Habilitation à Diriger des Recherches

de Université de Paris-Nord

Spécialité : Informatique

Présentée par

Sourour ELLOUMI

Contributions à l'Optimisation Combinatoire

Soutenue le 26 juin 2002 devant le Jury :

Président	Pr.	Gérard Plateau
Rapporteurs	Pr.	Pierre Hansen
	Pr.	Nelson Maculan
	Pr.	Philippe Michelon
Examinateurs	Pr.	Alain Billionnet
	Pr.	Marie-Christine Costa
	Pr.	Martine Labbé
	Pr.	Christian Lavault
	Pr.	Michel Minoux

Remerciements

Je tiens à remercier Gérard Plateau, Professeur à l'Université de Paris-Nord, d'avoir bien voulu parrainer ma démarche en vue de l'obtention d'une Habilitation à Diriger des Recherches. Son accueil était très chaleureux et la rédaction de ce mémoire a bénéficié de ses nombreux conseils. Je le remercie d'avoir également bien voulu présider le Jury.

Pierre Hansen, Professeur à l'Ecole des Hautes Etudes Commerciales de Montréal, Nelson Maculan, Professeur à l'Université Fédérale Rio de Janeiro, et Philippe Michelon, Professeur à l'Université d'Avignon, ont accepté d'être mes rapporteurs. Je suis fière de l'intérêt qu'il témoignent ainsi à mon travail et je les remercie chaleureusement pour leur contribution.

Michel Minoux, Professeur à l'Université de Paris 6, et Christian Lavault, Professeur à l'Université de Paris-Nord, me font l'honneur de participer au Jury. Je les remercie vivement pour leur marque d'intérêt à mon travail.

Je tiens également à exprimer ma gratitude à Martine Labbé, Professeur à l'Université Libre de Bruxelles, pour la confiance qu'elle continue de me témoigner depuis deux ans. Son accueil très chaleureux à Bruxelles a permis à notre collaboration d'être rapidement fructueuse. Je suis particulièrement sensible à sa présence dans ce Jury.

Je remercie très chaleureusement Marie-Christine Costa, Professeur au Cnam, qui m'a régulièrement rappelé certaines échéances. Son énergie positive et son souci constant des autres nous aide tous, au sein de l'équipe Optimisation Combinatoire du CEDRIC, à dépasser nos doutes. Ses nombreux conseils m'ont beaucoup aidée lors de la préparation de ce travail.

Je suis tout particulièrement reconnaissante à Alain Billionnet, Professeur à l'Institut d'Informatique d'Entreprise, de m'avoir fait découvrir l'Optimisation Combinatoire lors de mes études à l'IIE. Plus tard, j'ai naturellement choisi de préparer un Doctorat dans cette discipline. C'est donc avec lui que j'ai appris le métier de chercheur. En plus de sa contribution comme co-auteur dans les travaux que je présente ici, il a relu les premières versions de ce document et proposé plusieurs améliorations. Je le remercie vivement pour la grande disponibilité et l'écoute qu'il m'accorde depuis maintenant une quinzaine d'années.

Je voudrais également associer ceux de mes co-auteurs qui ne font pas partie du Jury. Leur contribution est bien sûr essentielle dans le dossier de synthèse que je présente aujourd'hui. De plus, je suis heureuse d'avoir partagé avec eux les joies et les déceptions liées aux procédures de publication. Il s'agit de Jean Cardinal, Chargé de Cours à l'ULB, Alain Faye, Maître de Conférences à l'IIE, Leila Grouz-Djerbi, ATER à l'Université de Paris 10, Yves Pochet, Professeur à l'Université Catholique de Louvain, et Eric Soutif, Maître de Conférences à l'Université de Paris 1.

