changent à la fois, respectivement
(6) Ajustement dynamique du nombre de multiplicateur à changer d'une itération à l'autre.
(7) La meilleure heuristique pour chaque instance des 270 problèmes issu de Chu et Beasley [28]

Tab. 3.3 – Résultats de l'heuristique avec plusieurs multiplicateurs mais sans la résolution exacte du sous-problème

	v1E$^{(1)}$	v2E$^{(2)}$	v3E$^{(3)}$	v4E$^{(4)}$	v5E$^{(5)}$	v6E$^{(6)}$	$\max_{i=1,\ldots,6} V_i^{(7)}$
Nombre de fois que la heuristique est la meilleure	1	28	23	41	32	145	
Écart Moyen	5396.8	3368.1	3085.5	3066.9	2206.6	1709.6	1336
Écart Max	14867	11244	10727	12176	8651	6432	5788
Écart Min	129	31	29	166	0	0	0

(1)Heuristique de Magazine et Oguz [78]
(2)Heuristique de Volgenant et Zoon [123]
(3),(4),(5) Des variations de l'heuristique originale avec 3, 4, 5 multiplicateurs qui changent à la fois, respectivement
(6) Ajustement dynamique du nombre de multiplicateur à changer d'une itération à l'autre.
(7) La meilleure heuristique pour chaque instance des 270 problèmes issu de Chu et Beasley [28]

Tab. 3.4 – Résultats de l'heuristique avec plusieurs multiplicateurs et la résolution exacte du sous-problème

3.4.2 Résultats sur de petits problèmes.

La première expérience pour tester les différentes stratégies de propagation a été réalisée sur des petits problèmes de la vie réelle. Le nombre de variables est compris entre $n = 6$ à $n = 105$ et le nombre de contraintes entre $m = 2$ jusqu'à $m = 30$ et les valeurs des solutions optimales sont connues. Le tableau 3.5 illustre les caractéristiques de ces 54 instances de petite taille.

Nom du prob.	Var	Ncontr	z_{opt}	Nom du prob.	Var	Ncontr	z_{opt}
Petersen 1	15	10	4015	WEISH 12	50	5	6339
Petersen 2	6	10	3800	WEISH 13	50	5	6159
Petersen 3	20	10	6120	WEISH 14	60	5	6954
Petersen 4	28	10	12400	WEISH 15	60	5	7486
Petersen 5	39	5	10618	WEISH 16	60	5	7289
Petersen 6	50	5	16537	WEISH 17	60	5	8633
SENTO 1	60	30	7772	WEISH 18	70	5	9580
SENTO 2	60	30	8722	WEISH 19	70	5	7698
WEING 1	28	2	141278	WEISH 20	70	5	9450
WEING 2	28	2	130883	WEISH 21	70	5	9074
WEING 3	28	2	95677	WEISH 22	80	5	8947
WEING 4	28	2	119337	WEISH 23	80	5	8344
WEING 5	28	2	98796	WEISH 24	80	5	10220
WEING 6	28	2	130623	WEISH 25	80	5	9939
WEING 7	105	2	1095445	WEISH 26	90	5	9584
WEING 8	105	2	624319	WEISH 27	90	5	9819
WEISH 1	30	5	4554	WEISH 28	90	5	9492
WEISH 2	30	5	4536	WEISH 29	90	5	9410
WEISH 3	30	5	4115	WEISH 30	90	5	11191
WEISH 4	30	5	4561	PB 1	27	4	3090
WEISH 5	30	5	4514	PB 2	34	4	3186
WEISH 6	40	5	5557	PB 3	29	2	95168
WEISH 7	40	5	5567	PB 4	20	10	2139
WEISH 8	40	5	5605	PB 5	40	30	776
WEISH 9	40	5	5246	PB 6	37	30	1035
WEISH 10	50	5	6339	HP 1	28	4	3418
WEISH 11	50	5	5643	HP 2	35	4	3186

TAB. 3.5 – Caractéristiques des problèmes de petite taille

Nous avons résolu ces problèmes à l'aide de la librairie d'ILOG HYBRID 1.1 et nous les avons comparés avec ILOG CPLEX 7.1 par défaut (c'est-à-dire avec les coupes incluses).

Les résultats comparatifs sont illustrés à l'aide du tableau 3.6. Cinq colonnes sont présentées. La première indique le type de stratégie utilisée ; la deuxième le nombre de solutions optimales trouvées à la racine de l'arbre de recherche ; la troisième indique le temps moyen en secondes pour résoudre une des instances présentées dans le tableau 3.5 ; la

quatrième colonne illustre le nombre moyen d'itérations du simplexe obtenu pour résoudre toutes les instances ; enfin, la colonne 5 indique le nombre moyen de nœuds parcourus dans l'arbre de recherche.

Stratégie(1)	NbreSolOpt(2)	Temps(sec)(3)	Nbre moyen(4) d'itérations Simplexe	Taille moyenne (5) de l'arbre de recherche
CPLEX 7.1	8	0,26	240,56	60.46
HYBRID 1.1	0	0,47	364,70	88,85
Méthode de base Algorithme 2	0	0,35	242,18	42
Cont. Coût Red.	3	0,33	217,74	33,33
Coupes Gomory	2	0,51	352,57	37,15
CCR et CCRR	3	0,41	216,75	32,39
Cont. Logiques	5	0,57	289,63	31,96

(1) Différentes stratégies de résolution.
(2) Le nombre de solutions optimales trouvées et prouvées à la racine d'un arbre de recherche
(3) Le temps moyen, en secondes, nécessaire pour résoudre une des 54 instances décrites au tableau 3.5
(4) Le nombre moyen d'itérations nécessaire pour résoudre une des 54 instances décrites au tableau 3.5
(5) Taille moyenne de l'arbre de séparation et évaluation pour résoudre une des 54 instances décrites au tableau 3.5

TAB. 3.6 – Comparaison des différentes stratégies pour résoudre le PSDM de la "OR-Library" .

Regardons le tableau 3.6 et rappelons que la borne inférieure initiale est égale à la valeur de la solution optimale pour tous les problèmes. La deuxième colonne montre que CPLEX 7.1 est supérieur aux autres stratégies. Néanmoins, il convient de souligner que les apoproches hybrides que nous proposons sont assez sensiblement plus performantes en terme d'itérations du simplexe et de la taille de l'arbre de recherche. Ceci suggère quelques commentaires : d'une part, il est permis d'espérer que, pour des problèmes plus difficiles où la taille de l'arbre de recherche de CPLEX serait très grande, nous pourrions être compétitif en terme de cpu également. De plus, la colonne (2) nous suggère que les coupes générées par CPLEX sont très efficaces. Il serait sans doute intéressant, mais techniquement impossible (CPLEX ne donne aucun accès aux coupes qu'il génère), de considérer ces coupes comme les autres contraintes dans une approche hybride.

Dans les expérimentations de la section 3.4.3, portant sur des problèmes de grande taille, nous avons retenu la méthode hybride consistant à n'introduire que la contrainte de coûts réduits.

3.4.3 Résultats pour les problèmes de grande taille.

Pour la deuxième expérience, nous avons choisi l'ensemble de problèmes tests utilisé par Chu et Beasley[28] et disponible via web. Ces données ont été générées en utilisant la procédure suggérée par Fréville et Plateau[36]. Le nombre de contraintes m a été fixé à 5 et 10 et le nombre de variables n a été fixé à 100. Trente problèmes ont été générés pour chaque combinaison, en donnant un total de 60 instances.

Bien malheureusement, ces expérimentations sont loin de conforter nos espoirs quant au comportement de notre méthode relativement à celui de CPLEX. Au contraire, les phénomènes constatés à la section précédente se trouvent amplifiés : CPLEX est beaucoup plus rapide en dépit d'un arbre de recherche et d'un nombre d'itérations du simplexe largement supérieurs. Cela laisse à penser que l'ajout de la PPC à la PL pour résoudre ce type de problème ne permet de couper que des nœuds de bas de l'arbre de recherche. Ainsi, propager les contraintes dans le haut de l'arbre ne représente qu'une perte de temps.

Nous pensons cependant que certaines pistes restent à envisager : mieux propager la contrainte "ensemble", développer des schémas de branchement plus adaptés à la PPC, etc.

De plus, il est possible que sur d'autres types de problèmes les approches présentées dans ce chapitre soient plus performantes que CPLEX : nous pensons en particulier aux problèmes où les méthodes de résolution présentées dans la littérature sont, en proportions équivalentes, soit de type programmation mathématique, soit de type programmation par contraintes (l'ordonnancement par exemple).

Stratégie(1)	Temps(sec)(2)	Nbre moyen(3) d'itérations Simplexe	Taille moyenne (4) de l'arbre de recherche
CPLEX 7.1	19,4	92845,9	46387,7
Cont. Coût Red.	58,7	149613	37079,6

(1) Différentes stratégies de résolution.
(2) Le temps moyen, en secondes, nécessaire pour résoudre une des 30 instances de taille $n = 100$ et $m = 5$
(3) Le nombre moyen d'itérations nécessaire pour résoudre une des 30 instances de taille $n = 100$ et $m = 5$
(4) Taille moyenne de l'arbre de séparation et évaluation pour résoudre une des 30 instances $n = 100$ et $m = 5$

Tab. 3.7 – Comparaison des différentes stratégies pour résoudre le PSDM de la "OR-Library" $n = 100$ et $m = 5$.

Stratégie(1)	Temps(sec)(2)	Nbre moyen(3) d'itérations Simplexe	Taille moyenne (4) de l'arbre de recherche
CPLEX 7.1	126,1	863315	251302
Cont. Coût Red.	1279,5	2856682	52097

(1) Différentes stratégies de résolution.
(2) Le temps moyen, en secondes, nécessaire pour résoudre une des 30 instances de taille $n = 100$ et $m = 10$
(3) Le nombre moyen d'itérations nécessaire pour résoudre une des 30 instances de taille $n = 100$ et $m = 10$
(4) Taille moyenne de l'arbre de séparation et évaluation pour résoudre une des 30 instances $n = 100$ et $m = 10$

TAB. 3.8 – Comparaison des différentes stratégies pour résoudre le PSDM de la "OR-Library" $n = 100$ et $m = 10$.

Deuxième partie

Le Problème d'Allocation de Fréquences

Chapitre 4

Etat de l'art des méthodes exactes et présentation d'un cas concret du problème d'allocation de fréquences

---

Dans ce chapitre, nous présentons un cas concret du problème d'allocation de fréquences (PAF) dans un réseau de télécommunications par voie hertzienne. Le réseau comporte un ensemble de trajets et il s'agit d'affecter une fréquence à chaque trajet, sachant que le nombre de fréquences disponibles est limité et que l'allocation de fréquences trop proches à un ensemble de trajets voisins peut provoquer des interférences. Nous présentons la formulation, généralement rencontrée dans la littérature et reposant sur des hypothèses simplificatrices, qui introduit des contraintes imposant un écart minimum aux fréquences allouées à seulement deux trajets voisins. Nous présentons ensuite la terminologie et les notations utilisées tout au long de cette partie. Enfin, nous décrivons les différents travaux liés à la résolution exacte du problème de minimisation du nombre de contraintes violées lorsqu'il n'y a pas de solution réalisable et du problème de minimisation de la largeur de la bande passante dans le cas contraire.

4.1 Introduction

Le sujet étudié dans cette partie du rapport a été proposé par le Centre ÉLectronique de l'ARmement (CELAR). Le CELAR, en tant que centre technique de la Délégation Générale pour l'Armement (DGA), est chargé, dans le cadre de programmes d'armement ou d'études amonts, de la réalisation et du suivi d'une partie des travaux visant à optimiser l'utilisation du spectre au sein des forces armées françaises. Dans ce contexte, le sujet proposé se rattache à celui de l'allocation des fréquences dans les réseaux de télécommunications par voie hertzienne. Les problèmes proposés par le CELAR ont fait partie du projet CALMA (Combinatorial Algorithms for Military Applications) [3] dans les années 1993-1995. Les problèmes présentés dans cette étude ont été enrichis pour prendre en compte la notion de contraintes cumulatives que nous présenterons dans le chapitre 5. Cette étude correspond à un projet où plusieurs équipes sont intervenues, comme décrit dans la section suivante.

Dans ce qui suit, la section 4.2 donne une description physique du problème, présente les objectifs du CELAR et précise notre contribution dans le projet. La section 4.3 présente les notations et la terminologie employées tout au long de la deuxième partie de ce mémoire. Enfin, un état de l'art est présenté dans la section 4.4.

4.2 Description physique du problème et objectifs

Un réseau de télécommunications par voie hertzienne est composé d'un ensemble de sites sur lesquels sont implantés les supports de transmission, à savoir, des antennes raccordées à des émetteurs/récepteurs. Une **liaison** hertzienne établit un lien entre deux sites géographiques. Elle est constituée par deux trajets. On appelle **trajet** un bond radioélectrique unidirectionnel, établi entre des antennes sur des sites géographiques distincts, dont la fréquence et la polarisation doivent être fixées, voir figure 4.1.

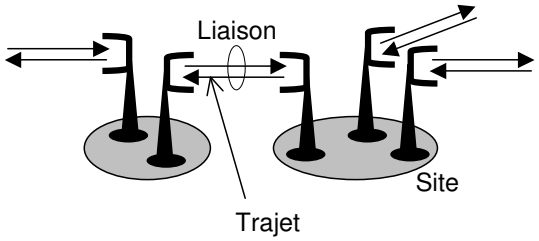

FIG. 4.1 – Sites, liaisons et trajets

De plus, à chaque trajet est associé un **système** qui représente le type de matériel utilisé avec ses caractéristiques d'émission et de réception. Si le nombre de trajets peut varier de quelques-uns à plusieurs milliers, le nombre de systèmes différents est souvent petit devant le nombre de trajets.

On définit une **ressource** comme un couple **(fréquence, polarisation)**, dont les composantes sont associées respectivement à la fréquence porteuse du signal transmis et à la polarisation de l'onde. Cette dernière consiste parfois en une variable bivalente (par exemple, rectiligne verticale ou rectiligne horizontale). Dans le cadre de la présente étude, on se limitera à une valeur de polarisation unique, ce qui revient à considérer uniquement la composante fréquentielle [1] de la ressource.

Pour chaque trajet, on définit ainsi un ensemble de fréquences (valeurs entières) qui sont susceptibles de lui être affectées : cet ensemble est appelé **domaine de fréquences du trajet**. Il est généralement déterminé par la réglementation, les contraintes liées aux matériels et/ou par des choix du gestionnaire de fréquences. Deux trajets peuvent avoir des domaines identiques.

Allouer des fréquences consiste donc à trouver une fréquence pour chaque trajet, dans le domaine de ressources fréquentielles associé à ce trajet et satisfaisant à la fois des contraintes impératives et des contraintes de Compatibilité Électro-Magnétique (**CEM**). En effet, les fréquences ne peuvent pas être allouées aux différents trajets sans précaution ; la localisation sur un même site de plusieurs émetteurs/récepteurs impose des contraintes quant aux ressources affectées pour garantir un fonctionnement optimal.

En plus des contraintes de domaine déjà présentées, plusieurs types de contraintes sont à prendre en compte :

1. contraintes impératives d'égalité ou de différence entre fréquences,
2. contraintes impératives duplex,
3. contraintes de pré-affectation,
4. contraintes de Compatibilité Électro-Magnétique, CEM.

On définit une **solution réalisable** comme l'allocation d'une ressource à chaque trajet satisfaisant l'ensemble des contraintes posées.

Malheureusement, de nombreux problèmes réels n'ont pas de solution réalisable. Des domaines de ressources trop restreints ou des exigences trop fortes peuvent souvent conduire à un tel diagnostic. L'allocateur doit donc pouvoir rechercher des solutions dégradées, en acceptant une perte de qualité satisfaisante. Pour cela, on définit deux types

[1] Le challenge ROADEF'2001 proposé par le CELAR [1] propose des instances de PAF avec polarisation

de contraintes :

- les contraintes **impératives**, de type 1 et 2 ci-dessus, ainsi que les contraintes de domaine, qui ne peuvent pas être relâchées ;
- les contraintes **souples**, de type 3 et 4 ci-dessus. Ces contraintes peuvent être violées avec pour conséquence une diminution de la qualité de transmision.

L'objectif du CELAR pour l'étude de ce cas concret d'allocation de fréquences concerne :

- une nouvelle modélisation des contraintes de Compatibilité Électro-Magnétique ;
- une étude de réalisabilité des instances réelles proposées ;
- la recherche de solutions optimales pour le problème suivant : une instance est supposée réalisable si les contraintes impératives peuvent être satisfaites. Lorsque le problème est réalisable vis à vis des contraintes impératives et des contraintes souples, il s'agit de minimiser la largeur de la bande passante occupée. Si au moins une des contraintes souple doit être violée, il s'agit de minimiser une somme pondérée du nombre de contraintes souples violées.

Dans le cadre cette étude, plusieurs équipes sont intervenues et ont travaillé sur différentes méthodes de résolution du problème. Des méthodes heuristiques de résolution à grand voisinage ont été éprouvées par Palpant *et al.* [98, 99]. Une méthode tabou basée sur un voisinage consistant a été testée par Vlasak et Vasquez [122]. Un algorithme de recuit simulé a été proposé par Sarzeaud [108].

Notre propre contribution porte sur la nouvelle modélisation des contraintes de Compatibilité Électro-Magnétique par des inégalités linéaires, sur l'étude de réalisabilité des problèmes et, enfin, sur le développement de méthodes exactes basées sur la programmation linéaire et la programmation par contraintes pour résoudre les deux problèmes d'optimisation.

4.3 Notation et Terminologie

T est l'ensemble des trajets du problème. f_i désigne la fréquence affectée au trajet $i \in T$. Les domaines fréquentiels associés au problème sont notés $F_1 \dots F_N$. En général, N est petit devant $|T|$. Le domaine fréquentiel d'un trajet $i \in T$ est noté $\varphi(i) \in \{1, \dots, N\}$.

Dans la suite, nous présentons la formulation mathématique des contraintes impératives (4.3.1) et de préaffectation (4.3.2), la formulation binaire des contraintes de compatibilité électromagnétique rencontrée dans la littérature (4.3.3) et les fonctions objectifs

considérées (4.3.4).

4.3.1 Contraintes impératives

Contraintes duplex

Les contraintes dites **duplex** se traduisent par un écart $\epsilon_{ij} \geq 0$ entre les fréquences des deux trajets d'une même liaison. Il existe deux types de contraintes duplex : les contraintes duplex d'écart imposé,

$$|f_i - f_j| = \epsilon_{ij}, \tag{4.1}$$

et les contraintes duplex d'écart interdit :

$$|f_i - f_j| \neq \epsilon_{ij} \tag{4.2}$$

Contraintes d'égalité et de différence

Ces contraintes sont impératives et se traduisent simplement par une égalité ou une différence de fréquences entre trajets :

$$f_i = f_j \tag{4.3}$$

$$f_i \neq f_j \tag{4.4}$$

Ces contraintes sont également impératives.

4.3.2 Contraintes de pré-affectation

Ces contraintes traduisent la nécessité d'allouer une fréquence prédéterminée $p_i \in F_{\varphi(i)}$ à un trajet donné. Il s'agit généralement d'une contrainte souple.

$$f_i = p_i \tag{4.5}$$

4.3.3 Contraintes CEM

L'allocation des ressources doit se baser sur l'environnement électromagnétique de la zone et prendre en compte toutes les perturbations susceptibles de dégrader les performances des liaisons. Cet environnement électromagnétique est composé de systèmes pouvant être de nature différente, plusieurs équipements pouvant être amenés à communiquer à proximité les uns des autres.

La qualité de communication est basée sur des calculs d'ingénierie de liaison et de compatibilité électromagnétique, prenant en compte les émissions générées par des émetteurs voisins susceptibles de perturber un récepteur donné. Un critère souvent considéré est le C/I qui exprime, en fonction des caractéristiques techniques des équipements, un seuil limite à ne pas dépasser entre la puissance utile C par le perturbé et la puissance perturbatrice I reçue émanant de tous les émetteurs avoisinants.

Les approches les plus répandues considèrent que ce "droit de perturber", défini par le critère C/I, est réparti uniformément sur les différents perturbateurs. On passe ainsi d'une situation avec N perturbateurs sur un récepteur à une situation plus simple consistant N situations élémentaires avec un perturbateur et un récepteur (voir figure 4.2). Cette simplification permet de définir les contraintes CEM habituelles de la façon suivante :

$$|f_i - f_j| \geq \delta_{ij}, \tag{4.6}$$

où $\delta_{ij} \geq 1$ représente l'écart minimal imposé entre f_i et f_j.

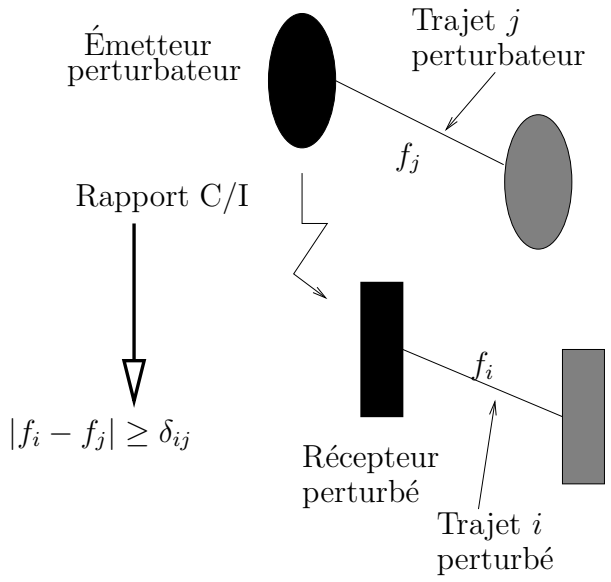

FIG. 4.2 – Une situation élémentaire avec un perturbateur et un récepteur.

Ces contraintes sont de type souple. Notons que les contraintes de différence (4.4) correspondent au cas particulier $\delta_{ij} = 1$.

Nous présenterons dans le chapitre 5 une modélisation de contraintes CEM plus proche de la réalité. Dans la suite, nous nous consacrons à l'étude de la littérature concernant le problème présenté ci-dessus, dont une grande majorité reprend cette hypothèse simplificatrice.

4.3.4 Fonctions objectifs

Au cas où toutes les contraintes souples peuvent être satisfaites, il s'agit de minimiser la largeur de la bande passante, ce que l'on exprime comme suit :

$$\min(\max_{i \in T} f_i - \min_{i \in T} f_j) \tag{4.7}$$

Dans le cas contraire, il s'agit de minimiser une somme pondérée du nombre de contraintes souples violées, ce qui s'exprime par :

$$\min(\sum q_{ij} \#(|f_i - f_j| < \delta_{ij}) + \sum m_i \#(|f_i - p_i| > 0)) \tag{4.8}$$

où q_{ij} est le coût de violation de la contrainte CEM portant sur les trajets i et j, m_i est le coût de violation d'une contrainte de préaffectation et $\#(c)$ vaut 1 si la condition c est vérifiée et 0 sinon.

4.4 L'état de l'art

Nous présentons le concept de graphe de contraintes et la complexité du problème dans la section 4.4.1. Dans la section 4.4.2, nous introduisons la formulation de PLNE généralement rencontrée dans la littérature, dont nous dérivons la notion de graphe des conflits et qui nous permet de présenter l'état de l'art en suivant une classification en fonction des deux types de fonctions objectifs présentées dans la section 4.3.4. Notre étude concernant uniquement les méthodes exactes, nous ne présentons pas dans cet état de l'art les heuristiques proposées pour résoudre ces problèmes. Nous renvoyons le lecteur intéressé à l'article d'Aardal *et al.* [2].

4.4.1 Graphe de contraintes et complexité

Le PAF peut être représenté par un graphe de contraintes $G = (T, E)$, aussi appelé graphe d'interférences. Dans ce graphe, les nœuds correspondent aux trajets du réseau. Une arête est définie entre deux nœuds si une contrainte existe entre les deux trajets. La figure 4.3 présente une partie du graphe de contraintes pour l'instance FAPPG01_0016 du CELAR. En supposant que le problème ne comporte que des contraintes CEM (4.6), que les écarts δ_{ij} sont unitaires, et que les domaines des fréquences associées aux trajets sont tous égaux à l'ensemble des entiers de 1 à k, la question de savoir s'il existe une solution réalisable au PAF, revient à savoir s'il existe une k-coloration du graphe (T, E), ce qui est un problème NP-complet pour $k \geq 3$.

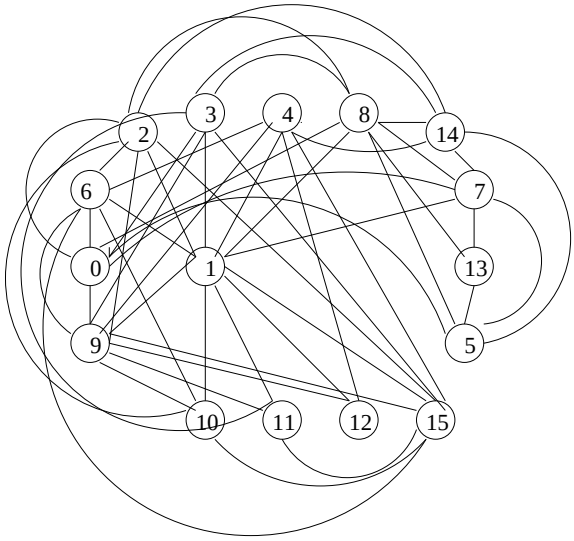

FIG. 4.3 – Un graphe de contraintes partiel de l'instance FAPPG01_0016 du CELAR

4.4.2 Formulation de PLNE, graphe des conflits et classification

Une formulation de PLNE possible pour le PAF avec l'ensemble des contraintes (4.3), ..., (4.6) utilise des variables binaires représentant le choix d'une fréquence pour un certain trajet. Pour chaque trajet $i \in T$ et pour chaque fréquence $f \in F_{\varphi(i)}$, on définit la variable x_{if} comme suit.

$$x_{if} = \begin{cases} 1 & \text{si la fréquence } f \in F_{\varphi(i)} \text{ est allouée au trajet } i \in T, \\ 0 & \text{sinon.} \end{cases} \tag{4.9}$$

À partir de ces variables, nous pouvons définir le graphe des conflits, utilisé dans

certaines approches, qui comporte autant de nœuds que de variables x_{if}, c'est-à-dire $\sum_{i \in T} |F_{\varphi(i)}|$ nœuds. Une arête entre deux nœuds distincts x_{iv} et x_{jg} est définie si $i = j$ ou si l'affectation de v à f_i et de g à f_j viole une des contraintes (4.3), ..., (4.6). Ce graphe représente ainsi l'ensemble des contraintes portant sur les variables x_{if}.

La condition qu'une fréquence unique doit être affectée à un trajet est représentée par les contraintes suivantes :

$$\sum_{f \in F_{\varphi(i)}} x_{if} = 1 \quad \forall i \in T \tag{4.10}$$

Les contraintes impératives du type (4.3) ne sont pas traduites en contraintes linéaires car un prétraitement trivial permet de supprimer une des deux variables intervenant dans l'égalité. Par contre, les contraintes du type (4.1), (4.2) et (4.4) sont représentées comme suit :

$$x_{if} \leq x_{j(f+\epsilon)} + x_{j(f-\epsilon)} \qquad \forall (i,j) \in CI_1, \forall f \in F_{\varphi(i)} \tag{4.11}$$

$$x_{if} + x_{j(f+\epsilon)} + x_{j(f-\epsilon)} \leq 1 \qquad \forall (i,j) \in CI_2, \forall f \in F_{\varphi(i)} \tag{4.12}$$

$$x_{if} + x_{jf} \leq 1 \quad \forall (i,j) \in CI_4, \forall f \in F_{\varphi(i)} \cap F_{\varphi(j)} \tag{4.13}$$

où CI_1, CI_2, CI_4 désignent les ensembles de paires de trajets concernées par les contraintes (4.1), (4.2) et (4.4), respectivement.

Par souci de clarté, nous allons distinguer dans la suite du chapitre deux types de problèmes dont dépendent les formulations linéaires de la fonction objectif et des contraintes souples de pré-affectation (4.5) et CEM (4.6) : les problèmes non réalisables **PAFNR** et les problèmes réalisables **PAFR**. Les problèmes non réalisables sont les problèmes pour lesquels il n'existe pas d'allocation de fréquences sans violer au moins une des contraintes du type (4.5) ou (4.6). Quant aux problèmes réalisables, ce sont les problèmes pour lesquels toutes les contraintes de type (4.5) et (4.6) peuvent être satisfaites. Nous présenterons la suite de la formulation de PLNE et l'état de l'art pour le PAFNR dans la section 4.4.3 et pour le PAFR dans la section 4.4.4.

4.4.3 Problème d'allocation de fréquences non réalisable

Tel qu'il est défini ci-dessus, un problème non réalisable[2] se caractérise par la présence d'interférences dans le réseau. Par conséquent, il est nécessaire de faire des compromis sur les exigences que doivent satisfaire les solutions. Les objectifs souvent traités dans la littérature sont :

- minimiser la probabilité moyenne de blocage des appels dans le réseau ;
- minimiser la demande totale non satisfaite ;
- minimiser le nombre de violations des contraintes de compatibilité électromagnetique CEM et de pré-affectation ;
- minimiser la somme pondérée des violations de contraintes CEM et de pré-affectation de façon à respecter en premier les contraintes les plus critiques.

Dans ce qui suit, nous nous consacrons principalement à l'étude de méthodes concernant les deux derniers objectifs, en relation avec l'étude proposée par le CELAR. Pour un état de l'art plus complet, nous recommandons la lecture de l'article d'Aardal *et al.* [2].

Formulation de PLNE du PAFNR

Dans la littérature, une façon de modéliser un PAFNR en PLNE est d'introduire dans la fonction objectif une pénalité $q_{ijfg} \geq 0$, qui représente le coût d'affectation simultanée de la fréquence f au trajet i et de la fréquence g au trajet j pour les contraintes souples de type CEM, et une pénalité $M_{if} \geq 0$ pour les contraintes de type pré-affectation, avec $m_{ip_i} = 0$. Une pénalité non nulle correspond à une arête dans le graphe d'interférences. La fonction objectif (4.8) à minimiser est de la forme :

$$\sum_{(i,j)\in CE} \sum_{f\in F_{\varphi(i)}, g\in F_{\varphi(j)}} q_{ijfg} x_{if} x_{jg} + \sum_{i\in CP} \sum_{f\in F_{\varphi(i)}} m_{if} x_{if}$$

où CE désigne l'ensemble des paires de trajets soumis aux contraintes CEM (4.6) et CP désigne l'ensemble des trajets soumis aux contraintes de pré-affectation (4.5). Nous notons que la fonction objectif est quadratique. Pour la linéariser, il est d'usage de remplacer chaque terme $x_{if}x_{jg}$ par la variable suivante :

[2] Ce problème est connu dans la littérature sous le nom de *fixed spectrum problems*

$$z_{ijfg} = \begin{cases} 1 & \text{si } x_{if} = x_{jg} = 1 \\ 0 & \text{sinon} \end{cases}$$

Alors, le PAF non réalisable s'écrit comme :

$$\begin{aligned} \min \quad & \sum_{(i,j)\in CE} \sum_{f\in F_{\varphi(i)}, g\in F_{\varphi(j)}} q_{ijfg} z_{ijfg} + \sum_{i\in CP} \sum_{f\in F_{\varphi(i)}} m_{if} x_{if} \\ \text{s-à} \quad & \\ & (4.9), \quad (4.10), \quad (4.13), \quad (4.11), \quad (4.12) \\ & x_{if} + x_{jg} \leq 1 + z_{ijfg} \quad \forall (i,j) \in CE, \quad f \in F_{\varphi(i)}, \quad g \in F\varphi(j) \end{aligned} \tag{4.14}$$

Notons qu'une seule des variables z_{ijfg} peut être égale à 1 pour toute solution entière réalisable. De même, une seule variable x_{if} peut être égale à 1 d'après la contrainte (4.10). Le problème revient ainsi à une minimisation du nombre pondéré de contraintes CEM et de pré-affectation non satisfaites, où le poids peut dépendre de l'ampleur de la violation.

Types d'instances Les travaux de la littérature les plus proches de la problématique posée par le CELAR sont ceux liés au projet CALMA [3] qui concerne la résolution de PAF comportant l'ensemble des contraintes présentées ci-dessus et, pour un sous-ensemble d'instances irréalisables, la minimisation de la somme pondérée du nombre de contraintes CEM et de pré-affectation violées. Nous présentons donc principalement les résultats liés au projet CALMA. Nous présentons également quelques approches remarquables pour des problèmes voisins, comme les instances Bell [5] concernant les réseaux de téléphonie mobile et la minimisation de la probabilité de blocage.

Bornes inférieures et méthodes exactes Pour le problème de minimisation des interférences, il existe très peu de méthodes exactes publiées. La plupart concerne uniquement la proposition de bornes inférieures, hormis les deux premières approches citées ci-après proposant également des méthodes exactes de recherche arborescente.