Je remercie tous ceux grâce auxquels j'ai toujours travaillé dans une ambiance amicale. Mes collègues de l'équipe Optimisation Combinatoire, ceux du 55 rue de Turbigo et, actuellement, ceux du Service de Mathématique de la Gestion. Ils sont trop nombreux pour que je les cite tous ici mais je tiens à leur dire que les repas, cafés et discussions partagés sont pour moi une source d'inspiration et de soutien au quotidien.

Enfin, puisque d'une certaine façon ce mémoire revient sur une tranche de ma vie, je le dédie à mes plus belles réalisations de ces dix dernières années, *Ilyès (9 ans) et Lina (6 ans)*, et à mon co-réalisateur, *Mohamed*.

Table des matières

Table des figures

Liste des tableaux

Introduction Générale

Résoudre un problème d'Optimisation Combinatoire consiste à trouver un ensemble particulier parmi une famille d'ensembles admissibles. L'ensemble trouvé est optimal dans le sens où il maximise ou minimise une certaine fonction objectif, définie pour chacun des ensembles admissibles. De tels problèmes se posent dans la vie courante, le plus souvent quand des choix doivent être effectués: achat, investissement, planification, gestion, etc. La multiplication des choix possibles et des contraintes à respecter devient rapidement insurmontable pour le cerveau humain. La Théorie de la Complexité nous dit que cette difficulté devient même insurmontable dans le pire cas pour un ordinateur, pour un très grand nombre de problèmes d'Optimisation Combinatoire, qualifiés de NP-difficiles. On pense qu'il n'existe pas d'algorithmes efficaces (polynomiaux) pour résoudre les problèmes de cette classe. Il faut alors faire appel à des outils mathématiques et algorithmiques pour essayer de résoudre, de façon exacte ou approchée, des instances de taille de plus en plus importante, pour un problème donné.

La programmation mathématique en variables 0-1 constitue un outil de choix pour traiter des problèmes d'Optimisation Combinatoire puisque l'appartenance ou non d'un élément à un ensemble peut être représentée par une variable qui prend deux valeurs possibles, 0 ou 1. Presque tous les problèmes bien connus de cette discipline sont ainsi modélisés : le problème du voyageur du commerce, les problèmes de localisation, de partitionnement, d'affectation, etc. Face à un nouveau problème, on va essayer de le modéliser par un programme, linéaire ou non. En général, on peut trouver plusieurs modèles pour un même problème, dont certains sont meilleurs que d'autres. Un intérêt immédiat de cette phase de modélisation est qu'elle permet d'utiliser les outils et méthodes de la programmation mathématique : relaxation, dualité, décomposition, réduction, conditions d'optimalité, etc. L'autre intérêt considérable vient des développements logiciels récents concernant certaines catégories de programmes mathématiques, en particulier les programmes linéaires et les programmes linéaires en nombres entiers. Mais ces développements sont insuffisants pour résoudre tous les problèmes. Il faut encore appliquer différentes méthodes et formulations, essayer de les comparer, de comprendre pourquoi certaines sont meilleures que d'autres et savoir comment utiliser au mieux les logiciels.

Le sujet de ma thèse de doctorat (commencée en 1987 et soutenue en 1991) était la programmation quadratique en variables 0-1, avec, comme problème d'application, le problème de placement de tâches dans les systèmes distribués. Ce problème consiste à déterminer un placement d'un ensemble de tâches sur un réseau de processeurs, de façon à minimiser le total des coûts d'exécution et de communication entre les tâches. Les principaux résultats de ce travail de thèse sont les suivants:

- Dans le cas où les coûts de communication sont uniformes (indépendants des processeurs) et où il existe des contraintes de capacité, nous avons analysé le comportement du dual lagrangien obtenu par relaxation de toutes les contraintes. Nous avons d'abord montré que la borne inférieure de la valeur optimale ainsi obtenue est égale à celle que l'on obtient par une méthode classique de linéarisation puis relaxation continue du problème de départ. De plus, nous avons montré un résultat négatif qui dit que, dans le cas général, le saut de dualité ne peut pas être borné. Mais nous avons également prouvé que la résolution du dual lagrangien reste intéressante car elle peut être incluse dans un algorithme plus sophistiqué de calcul de borne inférieure. Ce travail a fait l'objet de la publication (Billionnet, Elloumi (1992)).