Verfaillie *et al* [121] proposent une méthode exacte basée sur la "méthode des poupées russes". Etant donnée une solution partielle portant sur n variables, l'algorithme des poupées russes consiste à résoudre au plus n sous-problèmes, chaque sous problème pour $i = 1, .., n$ consistant à chercher une nouvelle solution (par une recherche en profondeur classique) en libérant i variables selon un ordre donné. Cette méthode a permis de résoudre

optimalement l'instance CALMA CELAR 06.

Koster *et al* [72] résolvent le problème par une procédure de séparation et d'évaluation basée sur la relaxation de programmation linéaire du PLNE et proposent des inégalités valides basées sur des structures de type clique et cycle dans le graphe des contraintes. Ils arrivent à résoudre 5 problèmes avec 100 trajets, 350 contraintes et de 2 à 6 fréquences dans chaque domaine des trajets.

Tiourine et al [115] proposent une borne inférieure et une technique de prétraitement. La borne inférieure est obtenue en résolvant exactement un problème en variables 0-1 sans contraintes avec une fonction objectif non linéaire qui ignore une partie des interférences possibles : celles entre deux trajets violant la contrainte de pré-affectation. Le prétraitement est obtenu en partitionnant le domaine d'un trajet et en testant la borne inférieure sur chaque sous-domaine. Si elle est supérieure à la meilleure borne supérieure connue, alors la partie du domaine du trajet est supprimée. En utilisant ce prétraitement pour deux instances du projet CALMA du CELAR, la taille des domaines a pu être réduite de 20%. Cette approche fonctionne bien si les deux conditions suivantes sont vérifiées. D'une part, un nombre significatif de trajets doivent avoir des fréquences pré-affectées et au moins quelqu'unes de ces fréquences ne peuvent pas être changées. D'autre part, les coûts d'interférences CEM doivent être plus petits que les coût de changement d'une fréquence préaffectée.

Koster et al[71] ont proposé un algorithme de programmation dynamique basé sur la décomposition en arbre du graphe de contraintes. Le problème est décomposé en sous-problèmes plus petits et ces derniers sont ensuite étendus graduellement au problème global. La décomposition est basée sur l'observation que, pour le graphe de contraintes, le nombre de fréquences possibles pour un trajet dépend seulement des fréquences affectées aux trajets interférant avec lui. En général, si S est un ensemble de séparation des sommets du graphe d'interférence G et si une affectation optimale pour S est connue, alors les solutions optimales pour toutes les composantes connexes séparées par S dépendent seulement de S et peuvent être calculées indépendamment les unes des autres. Le problème doit être résolu pour chaque affectation réalisable pour S, car aucune affectation optimale pour S n'est connue à l'avance. L'algorithme commence en calculant une décomposition en arbre du graphe G, ce qui donne une couverture de G par un ensemble de sous-graphes qui correspondent aux nœuds de l'arbre. Les sommets de ce sous-graphe forment un ensemble de séparation des sommets de G. À partir de cette décomposition, les sous-problèmes initiaux et l'ordre dans lequel ils sont étendus sont déterminés. Les sous-problèmes initiaux sont donnés par les sous-graphes qui correspondent aux nœuds feuilles dans l'arbre de décomposition. Par la suite, les sous-problèmes sont étendus en

fusionnant des nœuds de l'arbre. Toutes les solutions réalisables sont évaluées pour chaque sous-problème, un pré-traitement étant appliqué pour réduire sa taille, et des bornes étant employées pour limiter l'espace des solutions. Une borne supérieure et une borne inférieure du PAFNR sont calculées à partir des sous-problèmes. Cet algorithme a été testé sur les instances du projet CALMA et parvient à en résoudre 3 optimalement et à obtenir de très bonnes bornes inférieures sur les autres. Cependant, il faut noter que l'algorithme s'avère efficace uniquement si des techniques de réduction et de dominance sont utilisées.

La méthode de Montemanni *et al* [84] propose la minimisation de la somme pondérée associée aux contraintes CEM violées. Ils utilisent la relaxation de programmation linéaire et la renforcent en ajoutant deux types d'inégalités : une inégalité sur la borne inférieure de la fonction objectif et une inégalité sur la borne inférieure du nombre de contraintes violées. Ces bornes sont toutes deux obtenues à partir d'un sous-problème dérivé du modèle original. Ces sous-problèmes sont de type "clique" : une clique de niveau k, c'est-à-dire une clique qui est construite à partir de contraintes CEM qui imposent au moins une distance k entre les fréquences, et une autre clique qui n'est pas maximale, c'est-à-dire qui est contenue dans une autre clique plus grande. Les résultats montrent que la méthode obtient des résultats encourageants mais non uniformes sur toutes les instances testées. La taille de ces instances est relativement petite. Les mêmes auteurs, Montemanni *et al.* [83] proposent une amélioration de ces bornes au prix de temps de calculs prohibitifs. Certaines méthodes de calcul de bornes inférieures pour le FAPR (voir section 4.4.4) utilisent également des cliques de G.

Dans le cadre des instances Bell pour la téléphonie mobile, Adjakplé[5] et plus tard Jaumard et al[65] proposent une borne inférieure pour le problème de minimisation de la demande non satisfaite. L'originalité est l'introduction d'une formulation de programmation linéaire en variables 0-1 où chaque variable correspond à l'affectation potentielle d'un sous-ensemble de cellules à un bloc de fréquences, sur la base d'une formulation de problèmes de couverture. Une technique de génération de colonnes est proposée pour la résolution de la relaxation continue. Le sous-problème correspond à l'optimisation d'une fonction quadratique en variable 0-1 sans contraintes. Dans [5], les résultats obtenus montrent que certaines instances ne sont pas réalisables mais aucune méthode exacte n'est proposée. Dans [65], de petites instances sont résolues exactement.

4.4.4 Problème d'allocations de fréquences réalisable

Pour les PAFR, les deux principaux objectifs considérés dans la littérature sont :

- minimiser la largeur de la bande utilisée [3] ;
- minimiser le nombre de fréquences utilisées [4].

L'objectif de minimiser le nombre de fréquences utilisées vient des années 70 où les fréquences étaient vendues à l'unité et coûtaient très cher. Nous nous consacrons à l'étude de la littérature concernant le premier objectif, correspondant au projet proposé par le CELAR.

Formulation PLNE du PAFR

Le problème d'allocation de fréquences qui minimise la largeur de la bande utilisée suppose que les usagers paient pour l'ensemble complet de fréquences entre la plus grande et la plus petite d'entre elles. Ainsi, la différence entre la fréquence maximale et minimale (le "span") détermine le coût et doit être donc minimisée. Pour modéliser la fonction objectif (4.7), nous introduisons deux variables entières notées f_{max} et f_{min}, où f_{max} représente la fréquence maximale utilisée et f_{min} la fréquence minimale utilisée. Le PAFR s'écrit alors :

$$\min \quad f_{\max} - f_{\min} \tag{4.15}$$

$$\text{s-à} \quad (4.9), (4.10), (4.13), (4.11), (4.12)$$

$$f_{\max} \geq \sum_{v \in F_\varphi(i)} v x_{iv} \qquad \forall i \in T \tag{4.16}$$

$$f_{\min} \leq \sum_{v \in F_\varphi(i)} v x_{iv} \qquad \forall i \in T \tag{4.17}$$

$$x_{if} + x_{jg} \leq 1 \qquad \forall (i,j) \in CEM, f \in F_i, g \in Fj \tag{4.18}$$

$$f_{max} \geq 0, f_{min} \geq 0 \quad \text{et entiers} \tag{4.19}$$

Des formulations alternatives ont été proposées, par exemple par Giortzis et Turner [43] et Baybars [14]. Giortzis et Turner proposent une formulation basée sur des variables binaires plutôt que f_{max} et f_{min}. Les nouvelles variables considérées sont :

$$bs_f = \begin{cases} 1 & \text{si la fréquence } f \in \bigcup_{r=1\ldots N} F_r \text{ est la plus grande utilisée} \\ 0 & \text{autrement} \end{cases}$$

[3] Dans la littérature ce problème est connu sous le nom de *minimum span problem*
[4] Dans la littérature ce problème est connu sous le nom de *minimum order problem*

$$bi_f = \begin{cases} 1 & \text{si la fréquence } f \in \bigcup_{r=1\ldots N} F_r \text{ est la plus petite utilisée} \\ 0 & \text{autrement} \end{cases}$$

Avec ces variables, la fonction objectif (4.7) s'écrit :

$$\min \sum_{f \in \bigcup\limits_{r=1\ldots N} F_r} f \times bs_f - \sum_{f \in \bigcup\limits_{r=1\ldots N} F_r} f \times bi_f \tag{4.20}$$

et les contraintes suivantes doivent être introduites :

$$\sum_{f \in \bigcup\limits_{r=1\ldots N} F} bs_f = 1 \tag{4.21}$$

$$\sum_{f \in \bigcup\limits_{r=1\ldots N} F} bi_f = 1 \tag{4.22}$$

$$x_{if} + bs_g \leq 1 \quad \forall i \in T, f \in F_i, g \in \{ \bigcup_{r=1\ldots N} F_r | f > g\} \tag{4.23}$$

$$x_{if} + bi_g \leq 1 \quad \forall i \in T, f \in F_i, g \in \{ \bigcup_{r=1\ldots N} F_r | f < g\} \tag{4.24}$$

Les contraintes (4.21) et (4.22) assurent qu'il existe une unique fréquence maximale et une unique fréquence minimale. Les contraintes (4.23) et (4.24) interdisent d'affecter des fréquences plus grandes au maximum et d'affecter des fréquences plus petites au minimum, respectivement.

Les méthodes exactes On trouve dans la littérature pour le PAFR la description d'inégalités valides et de bornes inférieures que nous présentons ci-après. Toutefois aucun résultat expérimental ne valide leur utilisation à l'intérieur de méthodes exactes.

Les variables introduites dans le modèle de Giortzis et Turner [43] (voir le paragraphe précédent) produisent des cliques spécifiques qui, d'après Aardal *et al.* [2], n'ont jamais été testées expérimentalement.

$$\sum_{f \in F_i | f \leq g} x_{if} + \sum_{f \in F_i | f > g} bi_f \leq 1 \quad \forall i \in T, g \in F_i,$$
$$\sum_{f \in F_i | f \geq g} x_{if} + \sum_{f \in F_i | f < g} bi_f \leq 1 \quad \forall i \in T, g \in F_i$$

Une borne inférieure très intéressante pour ce problème est celle obtenue par Raychaudhuri [106]. Elle repose sur la constatation que les chemins hamiltoniens dans un graphe complet ont un lien avec le problème de minimisation de la largeur de la bande passante. On considère le graphe des contraintes G réduit aux contraintes CEM (4.6), en valuant les arêtes correspondantes par δ_{ij}, et aux contraintes d'écart imposé (4.1), en valuant les arêtes correspondantes par ϵ_{ij}. Toute solution réalisable du PAF peut être utilisée pour construire un chemin hamiltonien dans G de coût supérieur ou égal à celui du chemin hamiltonien de poids minimum. Dans [76], le théorème suivant est démontré :

Théorème 4.1 *Si G est un graphe complet, et $H(G)$ est la valeur d'un chemin hamiltonien de poids minimum dans G, alors $span(G) \geq H(G)$.*

Trouver $H(G)$ est aussi connu sur le nom de "Problème de Voyageur de Commerce ouvert" (PVC). C'est ainsi que cette borne est connue sur le nom de "borne PVC". Bien que la borne PVC puisse être utilisée pour beaucoup de problèmes d'allocation de fréquences, il existe deux inconvénients :

1. trouver un chemin hamiltonien de coût minimal est NP-Difficile ;
2. la borne PVC ne tient pas compte des contraintes entre deux nœuds non consécutifs.

Le PVC peut être modélisé comme un programme linéaire en nombres entiers comme suit, où T et E sont respectivement les arêtes et les nœuds du graphe G :

$$\begin{aligned}
\min \quad & \textstyle\sum_{(i,j) \in E} c_{ij} x_{ij} && (4.25)\\
s.. \quad & \textstyle\sum_{j | (i,j) \in E} x_{ij} = 2 \quad \forall i \in T && (4.26)\\
& \textstyle\sum_{i \in S, j \in T} x_{ij} \geq 2 \quad \forall S \subset T && (4.27)\\
& x_{ij} \in \{0,1\} \qquad (i,j) \in E && (4.28)
\end{aligned}$$

Les relaxations de ce modèle pour calculer une borne sont, soit la relaxation continue soit la relaxation les contraintes de sous-tours (4.27) obtenant ainsi un problème de cou-

plage parfait. Ces relaxations permettent d'obtenir des bornes plus rapidement mais au détriment de la qualité. Nous montrons dans le chapitre 5, que la borne du problème du couplage parfait est une très bonne approximation de la borne PVC pour les problèmes d'allocation de fréquences. En fait, une des caractéristiques du PAF est que, d'une part, les poids des arêtes sont asymétriques et, d'autre part, l'inégalité triangulaire n'est pas respectée. Ceci aide le problème du couplage à résoudre le problème de voyageur de commerce et, par conséquent, à donner une bonne borne inférieure pour le PAF. Ce problème peut être résolu par un algorithme polynomial. Allen *et al.* [6] introduisent de contraintes additionnelles pour améliorer la borne PVC en considérant les contraintes de séparation de fréquences entre les nœuds non consécutifs dans le chemin hamiltonien. Ces contraintes sont basées sur le concept de variables d'excès pour chaque arête du chemin. Allen *et al.* [6] indiquent que la borne inférieure trouvée en utilisant ces contraintes additionnelles améliore la borne PVC de 0.2% à 3%.

Il faut noter que pour la plupart des instances du PAF, le graphe de contraintes contient une grande quantité d'arêtes de poids nul. Les bornes obtenues sont souvent alors nulles. Cependant, de meilleures bornes peuvent être obtenues en considérant un sous-graphe du graphe de contraintes. En effet, toute borne inférieure de la largeur de la bande passante d'un sous-problème est aussi une borne inférieure de la largeur de la bande passante du problème original. Les cliques de G semblent constituer un bon choix de sous-graphe pour la détermination de la borne inférieure. Les cliques utilisées sont appelées "k-cliques". Une "k-clique" dans un graphe de contraintes G est un sous-graphe complet dans lequel chaque arête a un poids supérieur à k. Ce type de cliques est également utilisé pour le PAFNR (voir section 4.4.3). Si les techniques de bornes inférieures sont appliquées aux k-cliques, alors la borne résultante est généralement meilleure que celle résultant du graphe de contraintes complet.

Enfin, des bornes simples peuvent être obtenues en utilisant les cliques. Gamst [40] a généralisé la borne dite "standard" d'une clique comme suit : si une clique de taille t dans le graphe d'interférence contient des arêtes avec une distance minimale égale à d, alors l'écart de fréquences doit être au moins de $(t-1)d+1$. D'autres bornes plus sophistiquées qui tiennent compte de la structure du problème sont apparues dans la littérature. Nous adressons le lecteur à [85]. D'autre part, Janssen et Wentzall [64] démontrent que beaucoup de ces bornes peuvent être dérivées à partir d'une approche appellée "tile covers". Plus de détails sur les bornes inférieures peuvent être trouvés dans [76].

Programmation par contraintes Walser [124] a appliqué la programmation par contraintes pour le PAFR. Il propose de réduire, de manière heuristique, le graphe de

contraintes en construisant une couverture adéquate des trajets avec des stables. Tous les trajets (nœuds) dans stable sont identifiés à un nœud unique, ce qui revient à affecter la même fréquence à tous les trajets originaux. Pour obtenir une solution, la technique de "Limited Discrepancy search" est utilisée.

4.5 Conclusion

L'analyse de l'état de l'art montre que les méthodes exactes pour résoudre les problèmes d'allocation de fréquences sont rares, voire inexistantes pour le problème de minimisation de la largeur de la bande passante. La plupart des recherches dans le domaine concerne des heuristiques ou des méthodes de calcul de bornes inférieures. Les bornes inférieures rencontrées, aussi bien pour le PAFNR que pour le PAFR, utilisent de manière intensive les cliques (ou d'autres sous-structures) du graphe des contraintes qui constituent des sous-problèmes essentiels et semblent donner des relaxations meilleures que la relaxation de programmation linéaire de la formulation classique. Par ailleurs, la taille des problèmes traités par ces méthodes reste très modeste. La programmation par contraintes est encore très peu utilisée. Dans le chapitre 5, nous proposons d'ouvrir une piste de recherche dans cette voie en comparant une approche de PLNE utilisant la relaxation de programmation linéaire, avec une approche hybride de programmation par contraintes utilisant une relaxation basée sur des cliques du graphe de contraintes et résolue par programmation linéaire.

D'autre part nous constatons que la modélisation des contraintes CEM sous la forme binaire actuellement rencontrée dans la majorité des travaux n'a pas de justification pratique dans le contexte de l'étude proposée par le CELAR. Le chapitre suivant étudie la modélisation de ces interférences sous la forme de contraintes cumulatives.

Chapitre 5

Nouvelle formulation et méthodes exactes pour le problème d'allocations de fréquences

Dans ce chapitre nous présentons une nouvelle modélisation pour le problème d'allocations de fréquences basée sur une approche plus réaliste des perturbations de type CEM introduisant des contraintes cumulatives. L'hypothèse simplificatrice de répartition uniforme de perturbation est relâchée. Une contrainte cumulative intègre la fréquence associée au trajet et l'ensemble des fréquences associés aux trajets perturbateurs. En suivant le schéma de résolution propre à l'étude proposée par le CELAR, nous proposons une méthode de séparation, d'évaluation et de génération de plans coupants pour la résolution d'un PLNE qui utilise une formulation linéaire des contraintes cumulatives. Ce schéma est ensuite appliqué au sein d'une méthode hybride de programmation par contraintes associée à une relaxation résolue par programmation linéaire.

5.1 Une représentation réaliste des contraintes de compatibilité électromagnétique

Dans le chapitre précédent (voir section 4.3.3), nous avons présenté une formulation binaire des contraintes CEM basée sur le concept de droit de perturber. Il est généralement considéré que ce droit de perturber est réparti uniformément sur les différents perturbateurs. Cette répartition uniforme n'a pas de justification réelle sinon sa simplicité de prise en compte. Le modèle présenté ici lève cette hypothèse d'uniformité. Les contraintes d'écart introduites à la section 4.3.3 disparaissent donc au profit de contraintes moins fortes mais plus complexes qui prennent en compte simultanément les fréquences des trajets perturbateurs et la fréquence du trajet perturbé. La figure 5.1 illustre ces perturbations multiples.

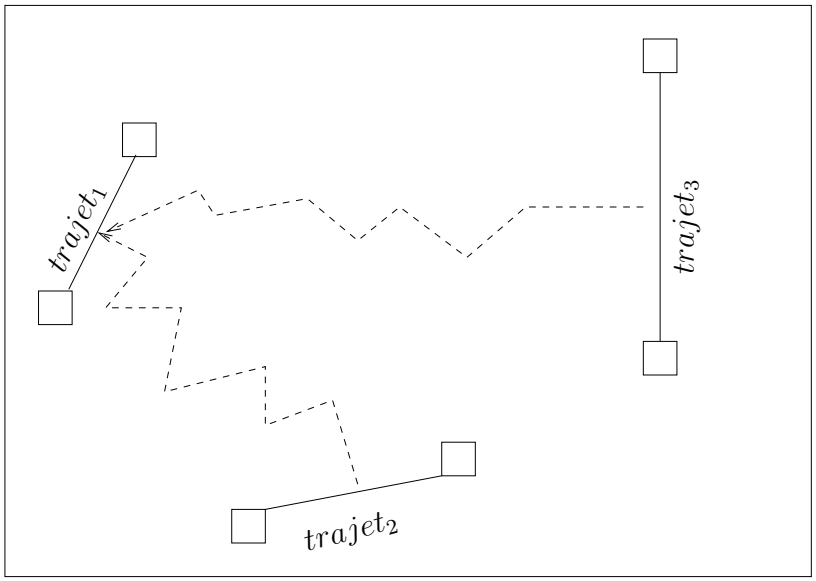

FIG. 5.1 – Perturbations multiples susceptibles de dégrader les performances des liaisons

5.1.1 Principe

Dans ce qui suit, nous présentons une nouvelle modélisation des contraintes CEM. Soit $\sigma(i)$ le système technologique associé au trajet i. On définit une fonction $T_{\sigma(i)\sigma(j)}$ de I N dans $\mathbb{R}$ telle que $T_{\sigma(i)\sigma(j)}(\delta)$ représente la perturbation du trajet j associé au système $\sigma(j)$ sur le trajet i associé au système $\sigma(i)$ quand l'écart entre leurs fréquences est égal à δ. Soit Λ_i un seuil acceptable pour le récepteur du trajet i calculé à partir du critère C/I. Soit P_i l'ensemble des trajets perturbateurs du trajet i. Si on considère un trajet perturbateur j de i, on pondère son influence avec un facteur multiplicatif λ_{ij}, qui prend en compte, entre autres, la distance et les orientations respectives des trajets considérés.

Les contraintes CEM deviennent alors :

$$\sum_{j \in P_i} \lambda_{ij} T_{\sigma(i)\sigma(j)}(|f_i - f_j|) \leq \Lambda_i \tag{5.1}$$

La fonction $T_{\sigma(i)\sigma(j)}(\delta)$ est positive, décroissante et tend vers 0 lorsque δ tend vers l'infini. La figure 5.2 montre la forme de la fonction $T_{\sigma(i)\sigma(j)}(\delta)$.

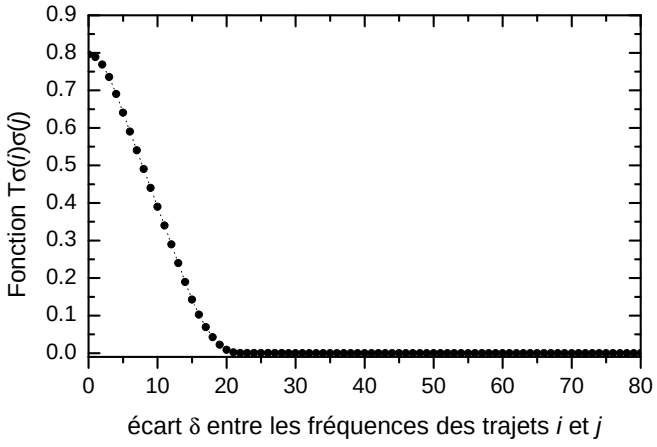

FIG. 5.2 – Une fonction $T_{\sigma(i)\sigma(j)}(\delta)$

C'est ce nouveau type de modélisation plus réaliste, abordé récemment dans la littérature par quelques études Dunkin et al.[27], Mannino et Sassano [79] que nous nous proposons de tester. Plus précisément, deux cas distincts seront considérés :

- Le premier quand les émetteurs perturbateurs sont colocalisés (sur le même site) avec le récepteur perturbé. On garde alors l'expression des contraintes sous la forme binaire (4.6) car ces perturbations, dites de type "champ proche", sont prises en compte sous forme de contraintes forfaitaires.
- Le second quand les perturbateurs sont éloignés du perturbé, on parle alors de perturbations de type "champ lointain". La nouvelle formulation (5.1) est considérée.

5.1.2 Formulation mathématique des problèmes étudiés

La spécificité du problème est de considérer la forme cumulative pour les contraintes CEM de type champ lointain. Ainsi, les contraintes prises en compte sont donc les suivantes :

- Contraintes impératives d'égalité ou d'inégalité entre fréquences,
- Contraintes impératives duplex,
- Contraintes CEM qui se répartissent en deux catégories :

– Contraintes en champ lointain sous la forme cumulative.
– Contraintes co-sites sous la forme binaire.

Nous rappelons les principales notations et contraintes présentées au chapitre 4. T est l'ensemble des trajets du problème. $F_1, F_2, \ldots, F_N$ représentent les domaines fréquentiels associés au problème. Pour chaque $i \in T$, on doit associer $f_i \in F_{\varphi(i)}$ où $\varphi(i)$ indique le domaine fréquentiel associé au trajet i. Les contraintes sont du type :

$$\begin{cases} |f_i - f_j| \geq \delta_{ij} & \forall (i,j) \in CE \\ \sum\limits_{j \in P_i} \lambda_{ij} T_{\sigma(i)\sigma(j)} (|f_i - f_j|) \leq \Lambda_i & \forall i \in CC \\ |f_i - f_j| = \epsilon_{ij} & \forall (i,j) \in CI_1 \\ |f_i - f_j| \neq \epsilon_{ij} & \forall (i,j) \in CI_2 \\ f_i = f_j & \forall (i,j) \in CI_3 \\ f_i \neq f_j & \forall (i,j) \in CI_4 \end{cases}$$

où CC désigne les ensembles de paires de trajets concernées par les contraintes (5.1).

Soit $V = \{(i,j) : |f_i - f_j| < \delta_{ij}\}$, l'ensemble de contraintes d'écart violées (4.6) et soit $W = \{ \sum\limits_{j \in P_i} \lambda_{ij} T_{\sigma(i)\sigma(j)} (|f_i - f_j|) > \Lambda_i\}$, l'ensemble de contraintes de type champ lointain (5.1) violées.

Nous rappelons que les fonctions de coût à optimiser sont les suivantes :
– Si le problème est réalisable, on cherchera à minimiser l'écart entre la fréquence maximale et la fréquence minimale des trajets alloués.
– Si le problème n'est pas réalisable, on cherchera à minimiser le nombre de contraintes non satisfaites avec des pondérations α et β respectivement pour les contraintes d'écart et champ lointain non satisfaites : $\alpha|V| + \beta|W|$.

5.1.3 Les instances du CELAR

À partir de données issues du terrain, le CELAR a généré 30 problèmes d'affectation de fréquences avec contraintes cumulatives, intitulés FAPPGx_y, où x dénote le numéro du problème et y représente le nombre de trajets. Les problèmes comportent de 16 à 2166 trajets, de 10 à 1229 contraintes impératives de type (4.3)…(4.2), de 16 à 4155 contraintes CEM binaires (4.6) et de 16 à 2015 contraintes CEM cumulative (5.1). Le tableau 5.1 illustre les caractéristiques de chaque instance.

FAPPG(1)	CIs(2)	CE(3)	CC(4)	FAPPG(1)	CIs(2)	CE(3)	CC(4)
01_0016	10	16	16	16_0038	21	128	37
02_0018	11	24	18	17_0040	22	92	36
03_0066	35	100	50	18_0052	28	116	42
04_0064	34	88	48	19_0770	387	2276	770
05_0064	34	80	64	20_1930	967	3896	1075
06_0182	93	245	172	21_1088	546	2789	1081
07_0182	93	245	172	22_0768	386	1604	757
08_0608	306	812	484	23_0034	19	53	24
09_1460	732	1862	1123	24_0048	38	80	39
10_1698	851	705	1292	25_0106	68	181	63
11_0164	84	439	101	26_0140	85	219	83
12_0902	453	2354	572	27_0154	92	255	134
13_0306	155	1142	244	28_0398	199	821	340
14_0194	99	587	163	29_0526	263	980	471
15_2454	1229	5135	2015	30_2166	1083	4155	1985

(1) Nom du problème FAPPGx_y, où x dénote le numéro du problème et y représente le nombre de trajets.
(2) Nombre de contraintes impératives du type (4.1), (4.2) (4.3) et (4.4).
(3) Nombre de contraintes CEM du type co-site (4.6)
(4) Nombre de contraintes CEM du type champ lointain (5.1)

TAB. 5.1 – Instances générées par le CELAR

5.2 Principes communes aux méthodes exactes proposées

Dans cette section, nous présentons les principes communs aux deux méthodes proposées. Dans la section 5.2.1 nous décrivons le schéma général de résolution décrivant les étapes enchaînées par les deux méthodes. La première d'entre elles, le prétraitement a une implémentation identique dans les deux méthodes et est présenté dans la section 5.2.2.

5.2.1 Schéma Général de résolution

La méthode de PLNE et la méthode hybride de PPC/PL présentées respectivement dans les sections 5.3 et 5.4 procèdent par trois phases :

Phase I : Pré-traitement. Le but de cette phase est de déduire des informations qui permettent de réduire la taille du problème (réduction des domaines des trajets, étude de la séparabilité du problème).

Phase II : Étude de la réalisabilité du problème. Le but de cette phase est d'essayer de donner une réponse à la question suivante : le problème possède t-il une solution réalisable ?

Phase III : Recherche de la solution optimale Selon la réponse obtenue dans la phase précédente, nous chercherons à trouver une solution qui minimise la largeur de la bande passante ou qui minimise la somme pondérée du nombre de contraintes de type CEM violées.

Dans ce qui suit, nous présentons dans la section 5.2.2 la phase I, commune aux deux méthodes.

5.2.2 Phase I : Pré-traitement

Nous avons implémenté trois méthodes de pré-traitement :

• **Réduction des domaines des trajets** : Nous utilisons les contraintes impératives duplex d'écart imposé (4.1) pour éliminer certaines valeurs des domaines des trajets. En effet, la procédure suivante peut être appliquée aux domaines de i et j indistinctement $\forall (i,j) \in CI_1$.

$$v \in F_{\varphi(i)}, \quad \text{si } \nexists w \in F_{\varphi(j)} : |f_i - f_j| = \epsilon_{ij} \Rightarrow F_{\varphi(i)} = F_{\varphi(i)} - v \tag{5.2}$$

$$w \in F_{\varphi(j)}, \quad \text{si } \nexists v \in F_{\varphi(i)} : |f_i - f_j| = \epsilon_{ij} \Rightarrow F_{\varphi(j)} = F_{\varphi(j)} - w \tag{5.3}$$

Les réductions des domaines résultantes pour 14 des 30 instances du CELAR s'illustrent dans le tableau 5.2 : Les résultats montrent que la procédure s'avérée très efficace sur 6

Scénario FAPPG	Taille de l'union des domaines	Nombre de valeurs éliminées	Pourcentage de valeurs éliminées
01_0016	10200	240	2.35%
02_0018	11600	240	2.07%
03_0066	39200	1280	3.26%
07_0182	108700	3520	3.24%
11_0164	97800	2240	2.29%
12_0902	540000	18320	3.39%
16_0038	3800	760	20.00%
17_0040	4000	800	20.00%
21_1088	533120	4352	0.81%
22_0768	299520	4608	1.54%
27_0154	9240	3080	33.33%
28_0398	99500	19900	20.00%
29_0526	210400	39976	19.00%
30_2166	866400	164616	19.00%

TAB. 5.2 – Réductions des domaines de 14 sur 30 instances générées par le CELAR

instances et qu'elle n'a eu aucun impact sur 16 instances. En fait, ce pré-traitement s'avère

efficace lorsque les trajets ont des domaines différents.

• **Irréalisabilité de contraintes cumulatives** : Nous avons implémenté un prétraitement qui vise à propager l'information donnée par les contraintes impératives duplex d'écart imposé (4.1) dans le contraintes cumulatives (5.1). En effet, les contraintes cumulatives (5.1) dépendent de l'écart entre les fréquences f_i et f_j et les contraintes d'écart imposé nous donnent la valeur exacte que doit respecter n'importe quelle solution qui satisfait ces contraintes. Nous profitons de l'information que contiennent ces contraintes, à savoir la valeur ϵ_{ij} et nous remplaçons cette valeur dans les contraintes (5.1) où ces fréquences interviennent afin d'obtenir des informations sur la réalisabilité du problème. Avec ce prétraitement nous avons prouvé que les instances FAPPG25_0106 et FAPPG27_0154 sont irréalisables. De plus, nous avons déterminé une borne inférieure pour ces deux instances égale à 2 contraintes cumulatives violées.

• **Séparabilité** : Nous pouvons identifier des sous-problèmes qui peuvent être résolus indépendamment les uns des autres sans altérer la solution du modèle en identifiant les composantes connexes dans le graphe de contraintes (section 4.4.1). Cette décomposition n'est pas directement possible dans la phase où le critère à optimiser est l'écart entre la fréquence maximale et la fréquence minimale des trajets alloués. En effet, l'écart entre la fréquence maximale et la fréquence minimale des trajets alloués peut faire intervenir deux composantes connexes différentes. Le tableau 5.3 illustre les résultats pour les instances tests.

Scénario FAPPG	Nombre de Composantes Connexes	Scénario FAPPG	Nombre de Composantes Connexes
01_0016	1	16_0038	1
02_0018	1	17_0040	1
03_0066	2	18_0052	1
04_0064	2	19_0770	1
05_0064	1	20_1930	136
06_0182	4	21_1088	1
07_0182	4	22_0768	2
08_0608	20	23_0034	1
09_1460	65	24_0048	1
10_1698	73	25_0106	1
11_0164	3	26_0140	2
12_0902	23	27_0154	1
13_0306	2	28_0398	9
14_0194	1	29_0526	17
15_2454	4	30_2166	46

TAB. 5.3 – Nombre de Composantes Connexes sur les 30 instances du CELAR

5.3 Une méthode de programmation linéaire en nombres entiers

Dans cette section nous présentons une méthode exacte qui utilise trois variantes d'une formulation de PLNE pour chacune des phases du schéma de résolution présenté en 5.2.1. La formulation utilisée pour l'étude de réalisabilité (phase II) est présentée en 5.3.1. La formulation pour la minimisation de la somme pondérée du nombre de contraintes violées (phase III) est présentée en 5.3.2. La formulation pour la minimisation de la largeur de la bande passante est présentée en 5.3.3. La méthode de résolution qui enchaîne les trois phases et présentée dans la section 5.3.4.