- Très peu de cas particuliers polynomiaux sont connus pour le problème de placement de tâches. Un d'entre eux est celui où les coûts de communication sont uniformes et se font à travers une structure d'arbre et où il n'y a pas de contraintes de capacité. Nous nous sommes intéressés à la recherche de k meilleures solutions (c-à-d k solutions différentes dont les valeurs sont les plus grandes dans l'ensemble des solutions admissibles). Nous avons élaboré un algorithme basé sur la programmation dynamique qui permet de calculer k meilleures solutions et dont la complexité est égale à k fois celle de la complexité de l'algorithme de calcul d'une solution optimale. Ce travail a fait l'objet de la publication (Billionnet, Elloumi (1995)).
- Nous avons également conçu et implémenté un algorithme général de calcul d'une borne inférieure pour un programme quadratique en 0-1, basé sur la notion de graphe de conflit. Nous avons mesuré la qualité de cette borne pour un problème particulier de placement de tâches sous contraintes de capacité, en la comparant à la valeur d'une solution admissible calculée par une méthode de recuit simulé.
- Dans un travail post-doctoral, nous avons effectué une synthèse des modèles et résultats concernant le problème de placement de tâches quand la communication entre les tâches se fait à travers un arbre. Ce travail a été publié dans (Billionnet, Elloumi (1994)).

Par la suite, j'ai continué à m'intéresser à la programmation quadratique en 0-1, mais j'ai également travaillé sur deux autres thèmes: la localisation discrète, en étudiant en particulier un problème de Min-Max et un problème de conception de réseau de télécommunication. L'objet de ce mémoire est de faire une synthèse de mes travaux de recherche depuis la fin de mon doctorat. Il est organisé en trois chapitres.

Le **Chapitre 1** est consacré à la programmation quadratique en 0-1. Plusieurs problèmes de la littérature, ayant diverses applications, se modélisent naturellement par des programmes mathématiques en 0-1, ayant des contraintes linéaires et une fonction objectif quadratique. Notre démarche est d'essayer de comprendre le comportement de différentes méthodes de calcul de bornes inférieures, appliquées à ces problèmes. Il s'agit principalement de méthodes de relaxation ou décomposition lagrangienne et de méthodes de linéarisation. Dans ce chapitre, nous présentons notre contribution dans l'ordre des résultats concernant un programme quadratique général vers ceux concernant des sous-ensembles particuliers d'un tel programme.

- Nous généralisons la méthode de décomposition de Chardaire et Sutter au cas d'un programme quadratique quelconque et caractérisons les choix qui conduisent à de meilleures performances de cette méthode. Nous démontrons aussi sa supériorité par rapport à des méthodes de linéarisation. Enfin, nous faisons une étude expérimentale comparée de ces méthodes en prenant comme application le problème du sac-à-dos quadratique qui a fait l'objet de plusieurs travaux récents dans la littérature. Les résultats de ces travaux ont été publiés dans (Elloumi, Faye, Soutif (2000)).

- Nous nous sommes également intéressés au problème de semi-affectation quadratique. Nous avons proposé une méthode de réduction et avons montré que cette méthode est optimale dans le sens où si une autre méthode vise à réécrire la fonction objectif sous forme d'une constante c plus une nouvelle fonction quadratique à coefficients tous positifs, la valeur de la constante c ne peut pas dépasser celle de notre méthode. Nous avons appliqué cette réduction comme étape de pré-traitement dans un processus de résolution du problème de la semi-affectation quadratique, par le logiciel Cplex, et constaté une division du temps de calcul par un facteur entre 2 et 50. Ce travail a fait l'objet de la publication (Billionnet, Elloumi (2001)).
- Nous avons souvent tenté de comparer les différentes méthodes de calcul de bornes inférieures entre elles, d'un point de vue expérimental. En effet, l'application d'une méthode à un problème particulier nécessite une étape de spécialisation de la méthode et d'optimisation de son implémentation. La comparaison se fait ensuite selon les deux critères de qualité des bornes et du temps de calcul qu'elles nécessitent. Dans le rapport CEDRIC (Elloumi, Soutif (2002)), nous faisons une comparaison expérimentale de différentes méthodes de décomposition et de linéarisation pour le problème de placement de tâches avec contraintes de capacité.