Les travaux présentés dans cette section sont basés sur [91, 100].

5.3.1 Un PLNE pour déterminer la réalisabilité du problème

Pour l'étude de la réalisabilité (phase II), nous proposons une formulation de PLNE en nous inspirant d'un modèle basé sur les variables indexées par la valeur des fréquences présenté en 4.4.2. Pour chaque trajet $i \in T$ et pour chaque valeur $v \in F_i$ possible pour f_i, ce modèle considère la variable binaire x_{iv} qui vaut 1 si et seulement si $f_i = v$. Pour modéliser les violations des contraintes du type (4.6), nous introduisons une variable continue c_{ij} qui vaut 0 si la contrainte associée à une paire $(i, j) \in CE$ est satisfaite. Enfin, pour la modélisation des violations des contraintes cumulatives (5.1), nous introduisons une variable continue $\Delta_i \geq 0$ qui vaut 0 si la contrainte associée à $i \in CC$ est satisfaite. Le

modèle suivant répond à la question : le problème possède t-il une solution réalisable ?

$$\min \quad \sum_{(i,j)\in CE} c_{ij} + \sum_{i\in CC} \Delta_i \tag{5.4}$$

s-à

$$\sum_{v\in F_{\varphi}(i)} x_{iv} = 1 \qquad \forall i \in T$$

$$x_{iv} \leq x_{j(v+\epsilon_{ij})} + x_{j(v-\epsilon_{ij})} \qquad \forall (i,j) \in \mathrm{CI}_1, \forall v \in F_{\varphi(i)}$$

$$x_{j(v+\epsilon_{ij})} + x_{j(v-\epsilon_{ij})} + x_{iv} \leq 1 \qquad \forall (i,j) \in \mathrm{CI}_2, \forall v \in F_{\varphi(i)}$$

$$x_{iv} + x_{jv} \leq 1 \qquad \forall (i,j) \in \mathrm{CI}_4, \forall v \in F_{\varphi(i)} \cap F_{\varphi(j)}$$

$$x_{iv} + \sum_{u\in V_{ijv}} x_{ju} \leq 1 + c_{ij} \qquad \forall (i,j) \in \mathrm{CE}, \forall v \in F_{\varphi(i)}, V_{ijv} \neq \emptyset \tag{5.5}$$

$$\sum_{j\in P_i} \lambda_{ij} \sum_{w\in F_{\varphi(j)}} T_{\sigma(i)\sigma(j)vw} x_{jw} \leq \Delta_i + (\Lambda_i + M_{iv}(1 - x_{iv})) \qquad \forall i \in \mathrm{CC}, \forall v \in F_{\varphi}(i) \tag{5.6}$$

$$\Delta_i \geq 0 \qquad \forall i \in \mathrm{CC} \tag{5.7}$$

$$0 \leq c_{ij} \leq 1 \qquad \forall (i,j) \in \mathrm{CE} \tag{5.8}$$

$$x_{iv} \in \{0,1\} \qquad \forall i \in T, \forall v \in F_{\varphi(i)} \tag{5.9}$$

où

$$M_{iv} = \sum_{j\in P_i} \lambda_{ij} \sum_{w\in F_{\varphi(j)}} T_{\sigma(i)\sigma(j)}(V_{ij}^{\min}) - \Lambda_i \qquad \forall i \in \mathrm{CC}, \forall v \in F_{\varphi(i)} \tag{5.10}$$

$$V_{ij}^{\min} = \min\{|\nu_1 - \nu_2| / \nu_1 \in F_{\varphi(i)} \wedge \nu_2 \in F_{\varphi(j)}\} \qquad \forall i \in CC, j \in P_i \tag{5.11}$$

Les quatre premières contraintes du modèle sont les contraintes impératives (4.10), (4.11), (4.12), (4.13) présentées dans la section 4.4.2. Notons que nous avons $\sum_{(i,j)\in CI_1} |F_{\varphi(i)}|$ contraintes impératives d'écart imposé (4.11), $\sum_{(i,j)\in CI_2} |F_{\varphi(i)}|$ contraintes impératives d'écart interdit (4.12) et $\sum_{(i,j)\in CI_4} |F_{\varphi(i)} \cap F_{\varphi(j)}|$ contraintes impératives de différence (4.13).

Les contraintes (5.5) correspondent aux contraintes CEM binaires classiques (4.6). CE désigne l'ensemble des paires de trajets pour lesquelles une contrainte de ce type existe. Nous renforçons la formulation (4.18) en introduisant V_{ijv} qui dénote l'ensemble de valeurs $u \in F_{\varphi(j)}$ telles que $|f_v - f_u| < \delta_{ij}$, c'est-à-dire les affectations de j violant la contrainte si $f_i = v$. Ainsi pour une contrainte d'écart (4.6) il y a autant de contraintes linéaires que de valeurs v possibles pour i telles que V_{ijv} soit non vide. La variable c_{ij} permet d'indiquer

si la contrainte est violée. En effet, l'écart δ_{ij} entre f_i et f_j est respecté si et seulement si $c_{ij} = 0$.

Les contraintes (5.6) correspondent aux contraintes CEM cumulatives (5.1). CC désigne l'ensemble des trajets intervenant en tant que trajet perturbé dans une contrainte cumulative. Soit $T_{\sigma(i)\sigma(j)vw} = T_{\sigma(i)\sigma(j)}(|v-w|)$, la valeur de la perturbation du trajet j sur le trajet i si $f_i = v$ et $f_j = w$. Alors, le membre de gauche de la contrainte représente la sommation des perturbations sur le trajet i lorsque f_i a la valeur v. Si $x_{iv} = 0$ la contrainte est toujours vérifiée si la constante M_{iv} est suffisamment grande. Nous cherchons à donner la plus petite valeur possible à cette constante pour l'expression (5.10). La variable continue Δ_i permet de vérifier si la contrainte est violée. En effet si $x_{iv} = 1$, la contrainte est vérifiée et la sommation des perturbateurs ne dépasse pas le seuil Λ_i si et seulement si $\Delta_i = 0$. Il y a $\sum_{i \in CC} |F_{\varphi(i)}|$ contraintes de ce type.

La fonction objectif (5.4) minimise une mesure d'interférence des violations des contraintes du type (4.6) et du type (5.1).

La résolution du programme ci-dessus donne une réponse à la réalisabilité du problème. Dans le cas où la réponse est positive (la valeur de la fonction objectif est 0) on obtient aussi une borne supérieure au problème de minimisation de la largeur de la bande passante. En effet, la solution trouvée est une allocation des fréquencess aux trajets et pour obtenir la valeur associée à cette solution il suffit de calculer l'écart entre les fréquences maximale et minimale allouées ($\theta = \max_{i \in T} f_i - \min_{i \in T} f_i$).

5.3.2 PLNE pour la minimisation de la somme pondérée des contraintes violées

Dans le cas où la réponse à la question de la phase II est négative, il se pose un problème dans le modèle présenté dans la section 5.3.1. En fait, $\sum_{i \in CC} \Delta_i$ ne compte pas le nombre de contraintes cumulatives violées (5.1) mais le degré de violations de ces contraintes cumulatives. Pour compter le nombre de contraintes cumulatives violées nous introduisons une variable binaire d_i qui vaut 0 si la contrainte cumulative $i \in CC$ est satisfaite et 1 si elle ne l'est pas. Alors, l'ensemble de contraintes cumulatives (5.6) peut être écrit de la forme suivante :

$$\sum_{j \in P_i} \lambda_{ij} \sum_{w \in F_{\varphi(j)}} T_{\sigma(i)\sigma(j)vw} x_{jw} \leq \Lambda_i + M_{iv}(1 - x_{iv} + d_i) \qquad \forall i \in \mathrm{CC}, \forall v \in F_{\varphi(i)} \quad (5.12)$$

Si $x_{iv} = 0$ la contrainte est toujours vérifiée si la constante M_{iv} est suffisamment grande. La variable binaire d_i permet de vérifier si la contrainte est violée. En effet si $x_{iv} = 1$, la contrainte est vérifiée et la sommation des perturbateurs ne dépasse pas le seuil Λ_i si et seulement si $d_i = 0$. Pour compter les contraintes du type (4.6) il suffit d'imposer aux variables $c_{ij}, \forall (i,j) \in CE$ à prendre les valeurs binaires $\{0,1\}$.

$$c_{ij} \in \{0,1\} \quad \forall (i,j) \in CE \tag{5.13}$$

Soit α le poids associé à la somme de contraintes violées et soit β le poids associé à la somme de contraintes cumulatives violées. Alors, le modèle à résoudre est le suivant, où la fonction objectif qui minimise la somme pondérée de contraintes violées est représentée par (5.14) :

$$\begin{aligned} \min \quad & \alpha \sum_{(i,j)\in CE} c_{ij} + \beta \sum_{i \in CC} d_i \\ \text{s.à} \quad & \\ & (4.10), (4.11), (4.12), (4.13), (5.5), (5.9), (5.12), (5.13) \end{aligned} \tag{5.14}$$

5.3.3 PLNE pour la minimisation de la largeur de la bande passante

Dans le cas où la réponse à la question de la phase II est positive, un autre programme linéaire en nombres entiers permet de représenter le problème de minimisation de la largeur de la bande utilisée, en posant $c_{ij} = 0, \forall (i,j) \in CEM$ et $d_i = 0, \forall i \in CC$. Nous rappelons ci-dessous la fonction objectif (4.15) et les contraintes 4.16 et 4.17 présentées à la section 4.4.4.

$$\begin{aligned} \min \quad & f_{\max} - f_{\min} \\ \text{s-à} \quad & (4.10), (4.11), (4.12), (4.13), (5.5), (5.9), (5.12) \\ & f_{\max} \geq \sum_{v \in F_{\varphi(i)}} v x_{iv} \qquad \forall i \in T \\ & f_{\min} \leq \sum_{v \in F_{\varphi(i)}} v x_{iv} \qquad \forall i \in T \\ & f_{\max} - f_{\min} \leq \theta \end{aligned} \tag{5.15}$$

La contrainte (5.15) impose une borne supérieure initiale (solution trouvée en résolvant le modèle proposé dans la section 5.3.1.

5.3.4 Procédure de séparation, évaluation et génération de coupes

La procédure de séparation, évaluation et génération de coupes[1] est souvent utilisé pour résoudre des problèmes en nombres entiers. Considérons le modèle de la section 5.3.1 sans les contraintes cumulatives (5.6). Tout d'abord, nous résolvons la relaxation linéaire associée à ce modèle. Deux cas sont possibles :

- La solution contient des variables dont les valeurs ne sont pas entières. Alors, nous utilisons une procédure de branchement jusqu'à l'obtention d'une solution entière.
- La solution est entière. Dans ce cas, nous vérifions si celle-ci satisfait les contraintes cumulatives (5.6). Si ce n'est pas le cas, alors nous utilisons une "méthode de coupe" pour couper cette solution de l'espace de recherche. Si la solution satisfait les contraintes cumulatives (5.6), alors nous avons prouvé que le problème est réalisable et en plus nous avons trouvé une première solution pour la phase d'optimisation de la largeur de la bande passante.

Nous appliquons ensuite le même principe pour la phase III d'optimisation comme décrit dans l'algorithme 5. Nous détaillons ci-après les stratégies de séparation et de génération de plans coupants utilisées.

Stratégie de séparation

Nous proposons deux schémas de sélection de séparation :

- Deux types de sélection de variables ont été testés : sélection de la variable dont la valeur est "plus fractionnaire" (près de 0.5) et sélection de la variable dont la valeur est plus proche à un entier (0 où 1, dans notre cas). Dans les deux cas, nous fixons la variable à la valeur 1. Notons qu'à chaque niveau de l'arbre de recherche nous branchons sur une seule variable.
- Branchement de type GUB ("Generalized Upper Bound") : à notre connaissance, c'est la première fois qu'une telle stratégie de branchement est utilisée pour le problème d'allocation de fréquences. En effet, il s'agit de brancher sur plusieurs variables à chaque niveau de l'arbre de recherche en les fixant à la valeur zéro. Ce type de branchement correspond à une recherche dichotomique. L'idée est la suivante : Si

[1] Branch and Cut

la solution courante x^* a certaines variables $x_1^*, , x_k^*$ fractionnaires, alors la règle de séparation standard propose de partitionner S^2 en $S_1 = S \cap \{x/x_j = 0\}$ et $S_2 = S \cap \{x/x_j = 1\}$ pour un certain $j \in \{1, \ldots, k\}$. Si nous appliquons ce principe à la contrainte (4.10), l'ensemble $S_1 = \{x/x_j = 0\}$ contient $k-1$ possibilités de fixer une variable $i \neq j$ à 1 ($\{x/x_i = 1\}_{i \neq j}$) alors que $S_2 = \{x/x_j = 1\}$ contient une seule possibilité et l'arbre de recherche n'est pas équilibré. Notre stratégie est donc, de diviser S en deux ensembles S_1 et S_2 qui soient approximativement de la même taille, par exemple :

⋆ $S_1 = S \cap \{x/x_{ji} = 0 | i = 1, \ldots, r\}$ et $S_2 = S \cap \{x/x_{ji} = 0 | i = r+1, \ldots, k\}$ où $r = \min\{t : \sum_{i=0}^{t} x_{ji}^* \geq \frac{1}{2}\}$.

Le tableau 5.4 compare les deux schémas de branchement pour l'étude de réalisabilité, c'est-à-dire la résolution du PLNE présentée à la section 5.3.1 sur un ensemble d'instances du CELAR. Notons que pour 7 des 9 instances étudiées ici un temps limite d'exécution a

Scénario FAPPG	Branchement GUB, objectif réalisabilité			Branchement Standard, objectif réalisabilité		
	Nb.Noeuds	Objectif	Temps	Nb.Noeuds	Objectif	Temps
01_0016	1939	0.3414	TiLim	1651	0.1585	TiLim
02_0018	318	9.1254	TiLim	748	8.1880	TiLim
16_0038*	552	670.7920	TiLim	824	1038.8997	TiLim
17_0040*	972	98.6458	TiLim	989	119.3641	TiLim
23_0034	7	0	2.58	35	0	10.6
24_0048	2966	0	906.23	621	0	287.4
25_0106*	6824	0.0489	TiLim	12337	0.0489	TiLim
26_0140	2059	2.1273	TiLim	1101	23.9919	TiLim
27_0154*	4443	0.3441	TiLim	4152	0.7246	TiLim

Tab. 5.4 – Comparaison des Stratégies de branchement

été atteint, ce qui ne permet pas de conclure sur la réalisabilité. Néanmoins, le branchement GUB s'avère intéressant comme l'atteste la valeur de la meilleure borne supérieure obtenue au bout du temps limite. Ces bornes supérieures ne sont pas entières car la fonction objectif contient de sommes de degrés de violations (Δ_i) des contraintes cumulatives. Pour 6 instances sur 9, le branchement GUB obtient de meilleurs résultats. Un autre avantage de ce type de branchement est une meilleure gestion de la mémoire.

Stratégie de coupes

Nous rappelons que nous n'ajoutons pas les contraintes cumulatives (5.6) à la racine de l'arbre de recherche. Lorsque nous avons trouvé une solution entière, nous devons vérifier si

[2] Où S est l'ensemble de contraintes restant du modèle de la section 5.3.1

elle satisfait les contraintes cumulatives. Pour chaque trajet $i \in CC$, nous nous intéressons à une seule contrainte cumulative, celle liée à la valeur $v \in F_{\varphi(i)}$ telle que $x_{iv} = 1$. En effet, si $x_{iv} = 0$, la contrainte cumulative est inactive. Si une contrainte cumulative n'est pas satisfaite nous pouvons ajouter au modèle de la section 5.3.1 la contrainte (5.6).

Cependant, un problème se pose dans la résolution des modèles PLNE pour le traitement des contraintes cumulatives. En fait, le grand M donne des relaxations linéaires si mauvaises qu'il est nécessaire de les gérer d'une autre manière. Nous proposons ainsi de représenter les contraintes cumulatives (5.1) par des contraintes dites de "cover inequalities" :

$$x_{iv} + \sum_{(j,w)\in P} x_{jw} \leq |P| + d_i, \forall i \in T, \forall v \in F_i, \forall P \in \mathcal{P}_{iv} \tag{5.16}$$

où $P \in \mathcal{P}_{iv}$ est un ensemble de couples (trajet,valeur) violant la contrainte cumulative pour $f_i = v$[3]. L'inconvénient de ces contraintes est leur nombre, en théorie exponentiel. Leur utilisation au sein d'un algorithme de coupe permet de réduire le nombre de contraintes ajoutées à chaque étape. Cependant, les modèles présentés dans les sections 5.3.2 et 5.3.3 sont résolus en considérant ces contraintes, c'est-à-dire les contraintes (5.16).

Enfin, l'algorithme 5 illustre la procédure de résolution du principe général décrit dans la section 5.2.1 sans considérer la phase I.

Algorithme 5 Procédure de séparation, évaluation et génération de coupes pour le PAF.

1: Résoudre la relaxation de programmation linéaire sans les contraintes cumulatives (5.12) du modèle de la section 5.3.1, aller à 2.
2: Si la meilleure borne inférieure est strictement supérieure à 0, alors le problème est prouvé irréalisable. Aller à 6.
3: Si la solution est fractionnaire brancher et revenir à 1. Si une solution entière est trouvée aller à 4.
4: Si la solution entière trouvée satisfait les contraintes cumulatives (5.1) aller à 7. Autrement aller à 5.
5: Ajouter les contraintes violées (5.6) au modèle et résoudre le programme linéaire correspondant, revenir à 2.
6: Changer le modèle par celui qui est proposé à la section 5.3.2. Aller à 1 avec ce nouveau modèle sans exécuter l'étape 2.
7: Changer le modèle par celui qui est proposé à la section 5.3.3. Aller à 1 avec ce nouveau modèle sans exécuter l'étape 2.

Les résultats expérimentaux de cet algorithme sont illustrés dans la section 5.5.

[3]Nous vous rappelons qu'il suffit de fixer $d_i = 0$ pour le cas où nous minimisons la largeur de la bande passante.

5.4 Une méthode hybride PPC/PL

D'après l'état de l'art effectué au chapitre 4, la programmation par contraintes a été très peu utilisée en affectation de fréquences. Nous proposons, dans ce chapitre, d'utiliser la PPC hybridée avec la résolution d'une relaxation du problème par programmation linéaire. Nous présentons cette relaxation en 5.4.1 et le modèle PPC en 5.4.2. La méthode hybride est présentée en 5.4.3.

Cette approche a été présentée dans [88].

5.4.1 Relaxation par la Programmation Linéaire

Le but de la relaxation considérée est d'étudier la réalisabilité du problème. Cette relaxation travaille sur le graphe de contraintes renforcé par la déduction de contraintes électromagnétiques du type (4.6) à partir de contraintes cumulatives (5.1).

Nous présentons la méthode de déduction, puis la relaxation considérée et sa résolution par programmation linéaire.

Méthode de déduction

On considère les contraintes cumulatives (5.1) :

$$\sum_{j \in P_i} \lambda_{ij} T_{\sigma(i)\sigma(j)}(|f_i - f_j|) \leq \Lambda_i \quad \forall i \in CC \tag{5.17}$$

Pour chaque $j \in P_i$, nous pouvons déduire de contraintes électromagnétiques du type (4.6) de la manière suivante :

$$|f_i - f_j| \geq \min_{e \in IN} \{e | \lambda_{ij} T_{\sigma(i)\sigma(j)}(e) \leq \Lambda_i\} \quad \forall j \in P_i, i \in CC \tag{5.18}$$

Nous construisons le graphe de contraintes associé au problème comportant :

- les contraintes binaires CEM co-sites d'origine (4.6),
- les contraintes impératives d'écart imposé (4.11) et
- les contraintes CEM binaires déduites des contraintes cumulatives (5.18).

Résolution par la Programmation Linéaire

Soit G le graphe de contraintes et soit $C(G,k)$ une k-clique de G. Nous résolvons la relaxation du problème de voyageur de commerce associé à $C(G,k)$ connue sous le nom de couplage parfait (voir 4.4.4) en utilisant la programmation linéaire. Soit $v(cp)$ la valeur de la fonction objectif du problème du couplage parfait. Il est bien connu que la valeur de la fonction objectif de ce problème est une borne inférieure sur le problème de la largeur de la bande passante. Soit

$$\chi = \max\{f : f \in \bigcup_{k=1}^{k=N} F_{\varphi(k)}\} - \min\{f : f \in \bigcup_{k=1}^{k=N} F_{\varphi(k)}\}$$

Théorème 5.1 *Si $v(cp) > \chi$ alors le problème est irréalisable.*

Dans ce cas, il n'existe pas une fréquence maximale et minimale dans l'union de domaines fréquentiels qui puisse satisfaire toutes les contraintes du problème.

Dans ce qui suit, nous présentons un modèle basé sur la programmation par contraintes du problème décrit dans la section 5.1.2.

5.4.2 Modèle PPC

Nous présentons un modèle basé sur la programmation par contraintes qui comporte les variables de décision suivantes :

- f_i pour tout $i \in T$ de domaine $F_{\varphi(i)}$. Ces variables représentent directement les fréquences à affecter à chaque trajet.
- d_{ij} pour toute paire (i,j) impliquées dans une contrainte. d_{ij} représente la distance $|f_i - f_j|$ entre la fréquence affectée au trajet i et la fréquence affectée au trajet j.
- t_{ij} pour tout (i,j) impliquées dans une contrainte cumulative CC. t_{ij} représente la valeur effective de la fonction de perturbation.

Alors, la modélisation des contraintes (définies dans la section 5.1.2) liant ces variables de décision sont les suivantes :

$$d_{ij} = |f_i - f_j| \qquad \forall i, j \in T \tag{5.19}$$

$$d_{ij} = \epsilon_{ij} \qquad \forall (i,j) \in CI_1 \tag{5.20}$$

$$d_{ij} \neq \epsilon_{ij} \qquad \forall (i,j) \in CI_2 \tag{5.21}$$

$$d_{ij} \geq \delta_{ij} \qquad \forall (i,j) \in CE \tag{5.22}$$

$$t_{ij} = T_{\sigma(i)\sigma(j)}[d_{ij}] \qquad \forall (i,j) \in CC \tag{5.23}$$

$$\sum_{j \in P_i} \lambda_{ij} \sum_{w \in F_{\varphi(j)}} t_{ij} \leq \Lambda_i \quad \forall i \in CC \tag{5.24}$$

La contrainte (5.23), appelée contrainte élément (voir section 1.4, page 8), impose à la variable t_{ij} d'être l'élément d'un tableau (la fonction de perturbation) indicé par une autre variable de décision : la distance d_{ij}.

5.4.3 Résolution basée sur le modèle hybride

Phase II : Étude de la réalisabilité du problème

Pour l'étude de la réalisabilité, nous utilisons la relaxation proposée en 5.4.1 et nous proposons la résolution de plusieurs cliques qui contiennent des contraintes (arêtes) d'écart (poids) au moins égale à k, où k est variable. Il n'existe pas un moyen pour déterminer a priori la "meilleure" clique qui nous donnera la meilleure valeur du problème de couplage parfait. La générations de cliques est faite en utilisant une heuristique très simple. Cette heuristique ajoute des trajets selon l'ordre décroissant des degrés (le nombre de contraintes liées à ce trajet). Il faut noter, que les seules contraintes prises en compte (dans le degré d'un trajet) sont celles qu'ont un poids minimum de k. Enfin, pour chaque clique trouvée nous calculons la valeur optimale du problème de couplage parfait. La valeur optimale la plus grande est choisie pour tester la réalisabilité du problème selon le théorème 5.1. Si cette valeur est plus grande que l'écart maximal de l'union de domaines fréquentiels, alors nous déduisons que le problème est irréalisable. Dans le cas contraire, nous ne pouvons rien déduire.

Phase III : Recherche de la solution optimale

Dans cette section, nous nous consacrons à la recherche de la solution optimale (phase III) par une recherche arborescente utilisant la relaxation de programmation linéaire et basée sur la programmation par contraintes.

Dans le cas où le problème est irréalisable, nous calculons à la racine de l'arbre une borne inférieure sur le nombre de contraintes violées en utilisant la différence entre la valeur optimale de la k-clique qui prouve l'irréalisabilité et l'écart maximal de l'union de domaines fréquentiels. L'algorithme 6 illustre le calcul de cette borne.

Algorithme 6 Cacul d'une borne inférieure pour le PAF non réalisable.

Soit y un vecteur de contraintes ordonné selon l'ordre décroissant des écart (poids).
Soit $BIR = 0$ la borne inférieure du nombre de contraintes violées.
Soit $v(cp)^*$ la valeur optimale de la meilleure k-clique qui a dit que le problème est irréalisable.
Soit *ecart* l'écart maximal entre la plus grande fréquence et la plus petite fréquence de l'union de domaines fréquentiels.
Soit diff $= v(cp)^* - ecart$ l'écart entre la valeur optimale de la k-clique et l'écart maximal.

```
1: i=0
2: tant que diff>0 faire
3:    BIR = BIR + 1
4:    diff = diff - y[i]
5:    i = i + 1
6: Fin tant que
```

Dans le cas où le problème est réalisable, nous avons deux bornes. Une borne supérieure qui est obtenue par la valeur de la solution entière courante qui satisfait toutes les contraintes et une borne inférieure donnée par la valeur optimale de la meilleure k-clique.

Résolution du problème par recherche arborescente

Pour les deux objectifs possibles de la phase II, le problème est résolu par application d'une recherche arborescente, par l'intermédiaire des "goals" d'Ilog Solver [62]. Le goal spécifie le travail effectué à un nœud de l'arbre de recherche et notamment, quelle est la variable à affecter (parmi les f_i) puis quelle est la valeur (ou l'ensemble de valeurs) à explorer en premier pour cette variable (chaque nœud fils du nœud courant correspond à une valeur ou un sous-ensemble de valeurs possibles pour la variable sélectionnée). La règle de sélection de la variable ainsi que la règle de sélection du nœud fils à explorer en premier constituent l'heuristique de branchement. Par ailleurs, à chaque nœud, un algorithme de filtrage permet de réduire les domaines des autres variables suite à la décision de

branchement, par propagation de contraintes. Enfin, à chaque nœud la relaxation présentée en 5.4.1 est résolue par programmation linéaire dans le but de prouvée l'irréalisabilité du nœud. Nous présentons ci-après l'heuristique de branchement et la méthode de filtrage utilisée.

- ***Heuristique de branchement***

Nous avons sélectionné la variable f_i qui a le plus petite domaine et nous avons branché sur la valeur la plus petite de son domaine. La recherche de la solution optimale se base sur la résolution d'une série de problèmes de réalisabilité où la contrainte de la fonction objectif est mise à jour lorsqu'une nouvelle borne supérieure est trouvée.

- ***Algorithmes de filtrage***

Nous avons testé les algorithmes de filtrage par défaut de Ilog Solver, c'est à dire l'algorithme d'Arc-Consistance AC5 dont le principe est le suivant. Un ensemble de contraintes binaires C est considéré, composé des contraintes binaires du problème initial et des contraintes introduites par branchement. L'algorithme d'Arc-Consistance consiste à sélectionner une contrainte c dans C puis à appliquer un algorithme de filtrage à c (c'est-à-dire de suppression des valeurs inconsistantes dans le domaine des deux variables intervenant dans la contrainte). L'algorithme de filtrage renvoie une liste L des variables dont le domaine a été ainsi modifié. c est enlevée de C puis chaque contrainte impliquant chaque variable x de L est ajoutée dans C si elle n'y est pas déjà. L'algorithme s'arrête quand C est vide. Plus de détails peuvent être trouvés dans l'article de Van Hentenryck et al. [51]

Dans ce qui suit, nous présentons les principaux résultats expérimentaux.

5.5 Résultats

Toutes les procédures ont été développées sur un Pentium II cadencé à 350MHz et possédant 256 Mo de RAM. Le système d'exploitation est Windows 98 DE, le langage orienté objet est le C++, compilé avec Microsoft Visual C++ 6.0. Pour la modélisation du PSDM, nous avons utilisé la librairie d'ILOG CONCERT version 1.2 qui fait office d'interface avec ILOG CPLEX version 7.5 pour l'optimisation des programmes linéaires, et d'interface avec ILOG SOLVER version 5.2 pour la résolution de la programmation par contraintes.

Scénario[(1)] FAPPG	NVCC[(2)]	NVCEM[(3)]	NCCCuts[(4)]	NNœuds[(5)]	Bande[(6)]	T (sec)[(7)]
01_0016[c]	0	0	57	219	640	1443
02_0018	4	0	383	1292	699	11745
03_0066	12	6	419	1042	655	13677,3
04_0064	0	3	58	88	767	4893,16
05_0064	26	7	145	154	799	5594,04
06_0182	120	65	71	99	799	995,45
07_0182	117	68	74	80	699	939,37
08_0608	201	87	494	1411	848	8235,43
11_0164	52	52	48	58	682	555,93
13_0306	177	135	72	87	399	1561,92
14_0194	92	120	58	102	399	729,52
16_0038[a]	38	31	273	1654	149	3599
17_0040[a]	26	25	146	1068	98	1688
18_0052	99	59	13	29	496	7588
22_0768	635	283	224	85	898	1924,84
23_0034[c]	0	0	18	7	540	2,77
24_0048[c]	0	0	99	138	540	81,97
25_0106[b]	10	0	654	9178	540	11952,9
26_0140	8	5	842	7268	496	12444,4
27_0154[b]	47	1	1396	4783	490	12361,8
28_0398	23	18	1323	3901	698	43358
29_0526	152	84	716	2078	912	21473,3

(1) Scénario généré par le CELAR.
(2) Nombre de contraintes de type (5.1) non satisfaites par la meilleure solution trouvée.
(3) Nombre de contraintes de type (4.6) non satisfaites par la meilleure solution trouvée.
(4) Nombre de coupes de type (5.16) ajoutées au modèle pendant la recherche.
(5) Nombre de nœuds générés pour trouver la meilleure solution.
(6) La largeur de la bande passante de la meilleure solution.
(7) Le temps en secondes pour trouver la meilleure solution.

Tab. 5.5 – Résultats expérimentaux de la méthode exacte basée sur la PLNE

5.5.1 Résultats expérimentaux de la méthode exacte PLNE

Le tableau 5.5 montre les résultats obtenus par la méthode exacte de PLNE avec un critère d'arrêt de 10800 secondes, sauf pour les dernières 5 instances où le critère d'arrêt est la taille de la mémoire. Tout d'abord, 8 instances n'ont pas été résolues du fait de leur grande taille (nombre élevé de variables et de contraintes). Les problèmes marqués par (b) ont été prouvés irréalisables par le seul pré-traitement des contraintes cumulatives. En effet, dans ces deux cas, une contrainte impérative d'égalité n'est pas compatible avec une contrainte cumulative. Les problèmes (a) ont été prouvés irréalisables à la racine de l'arbre de recherche par la résolution de la relaxation linéaire du modèle présenté à la section 5.3.1. Ceci signifie qu'il y a des incompatibilités entre les contraintes CEM binaires et les contraintes impératives (nous ne générons pas de contraintes cumulatives à la racine). Trois instances (marquées par (c)) ont été prouvées réalisables par la résolution exacte du modèle de la section 5.3.1. Cependant, pendant la recherche de la solution optimale, la solution obtenue dans la phase II n'a pas été améliorée. Pour les instances FAPPG01_0016 et FAPPG23_0034 nous avons laissé tourner la machine pendant 24 heures ; la valeur de la solution obtenue pour la FAPPG01_0016 est égale à 548 et pour la FAPPG23_0034 à 380 mais sans prouver l'optimalité de ces solutions. Les résultats, à quelques exceptions près, ceux-ci sont très mauvais. En effet, pour résoudre plus de 9 instances, on a dû séparer (partitionner) certains problèmes en k sous-problèmes avec, pour conséquence, la détérioration de la solution. Tous les problèmes ont été résolus en utilisant une stratégie de branchement type GUB.

Un des principaux inconvénients est la très mauvaise borne inférieure de la relaxation linéaire (phase d'optimisation de la largeur de la bande passante) de la plupart des instances. En fait, la borne inférieure obtenue à la racine est égale à 0. Une borne inférieure triviale pour le problème est $BI = \max(\epsilon_{ij}), \forall (i,j) \in CI_1$. D'autre part, la solution trouvée est dégénérée ce qui ne facilite pas les choses. Enfin, nous pouvons constater que le nombre de coupes ajoutées dans la résolution, des instances prouvées réalisables, des modèles est petite par rapport à son grand nombre, en théorie exponentiel.

5.5.2 Résultats expérimentaux de la méthode exacte hybride PPC/PL

Le tableau 5.6 montre les résultats obtenus par la méthode hybride PPC/PL. Cette fois, 5 instances ont été prouvées réalisables par la PPC, ce qui revient à dire que la valeur optimale du problème de couplage parfait lié à la meilleure k-clique devient ainsi une borne inférieure pour le problème ; cette borne inférieure est donnée à la colonne (3) pour les instances identifiées par la lettre (a). La colonne (4) montre, pour ces mêmes instances,

une borne supérieure de la largeur de la bande passante.

D'autre part, les 4 instances identifiées par (b) sont prouvées irréalisables en utilisant le théorème 5.1. Pour ces instances, la colonne (5) donne une borne inférieure du nombre de contraintes violées.