Le **Chapitre 2** traite d'un problème de Min-Max en localisation. Nous considérons précisément le problème du p-Centre. Il s'agit de localiser au maximum p centres parmi M sites potentiels et d'allouer chacun des N clients au centre le plus proche, de façon que la distance maximale entre un client et le centre qui le dessert soit minimale. Une application directe de ce problème se trouve dans la localisation de casernes de pompiers de manière à minimiser le maximum des rayons de couverture des différentes casernes. Une formulation de ce problème par un programme linéaire en variables 0-1 est connue depuis une trentaine d'années, mais on pensait encore récemment qu'il faisait partie des problèmes difficiles pour lesquels on ne connaissait pas de méthode exacte permettant de résoudre des instances de plus d'une centaine de sites et de clients. Dans un travail très récent, Daskin présente une méthode exacte et l'applique aux 40 instances du problème de la p-médiane figurant dans ORlib (librairie électronique de jeux de test) et dont la taille atteint 900 centres et 900 clients. Il obtient des solutions optimales en un temps moyen de 90 secondes, avec un temps maximal de 25 minutes pour une instance.
Nous avons d'une part trouvé une nouvelle formulation du problème du p-Centre par un programme linéaire en 0-1, de taille polynomiale. Nous avons montré la supériorité de la relaxation linéaire de cette nouvelle formulation par rapport à la formulation classique. D'autre part, nous avons proposé une nouvelle méthode de calcul de borne inférieure. Cette méthode est polynomiale et la borne qu'elle fournit peut aisément être interprétée comme la valeur optimale du programme linéaire mixte obtenu par relaxation d'un sous-ensemble des variables dans la nouvelle formulation. Nos résultats expérimentaux montrent la bonne qualité de cette borne inférieure. Pour les instances de la p-médiane citées, nous avons un temps moyen de 9 secondes avec un maximum à 1 minute et nous arrivons à résoudre différentes instances tirées de TSPlib, dont la taille atteint 1817 centres et 1817 clients et pour lesquelles on n'avait jamais réussi à estimer la valeur optimale. Ce travail fait l'objet de l'article en cours de publication (Elloumi, Labbé, Pochet (2001)).

Par ailleurs, nous montrons que notre formulation et notre méthode de résolution s'étendent aisément à un problème de codage, qui se pose en Théorie de l'Information. Le problème est, connaissant N données et une certaine distance entre elles, d'attribuer une abscisse et une ordonnée à chacune des données de façon à minimiser l'erreur potentielle. Cette erreur, pour une solution donnée, se mesure comme la distance maximale entre deux données auxquelles on attribue la même abscisse ou la même ordonnée. Il s'agit de nouveau d'un problème de Min-Max auquel nous avons réussi à appliquer les mêmes idées qu'au problème du p-Centre. Un travail préliminaire sur un problème proche de ce problème fait l'objet de la soumission (Cardinal, Elloumi, Labbé (2002)).