Si nous comparons les tableaux 5.5 et 5.6, nous constatons que l'approche hybride obtient de meilleurs résultats que l'approche de PLNE pour 23 problèmes sur 30, en un temps de calcul nettement inférieur. Néanmoins, la PLNE est supérieure pour 5 problèmes. Ceci illustre que malgré ses résultats globalement mauvais la PLNE ne peut être écartée.

L'approche hybride a su combiner les avantages d'une relaxation de programmation linéaire qui a pu prouver l'irréalisabilité avec ceux de la programmation par contraintes pour la recherche de solutions.

Le tableau 5.5.2 compare nos meilleurs résultats avec les résultats obtenus par les heuristiques de résolution à grand voisinage (LNS) [98, 99], la méthode tabou basée sur un voisinage consistant (TA) [122] et la méthode de recuit simulé (SA) [108]. Même si notre méthode est loin des meilleures solutions obtenues par les heuristiques, nous trouvons la meilleure solution de largeur de bande pour 3 instances. De plus, la borne inférieure que nous proposons prouve l'optimalité des instances FAPPG25_0106 et FAPPG27_0154 (voir le tableau 5.5).

FAPPG[(1)]	EM[(2)]	V(cp)[(3)]	Bande[(4)]	BI(BS)[(5)]	TpT[(6)] (sec)	TRPL[(7)] (sec)	TCBI[(8)]	NCC[(9)]
01_0016[a]	699	321	548	0(0)	97	0,04	8	1
02_0018[a]	699	505	650	0(0)	15	0,09	8	1
03_0066	699	400	***	0(43)	0,16	0,04	6(1)	2
04_0064[a]	799	401	547	0(0)	206	0,03	8(1)	2
05_0064	799	453	***	0(7)	0,33	0,12	8	1
06_0182	799	396	***	0(20)	1,42	0,1	6(2)	4
07_0182	699	396	***	0(176)	1,54	0,13	6(2)	4
08_0608	848	318	***	0(15)	46,6	0,42	5(1)	20
09_1460	898	500	***	0(21)	837,17	1,41	6(1)	65
10_1698	1000	480	***	0(23)	1513,56	0,77	4(1)	73
11_0164	699	375	***	0(5)	0,68	0,06	6(2)	3
12_0902	699	423	***	0(33)	91,25	0,33	7(9)	23
13_0306	399	319	***	0(120)	2,37	0,05	5(1)	2
14_0194	399	184	***	0(265)	0,55	0,02	4(1)	1
15_2454	399	240	***	0(2823)	181,39	0,08	4(1)	4
16_0038[b]	149	I(398)	***	1(69)	0,27	0,16	17	1
17_0040[b]	99	I(238)	***	2(51)	0,03	0,08	4	1
18_0052	496	344	***	0(20)	608,1	0,01	6	1
19_0770[b]	496	I(800)	***	4(882)	38,8	0,68	11	1
20_1930[b]	496	I(536)	***	1(285)	2005,82	2,04	9(17)	136
21_1088	998	478	***	0(383)	40,82	0,11	7	1
22_0768	898	343	***	0(157)	20,23	0,13	10(1)	2
23_0034[a]	540	300	380	0(0)	1	0,02	2	1
24_0048[a]	540	300	410	0(0)	10	0,03	2	1
25_0106[c]	540	I PréTr.	***	2(29)	0,13	0	****	1
26_0140	496	304	***	0(88)	15	0,04	6(1)	2
27_0154[c]	495	I PréTr.	***	2(82)	0,49	0	****	1
28_0398	698	400	***	0(94)	8,3	0,23	2	9
29_0526	918	412	***	0(131)	1422,35	0,42	6(2)	17
30_2166	918	774	***	0(871)	1284,99	1,88	11(6)	46

(1) Nom de l'instance générée par le CELAR.
(2) Écart entre la fréquence maximale et la fréquence minimale de l'union de domaines.
(3) La valeur optimale de la meilleure k-clique générée.
(4) La meilleure borne supérieure de la largeur de la bande passante trouvée si le problème est prouvé réalisable.
(5) La borne inférieure (la meilleure borne supérieure) du nombre de contraintes violées si le problème a été prouvé irréalisable.
(6) Le temps de calcul en secondes de la meilleure solution.
(7) Le temps moyen en secondes pour la résolution du problème de couplage parfait.
(8) Taille de la clique de meilleure valeur relâchée (Entre parenthèses la composante où elle est trouvée).
(9) Le nombre de composantes connexes.

Tab. 5.6 – Résultats de la réalisabilité dans la méthode hybride PPC/PL

instance	LNS [99]		SA [108]		TS [122]		PLNE PPC/PL	
	Nb Violées	bande	Nb Violées	bande	Nb Violées	bande	Nb Violées	bande
FAPPG01_0016	**0**	**548**	0	549	**0**	**548**	**0**	**548**
FAPPG02_0018	**0**	**629**	**0**	**629**	**0**	**629**	0	650
FAPPG03_0066	2	599	**2**	**580**	2	623	43	***
FAPPG04_0064	0	520	0	520	**0**	**519**	0	547
FAPPG05_0064	**0**	**599**	0	623	0	676	7	***
FAPPG06_0182	**0**	**718**	**0**	**718**	0	758	20	***
FAPPG07_0182	6	666	**4**	**698**	8	687	176	***
FAPPG08_0608	**0**	**620**	0	646	0	642	15	***
FAPPG09_1460	**0**	**544**	0	656	0	860	21	***
FAPPG10_1698	**0**	**412**	0	692	0	849	23	***
FAPPG11_0164	0	604	0	656	**0**	**601**	5	***
FAPPG12_0902	**0**	**572**	1	639	2	634	33	***
FAPPG13_0306	9	399	**6**	**399**	8	380	120	***
FAPPG14_0194	0	398	0	360	**0**	**354**	265	***
FAPPG15_2454	**31**	**399**	73	399	44	399	2823	***
FAPPG16_0038	**46**	**146**	**46**	**146**	**46**	**146**	69	***
FAPPG17_0040	46	99	**45**	**98**	**45**	**98**	51	***
FAPPG18_0052	**0**	**404**	1	476	0	408	20	***
FAPPG20_1930	**150**	**492**	193	496	152	496	285	***
FAPPG21_1088	**0**	**982**	2	964	4	994	383	***
FAPPG22_0768	**0**	**788**	0	818	1	894	157	***
FAPPG23_0034	**0**	**380**	**0**	**380**	**0**	**380**	**0**	**380**
FAPPG24_0048	**0**	**410**	0	430	**0**	**410**	**0**	**410**
FAPPG25_0106	2	540	**2**	**490**	2	540	10	***
FAPPG26_0140	**0**	**480**	1	492	**0**	**480**	13	***
FAPPG27_0154	**2**	**490**	4	490	**2**	**490**	48	***
FAPPG28_0398	**0**	**610**	0	646	0	638	41	***
FAPPG29_0526	**0**	**542**	0	852	0	866	131	***
FAPPG30_2166	**23**	**912**	27	912	32	912	871	***

TAB. 5.7 – Comparaison des bornes supérieures avec les meilleures heuristiques

Conclusions et perspectives

Pour chacune des deux parties du mémoire, nous établissons une synthèse des différents aspects sur lesquels portent nos contributions avant d'évoquer les perspectives de recherches ouvertes par le travail présenté.

Dans la première partie du mémoire, nous nous sommes intéressés aux problèmes de sac à dos multidimensionnel en variables binaires. L'étude de ce type de problèmes a connu un fort intérêt vis à vis de son application à plusieurs problèmes pratiques. Nous avons proposé une méthode exacte de type hybride PPC/PL pour résoudre ce problème. Nous pensons que c'est la première fois qu'une approche de ce type est proposé comme le confirme l'état de l'art récent de Fréville [32]. Malheureusement, les expérimentations sont loin de conforter nos espoirs quant au comportement de notre méthode relativement à celui de CPLEX. CPLEX est beaucoup plus rapide en dépit d'un arbre de recherche et d'un nombre d'itérations du simplexe largement supérieurs à notre méthode. Cela laisse à penser que l'ajout de la PPC à la PL pour résoudre ce type de problème ne permet de couper que des nœuds de bas de l'arbre de recherche. Ainsi, propager les contraintes dans le haut de l'arbre ne représente qu'une perte de temps.

Nous pensons cependant que certaines pistes restent à envisager : mieux propager la contrainte "ensemble", développer des schémas de branchement plus adaptés à la PPC, développer des algorithmes de propagation de contraintes plus adaptés au problème.

Néanmoins, il est possible que sur d'autres types de problèmes les approches présentées dans ce mémoire soient plus performantes que CPLEX : nous pensons en particulier aux problèmes où les méthodes de résolution présentées dans la littérature sont, en proportions équivalentes, soit de type programmation mathématique, soit de type programmation par contraintes (l'ordonnancement par exemple).

Dans la deuxième partie de ce mémoire, nous nous sommes intéressés à un cas concret d'allocation de fréquences. Nous avons utilisé une modélisation originale des contraintes de compatibilité électro-magnétique sous la forme de contraintes cumulatives. Nous avons

proposé une formulation de PLNE de ce problème et deux méthodes pour le résoudre, à savoir : une méthode basée sur la PLNE et une autre méthode qui combine la programmation par contraintes et la programmation linéaire. Même si les résultats obtenus sont loin des résultats des heuristiques, les méthodes proposées restent intéressantes. En effet, elles sont en mesure de prouver, d'une part, si les problèmes comportent ou pas des solutions et, d'autre part, l'optimalité des résultats. Enfin, notre contribution est d'avoir proposé une des premières méthodes exactes pour la minimisation de la largeur de la bande passante ainsi que la première méthode exacte hybride PPC/PL en affectation de fréquences. Les bons résultats de cette méthode en comparaison avec ceux de la PLNE montrent l'intérêt de continuer dans cette voie.

Les perspectives de travail sont nombreuses. Nous envisageons de modéliser les contraintes cumulatives en utilisant les fonctions linéaires par morceaux dans une approche hybride. De même, nous envisageons de développer des algorithmes de filtrages qui utilisent les cliques du graphe de contraintes et également des algorithmes de filtrage inspirés de la méthode de pré-traitement des contraintes d'écart imposé sur les contraintes cumulatives.

Bibliographie

[1] Challenge ROADEF 2001. `http://http://www.prism.uvsq.fr/~vdc/ROADEF/CHALLENGES/2001/`.

[2] K. Aardal, S. Van Hoesel, A. Koster, C. Mannino, and A. Sassano. Models and solution techniques for frequency assignment problems. *4OR Quaterly Journal of the Belgian, French and Italian Operations Research Societies*, 1(4) :216–317, 2003.

[3] K. Aardal, C. Hurkens, J.K. Lenstra, and S. Tiourine. Algorithms for radio link frenquency assignment : The calma project. *Operations research.*, 50(6) :968–980, 2002.

[4] E.L. Aarts and J.K. Lenstra. *Local Search in Combinatorial Optimization.* Princeton University Press, 2003.

[5] P. Adjakplé. *Affectation de canaux dans les réseaux de téléphone mobile cellulaire.* PhD thesis, École Polytechnique de Montréal, Montréal, Canada, 1998.

[6] S. M. Allen, D. H. Smith, and S. Hurley. Lower bounding techniques for frequency assignment. *Discrete Mathematics*, 197/198 :41–52, 1999.

[7] J.-M. Alliot and T. Schiex. *Intelligence Artificielle and Informatique theorique.* Cépaduès Éditions, 1993.

[8] B. De Backer, V. Furnon, P. Shaw, P. Kilby, and P. Prosser. Solving vehicle routing problems using constraint programming and metaheuristics. *Journal of Heuristics*, 6(4) :501–523, 2000.

[9] B. De Backer, P. Kilby, and P. Prosser. Solving vehicle routing problems using constraint programming and methaheurisics. *Journal of Heuristics*, 6 :501–523, 2000.

[10] E. Balas. An additive algorithm for solving linear programs with zero-one variables. *Operations Research*, 13 :517–546, 1965.

[11] E. Balas and C.H. Martin. Pivot and complement-a heuristics for 0-1 programming. *Management Science*, 26 :86–96, 1980.

[12] P. Baptiste, C. Le Pape, and W. Nuijten. *Constraint-Based Scheduling : Applying Constraint Programming ti Scheduling Problems.* Kluwer Academic Publishers, USA., 2001.

[13] R. Barták. Constraint programming : In pursuit of the holy grail. http ://kti.mff.cuni.cz/∼bartak/constraints/, 1998.

[14] I. Baybars. Optimal assignment of broadcasting frequencies. *European Journal of Operations Research*, 9 :257–263, 1982.

[15] P. Brücker and S. Knust. A linear programming and constraint propagation-based lower bound for the rcpsp. *European Journal of Operational Research*, 127(2) :355–362, 2000.

[16] A.V. Cabot. An enumeration algorithm for knapsack problems. *Operations Research*, 18 :306–311, 1970.

[17] Y. Caseau and F. Laburthe. Solving small tsps with constraints. In *Proceedings of the 14th International Conference on Logic Programming.*, pages 316–330. MIT Press, 1997.

[18] D. J. Castelino, S. Hurley, and N. M. Stephens. A tabu search algorithm for frequency assignment. *Annals of Operations Research*, 63 :301–319, 1996.

[19] S.C. Brailsford C.N. Potts and B.M. Smith. Constraint satisfaction problems : Algorithms and applications. *European Journal Of Operational Research*, 119(3) :557–581, 1999.

[20] D. Costa. On the use of some known methods for T-colourings of graphs. *Annals of Operations Research*, 41 :343–358, 1993.

[21] Y. Crama and J.B. Mazzola. On the strength of relaxations of multidimensional knapsack problems. *INFOR*, 32 :219–225, 1994.

[22] G.B. Dantzig. *Linear Programming and Extensions.* Princeton University Press., 1963.

[23] K. Darby-Dowman, J. Little, G. Mitra, and M. Zaffalon. Constraint logic programming and integer programming approaches in solving an assignment scheduling problem. *Constraints*, 1(3) :245–264, 1997.

[24] S. Demassey, C. Artigues, and P. Michelon. A hybrid constraint propagation-cutting plane procedure for the rcpsp. In *Fourth International Workshop on Integration of AI and OR techniques in Constraint Programming for Combinatorial Optimisation Problems*, pages 321–331. Le Croisic, France, 2002.

[25] S. Demassey, C. Artigues, and Michelon P. Comparing lower bounds for the rcpsp under a hybrid constraint-linear programming approach. In *CoSolv'01 : Cooperative Solvers in Constraint Programming*, pages 109–123. Paphos, Cyprus, 2001.

[26] A. Drexl. A simulated annealing approach to the multiconstraint zero-one knapsack problem. *Computing*, 40 :1–8, 1988.

[27] N. W. Dunkin, J. E. Bater, P. G. Jeavons, and D. A. Cohen. Towards high order constraint representations for the frequency assignment problem. Technical Report CSD-TR-98-05, Royal Holloway, University of London, Egham, Surrey, UK, 1998.

[28] P.C. Chu et J.E. Beasley. A genetic algorithm for the multidimensional knapsack problem. *Journal of Heuristics*, 4 :63–86, 1998.

[29] H. Everett. Generalized lagrange multiplier method for solving problems of optimum allocation of resources. *Operations Research*, 14(11) :399–417, 1963.

[30] F. Fages. *Programmation logique par contraintes*. Ellipses, 1996.

[31] F. Foccaci. *Solving Combinatorial Optimization Problems in Constraint Programming*. PhD thesis, Università Degli Studi di Ferrara, Italia, 2000.

[32] A. Fréville. The multidimensional 0-1 knapsack problem : An overview. *European Journal of Operational Research*, 155(1) :1–21, 2004.

[33] A. Fréville, L. Lorena, and G. Plateau. Efficient surrogate algorithms for the 0-1 multiknapsack lagrangean and surrogate duals. Technical report, Université de Valenciennes, France, 1991.

[34] A. Fréville and G. Plateau. Heuristics and reduction methods for multiple constraints 0-1 linear programming problems. *European Journal of Operational Research*, 24 :206–215, 1986.

[35] A. Fréville and G. Plateau. An efficient preprocessing procedure for the multidimensional 0-1 knapsack problem. *Discrete Applied Mathematics*, 1990.

[36] A. Fréville and G. Plateau. Hard 0-1 multiknapsack test problems for size reduction methods. *Investigacion Operativa*, 1 :251–270, 1990.

[37] A. Fréville and G. Plateau. An exact surrogate dual search for the 0-1 bidimensional knapsack problem. *European Journal of Operational Research*, 1991.

[38] A. Fréville and G. Plateau. Sac à dos multidimensionnel en variables 0-1 : encadrement de la somme des variables à l'optimum. *RAIRO*, 1991.