Le **Chapitre 3** est consacré à un problème de conception d'un réseau de télécommunication, posé par France Télécom Recherche et Développement. Il s'agit de concevoir l'architecture et de dimensionner un réseau de radio-mobiles. On considère plus particulièrement la partie fixe du réseau d'accès radio-mobile, où on connaît le nombre et l'emplacement des stations de bases appelées *BTS* et on souhaite localiser des équipements appelés *BSC* qui auront chacun la charge de gérer et écouler le trafic d'un sous-ensemble de stations BTS. Dans la nouvelle architecture, la communication entre BTS et BSC se fait à travers une *boucle SDH* matérialisée par une fibre optique qui relie les BTS entre eux puis à l'équipement BSC qui les gère. Concevoir l'architecture d'un tel réseau revient à:

- déterminer l'emplacement et le type des équipements BSC parmi un ensemble de sites possibles, et un ensemble de types d'équipements possibles. Le type d'un équipement caractérise son coût et ses capacités techniques,
- définir les boucles SDH qui vont relier les BTS aux équipements BSC. Il faut que chaque BTS appartienne à exactement une boucle et que la capacité de chaque équipement BSC soit suffisante pour gérer la totalité des BTS qui lui sont reliées. D'autres contraintes imposent des limites sur la longueur et le nombre de BTS d'une boucle.

S'il existe plusieurs architectures possibles, on en cherche une de coût minimal. Le coût d'une architecture inclut le coût d'achat et d'utilisation des équipements BSC et le coût des boucles composé d'un coût fixe par boucle et du coût d'installation, supposé proportionnel à la longueur totale de fibre optique utilisée.

Le problème posé, que nous avons appelé *SDHNDP*, fait partie d'une classe de problèmes connue dans la littérature de l'Optimisation Combinatoire sous le nom de *problèmes de localisation et de routage*. Ces problèmes combinent les deux autres classes de problèmes que sont la *localisation discrète* et les *problèmes de tournées de véhicules*. Le SDHNDP contient un sous-problème supplémentaire qui est le dimensionnement de la capacité de chaque équipement à installer. Chacun de ces sous-problèmes est difficile et le problème SDHNDP consiste à les considérer tous à la fois. La première phase de ce travail a consisté à trouver une modélisation de ce problème sous forme d'un programme linéaire en variables mixtes 0-1. Ensuite, nous avons développé une heuristique de calcul d'une solution réalisable d'une part et d'autre part essayé d'améliorer la qualité de la meilleure borne inférieure connue par une série de pré-traitements. Dans l'état actuel de nos expérimentations, l'écart entre la borne inférieure et la meilleure solution est en moyenne de 15% pour les instances fournies par France Télécom. Ce travail a fait l'objet de la thèse de Leila Grouz-Djerbi et d'un rapport CEDRIC (Billionnet, Elloumi, Grouz-Djerbi (2002)).

Liste des publications jointes au dossier d'Habilitation

- Cette liste récapitule les publications citées dans l'Introduction Générale. Le texte intégral de ces publications fait également l'objet du Tome II de ce mémoire.
- Les références ci-dessus sont reprises dans la section Bibliographie, d'où l'indication [x]

Billionnet, Elloumi (1992) [8] : "Placement de tâches dans un système distribué et dualité lagrangienne". Revue d'Automatique, d'Informatique et de Recherche Opérationnelle (R.A.I.R.O.), série verte, vol. 26, num. 1, p. 83 à 97.

Billionnet, Elloumi (1994) [9] : "Placement des tâches d'un programme à structure arborescente sur un réseau de processeurs : synthèse de résultats récents". Information Systems and Operational Research (INFOR), vol. 32, num. 2, p.65-86.

Billionnet, Elloumi (1995) [10] : "An algorithm for finding the k-best allocations of a tree-structured program". Journal of Parallel and Distributed Computing (JPDC), vol 26, num. 2, p.225-232.

Elloumi, Faye, Soutif (2000) [26]: "Decomposition and Linearization for 0-1 Quadratic Programming". Annals of Operations Research, Vol. 99, p.79-93.

Billionnet, Elloumi (2001) [11] : "Best Reduction of the Quadratic Semi-Assignment Problem". Discrete Applied Mathematics, vol. 109, p.197-213.

Elloumi, Labbé, Pochet (2001) [27] : "A new formulation and exact solution method for the p-center problem". CORE Discussion Paper 2001/53. Accepté en Mai 2002 pour une publication à la revue INFORMS Journal on Computing.