[39] P. Galinier and J. Hao. Solving the progressive party problem by local search, 1998.

[40] A. Gamst. Some lower bounds for a class of frequency assignment problems. *IEEE Transactions on Vehicular Technology*, 35 :8–14, 1986.

[41] B. Gavish and H. Pirkul. Efficient algorithms for solving multiconstraint zeo-one knapsack problems to optimality. *Mathematical Programming*, 31 :78–105, 1985.

[42] P.C. Gilmore and R.E. Gomory. The theory and computation of knapsack functions. *Operations Research*, 14 :1045–1075, 1966.

[43] A. I. Giortzis and L. F. Turner. Application of mathematical programming to the fixed channel assignment problem in mobile radio networks. *IEE Proceedings - Communications*, 144(4) :257–264, 1997.

[44] F. Glover. Tabu search : Part i. *ORSA journal on Computing*, 1 :190–209, 1989.

[45] F. Glover. Tabu search : Part ii. *ORSA journal on Computing*, 2 :4–32, 1990.

[46] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, 1997.

[47] D.E. Goldberg. *Genetics Algorithms in Search, optimization and Machine Learning*. Addison-Wesley, 1989.

[48] S. Hanafi and A. Freville. An efficient tabu search approach for the 0-1 muldimensional knapsack problem. *European Journal of Operational Research*, 106 :659–675, 1998.

[49] J.-K. Hao, R. Dorne, and P. Galinier. Tabu search for frequency assignment in moblie radio networks. *Journal of Heuristics*, 4 :47–62, 1998.

[50] J.-K. Hao and L. Perrier. Tabu search for the frequency assignment problem in cellular radio networks. Technical Report LGI2P, EMA-EERIE, Parc Scientifique Georges Besse, Nimes, France, 1999.

[51] P. Van Hentenryck, Y. Deville, and C.-M. Teng. A generic arc consistency algorihm and its specializarions. *Artificial Intelligence*, 57 :291–321, 1992.

[52] F.S. Hillier. Efficient heuristics procedures for integer linear programming with an interior. *Operations Research*, 17 :600–637, 1969.

[53] W.J. Van Hoeve. Towards the integration of constraint logic programming and mathematical programming. Master's thesis, University of Twente, Netherlands, 2000.

[54] J. Hooker. *Logic-Based Methods for Optimization : Combining Optimization and Constraint Satisfaction*. Wiley Inter Science, USA., 2000.

[55] J.N. Hooker, H-J. Kim, and Ottoson G. A declarative modeling framework that integrates solution methods. *Annals of Operations Research*, 104(1) :141–161, 2001.

[56] J.N. Hooker and M.A. Osorio. Mixed logical linear programming. *DAMATH : Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science*, 96, 1999.

[57] J.N. Hooker, G. Ottoson, E. Thorsteinsson, and H-J. Kim. On integrating constraint propagation and linear programming for combinatorial optimization. In *In Proceedings of the sixteenth National Conference on Artificial Intelligence (AAAI-99*, volume 6, pages 136–141. Orlando, USA, 1999.

[58] J.N. Hooker, G. Ottosson, E. Thorsteinsson, and H-J. Kim. A scheme for unifying optimization and constraint programming. *Knowledge Engineering Review. Special issue on AI/OR*, 15(1) :11–30, 1999.

[59] S. Hurley and D. H. Smith. Fixed spectrum frequency assignment unsing natural algorithms. In *Proceedings of the first conference on genetic algorithms in engineering systems*, pages 373–378, Sheffield, 1995.

[60] ILOG. *ILOG CPLEX 7.1. User's Guide and References*, April 2001.

[61] ILOG. *ILOG HYBRID 1.1 : Cooperating Optimizers. User's Guide and References*, Febrary 2001.

[62] ILOG. *ILOG SOLVER 5.1. User's Guide and References*, April 2001.

[63] F. J. Jaimes-Romero, D. Munoz-Rodriguez, and S. Tekinay. Channel assignment in cellular systems using genetic algorithms. In *Proceedings of the 46th IEEE Vehicular Technology Conference*, pages 741–745, Atlanta, USA, 1996.

[64] J. Janssen and T. Wentzell. Lower bounds from tile covers for the channel assignment problem. Technical Report G-2000-09, GERAD, HEC, Montreal, Canada, 2000.

[65] B. Jaumard, O. Marcotte, C. Meyer, and T. Vovor. Erratum to "comparison of column generation models for channel assignment in cellular networks". *Discrete Applied Mathematics*, 118(3) :299–322, 2002.

[66] E. Kalvelagen. On solving the progressive party problem as a mip. Technical report, GAMS Development Corporation, Washington DC, 2001.

[67] S. Kirkpatrick, Gelatt Jr., and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220 :671–680, 1983.

[68] G.A. Kochenberger, B.A. McCarl, and F.P. Wymann. A heuristics for general integer programming. *Decision Sciences*, 5 :36–44, 1974.

[69] R. Kolisch, A. Sprecher, and A. Drexl. Characterization and generation of a general class of rcpsp. *Management Science*, 41 :1693–1703, 1995.

[70] G. Kondrak. A theoretical evaluation of selected backtracking algorithms. Master's thesis, University of Alberta, 1994.

[71] A. M. C. A. Koster, C. P. M. Van Hoesel, and A. W. J. Kolen. Solving frequency assignment problems via tree-decomposition. RM 99/011, UM, Maastricht, The Netherlands, 1999.

[72] M.C.A. Koster, P.M. Van Hoesel, and W.J. Kolen. The partial constraint satisfaction problem : Facets and lifting theorems. *Operations Research Letters.*, 23(3-5) :89–97, 1998.

[73] W. K. Lai and G. G. Coghill. Channel assignment through evolutionary optimization. *IEEE Transactions on Vehicular Technology*, 45 :91–95, 1996.

[74] T.L. Lau and E.P.K. Tsang. Guided genetic algorithm and its application to radio link frequency assignment problems. *Constraints*, 6(4) :373–398, 2001.

[75] J.S. Lee and M. Guignard. An approximative algorithm for multidimensional zero-one knapsack problems- a parametric approach. *Management Science*, 34 :402–410, 1988.

[76] R. Leese and S. Hurley. *Methods and Algorithms for Radio Channel Assignment*. Oxford University Press., 2002.

[77] R. Loulou and E. Michaelides. New greedy-like heuristics for the multidimensional 0-1 knapsack problem. *Operations Research*, 27 :1101–1114, 1979.

[78] M.J Magazine and O. Oguz. A heuristics algorithm for the multidimensional zero-one knapsack problem. *European Journal of Operational Research*, 16 :319–326, 1984.

[79] C. Mannino and A. Sassano. An enumerative algorithm for the frequency assignment problem. *Discrete Applied Mathematics*, 129(1) :155–169, 2003. Previously published as technical report 1096, Università di Roma La Sapienza, 1998.

[80] K. Marriot and P.J. Stuckey. *Programming with Constraints : An Introduction*. MIT Press, 1999.

[81] S. Martello, D. Pisinger, and P. Toth. New trends in exact algorithms for the 0-1 knapsack problem. *European Journal of Operational Research*, 123 :325–332, 2000.

[82] S. Martello and P. Toth. *Knapsack Problems : Algorithms and Computer Implementations*. Wiley, Chichester, UK., 1990.

[83] R. Montemanni, D.H. Smith, and S.M. Allen. An improved algorithm to determine lower bounds for the fixed spectrum frequency assignment problem. *European Journal of Operational Research*. À paraître.

[84] R. Montemanni, D.H. Smith, and S.M. Allen. Lower bounds for the fixed spectrum frequency assignment. *Annals of Operations Research.*, 107(1) :237–250, 2001.

[85] R. Murphey, P. Pardalos, and M. Resende. Frequency assignment problems, 1999.

[86] G.L Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley and Sons, 1988.

[87] C. Oliva, C. Artigues, and P. Michelon. Integrating linear programming and constraint programming techniques for solving mixed-integer programming problems. In *INFORMS : International Meeting, Miami, USA*, 2001.

[88] C. Oliva, C. Artigues, and P. Michelon. A hybrid approach for solving the frequency assignment problem. In *EURO/INFORMS : Joint International Meeting, Istanbul, Turkey*, 2003.

[89] C. Oliva, P. Michelon, and C. Artigues. Constraint and linear programming : Using reduced costs for solving the zero/one multiple knapsack problem. In *International Conference on Constraint Programming, CP 01, Proceedings of the workshop on*

Cooperative Solvers in Constraint Programming(CoSolv 01), Paphos, Cyprus, pages 87–98, 2001.

[90] C. Oliva, P. Michelon, and C. Artigues. Coopération de la programmation par contraintes et de la programmation linéaire pour résoudre le problème du sac à dos multidimensionnel. In *4ème congrès de la société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF 2002), Paris, France*, pages 234–235, 2002.

[91] C. Oliva, P. Michelon, C. Artigues, and M. Didi-Biha. Une approche de programmation linéaire pour le problème d'allocation de fréquences avec sommation de perturbateurs. In *5ème congrès de la société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF 2003), Avignon, France*, pages 120–121, 2003.

[92] OR-Library. `http://mscmga.ms.ic.ac.uk/jeb/orlib/mknapinfo.html`.

[93] M.A. Osorio, F. Glover, and P. Hammer. Cutting and surrogate constraint analysis for improved multidimensional knapsack solutions. Technical Report Report HCES-08-00, Hearing Center for Enterprise Science, 2000.

[94] M.A. Osorio, F. Glover, and P. Hammer. Cutting and surrogate constraint analysis for improved multidimensional knapsack solutions. *Annals of Operations Research*, 117(1) :71–93, 2002.

[95] G. Ottosson. *Integration of Constraint Programming and Integer Programming for Combinatorial Optimization*. PhD thesis, Computer Science Departement,Uppsala University, 2000.

[96] G. Ottosson and E. Thorsteinsson. Linear relaxations and reduced-cost based propagation of continuos variable subscripts. *Annals of Operations Research*, 115(1) :15–29, 2002.

[97] G. Ottosson, E. Thorsteinsson, and Hooker J.N. Mixed global constraints and inference in hybrid CLP–IP solvers. In Susanne Heipcke and Mark Wallace, editors, *Proceedings of the Fifth International Conference on Principles and Practice of Constraint Programming (CP-99)'s Post-Conference Workshop on Large Scale Combinatorial Optimisation and Constraints (LSCO&C)*, volume 4 of *Electronic Notes in Discrete Mathematics, www.elsevier.nl/locate/endm*. Elsevier Science, October 1999.

[98] M. Palpant, C. Artigues, and P. Michelon. A heuristic for solving the frequency assignment problem. In *Proceedings of the XI CLAIO*, Concepcion, Chile, 2002.

[99] M. Palpant, C. Artigues, and P. Michelon. A large neighborhood search method for solving the frequency assignment problem. Technical Report LIA report 385, Laboratoire d'Informatique d'Avignon, University of Avignon, 2003.

[100] M. Palpant, C. Oliva, C. Artigues, P. Michelon, M. Didi Biha, and T. Defaix. Affectation de fréquences avec sommation des perturbateurs : Modèles et heuristiques. In *4ème Conférence francophone de modélisation et simulation, MOSIM'03, Toulouse, France*, pages 299–304, 2003.

[101] G. Pesant, M. Gendreau, J.-Y. Potvin, and Rousseau J.-M. An exact constraint logic programming algorithm for the travelin salesman problem with time windows. *Transportation Science*, 32(1) :12–29, 1998.

[102] G. Pesant, M. Gendreau, J.-Y. Potvin, and Rousseau J.-M. On the flexibility of constraint programming models : From single to multiple time windows for the traveling salesman problem. *European Journal of Operational Reseach.*, 117(2) :253–263, 1999.

[103] G. Pesant, M. Gendreau, J-Y. Potvin, and J-M. Rousseau. An exact constraint logic programming algorithm for the traveling salesman problem with time windows. *Transportation Science*, 32(1) :12–29, 1998.

[104] K. Philip. The CHIP system. COSYTEC white paper, 1997.

[105] H. Pirkul. A heuristics solution procedure for the multiconstraint zero-one knapsack problem. *Naval Research Logistics*, 34 :161–172, 1987.

[106] J.-C. Régin. A filtering algorithm for constraints of difference in csps. In *Proceedings of AAAI-94*, pages 362–367, 1994.

[107] M. Rodosek, M.G. Wallace, and M.T. Hajian. A new approach to integrating mixed integer programming and constraint logic programming. *Annals of Operations Research*, 86(1) :63–87, 1999.

[108] O. Sarzeaud. Allocation de fréquences par échantillonnage de gibbs, recuit simulé et apprentissage par renforcement. In *5ème congrès de la société Française de Recherche Opérationnelle et d'Aide à la Décision, ROADEF 2003*, Avignon, France, 2003.

[109] M.W.P. Savelsbergh. Preprocessing and probing for mixed integer programming problems. *ORSA Journal on Computing*, 6(4) :445–454, 1994.

[110] S. Senju and Y. Toyoda. An approach to linear programming with 0-1 variables. *Management Science*, 15 :196–207, 1968.

[111] P. Shaw. Using constraint programming and local search methods to solve vehicle routing. In *Principles and Practice of Constraint Programming*, volume 1520 of *Lecture Notes in Computer Science*, pages 417–431. Pisa, Italy, 1998.

[112] W. Shih. A branch and bound method for the multiconstraint zero-one knapsack problem. *Journal of Operational Research Society*, 30 :369–378, 1979.

[113] B. Smith, S. Brailsford, P. Hubbard, and H.P. Williams. The progressive party problem : Integer linear programming and constraint programming compared. In *CP95 :*

Proceedings 1st International Conference on Principles and Practice of Constraint Programming, Marseilles, 1995.

[114] D.H. Smith, S.M. Allen, and S. Hurley. Characteristics of good meta-heuristic algorithms for the frequency assignment problem. *Annals of Operations Research*, 107(1) :285–301, 2001.

[115] S.R. Tiourine, C.A.J Hurkens, and J.K. Lenstra. Local search algorithms for the radio link frequency assignment problem. *Telecommunication Systems*, 13(2) :293–314, 2000.

[116] Toyoda. A simplified algorithm for obtaining approximate solutions to zero-one programming problems. *Management Science*, 21 :1417–1427, 1975.

[117] M. Trick. A dynamic programming approach for consistency and propagations for knapsack constraints. *Annals of Operations Research*, 118(1) :73–84, 2003.

[118] E. Tsang and C. Voudouris. Solving the radio link frequency assignment problem using guided local search. In *NATO Symposium on Radio Length Frequency Assignment*, Aalborg, Denmark, 1998. http ://cswww.essex.ac.uk/CSP/papers.html.

[119] C. Valenzuela, S. Hurley, and D. H. Smith. A permutation based genetic algorithm for minimum span frequency assignment. *Lecture Notes in Computer Science*, 1498 :907–916, 1998.

[120] M. Vasquez and J.K. Hao. A hybrid approach for the 0-1 multidimensional knapsack problem. In *Proceedings of IJCAI-01*, Seatle, Washigton, August 2001.

[121] G. Verfaillie, M. Lemaître, and T. Schiex. Russion doll search for solving constraint optimization problems. In *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI-96).*, pages 181–187, Portland, OR, USA, 1996.

[122] J. Vlasak and M. Vasquez. Résolution du problème d'attribution de fréquences avec sommation de perturbateurs. In *5ème congrès de la société Française de Recherche Opérationnelle et d'Aide à la Décision, ROADEF 2003*, Avignon, France, 2003.

[123] A. Volgenant and J.A. Zoon. An improved heuristic for the multidimensional 0-1 knapsack problems. *Journal of the Operational Research Society*, 41 :963–970, 1990.

[124] J. P. Walser. Feasible cellular frequency assignment using constraint programming abstractions. In *Proceedings of the Workshop on Constraint Programming Applications (CP96)*, Cambridge, Massachusetts, USA, 1996.

[125] H.M Weingartner and D.N. Ness. Methods for the solution of the multidimensional 0/1 knapsack problem. *Operations Research*, 15 :83–103, 1967.

[126] L.A. Wosley. *Integer Programming*. Series in Discrete Mathematics and Optimization. Wiley-Interscience, 1998.

[127] J. Würtz and T. Müller. Constructive disjunction revisited. In *20th German Anual Conference on Artificial Intelligence*, volume 1137 of *Lecture Notes in Artificial Intelligence*, pages 377–386. Desdren , Germany, Springer-Verlag, 1996.

[128] S.H. Zanackis. Heuristics 0-1 linear programming : An experimental comparison of three methods. *Management Science*, 24 :91–104, 1977.

[129] J. A. Zoellner and C. L. Beall. A breakthrough in spectrum conserving frequency assignment technology. *IEEE Transactions on Electromagnetic Compatiblity*, 19 :313–319, 1977.