Cardinal, Elloumi, Labbé (2002) [17]: "Local optimization of index assignments for multiple description coding". XIth European Signal Processing Conference (EUSIPCO-2002), Toulouse, Septembre 2002.

Elloumi, Soutif (2002) [29]: "Comparaison expérimentale de différentes bornes inférieures pour un problème de placement de tâches". Rapport CEDRIC 323.

Billionnet, Elloumi, Grouz-Djerbi (2002) [12]: "Designing radio-mobile access networks based on SDH rings". Rapport CEDRIC 325.

Chapitre 1

Programmation quadratique en 0-1

1.1 Introduction

Ici, nous nous intéressons au programme quadratique en 0-1 suivant:

$$(QP) \quad \min \quad f(x) \ : x \in X \subset \{0,1\}^n \tag{1.1}$$

où $f : \{0,1\}^n \to \mathbb{R}$ est une *fonction pseudo-Booléenne quadratique* (f.p.B.q.) qui peut s'écrire sous la forme $f(x) = \sum\limits_{i=1}^{n} c_{ii}x_i + \sum\limits_{i=1}^{n}\sum\limits_{j=i+1}^{n} c_{ij}x_ix_j$ et X est défini comme l'ensemble des solutions en 0-1 d'un système d'inégalités linéaires $X = \{x \in \{0,1\}^n \text{ t.q. } Ax \leq b\}$.

Le problème (QP) apparaît dans plusieurs applications dans le secteur économique. Balas et Mazolla ([2] et [3]) donnent une liste d'applications des programmes non linéaires en 0-1. Il a également de nombreux cas particuliers bien connus dans le domaine de l'Optimisation Combinatoire. L'optimisation d'une f.p.B.q. sans contraintes, qui représente le cas où $X = \{0,1\}^n$, fait l'objet d'une littérature abondante, dont un travail de synthèse récent par Boros et Hammer [15]. D'autres cas particuliers du problème avec contraintes sont l'affectation quadratique (cf., par exemple [55]) et le sac-à-dos quadratique (cf., par exemple [14]) . Ces cas particuliers sont NP-durs, donc (QP) l'est aussi. Un travail de synthèse sur la programmation non-linéaire en variables 0-1 se trouve dans [41], qui inclut également le cas des contraintes non linéaires.

Dans une approche de résolution exacte, les méthodes les plus fréquemment utilisées pour essayer de résoudre les problèmes NP-durs sont les méthodes d'énumération implicite. Ces méthodes ne sont efficaces que si on dispose d'un bon algorithme d'évaluation, c'est à dire de calcul d'une borne inférieure (supérieure dans le cas de la maximisation) de la valeur optimale.

Plusieurs méthodes de calcul de borne inférieure existent pour le problème (QP). L'approche la plus souvent utilisée consiste à linéariser le problème, c'est à dire à le reformuler par un programme linéaire en 0-1, équivalent. Ainsi, la résolution de la relaxation linéaire de ce dernier problème fournit une borne pour le premier. Dantzig [22], cité par Fortet [31], a proposé un des premiers schémas de linéarisation qui consiste à remplacer chaque produit de deux variables par une nouvelle variable et une famille de nouvelles inégalités. Nous appelons ce schéma de linéarisation *la linéarisation classique*. Glover [34] a par la suite proposé un schéma qui conduit à un programme linéaire de plus petite taille puisqu'il consiste à remplacer une sous-fonction de la fonction objectif par une variable et une famille de contraintes. Nous appelons ce schéma de linéarisation *la linéarisation compacte*. Adams et Sherali [1] ont plus tard proposé une amélioration de la linéarisation classique. Cette amélioration consiste à ajouter des contraintes redondantes en 0-1, au programme quadratique (QP), avant de lui appliquer la linéarisation classique. Nous appelons ce schéma de linéarisation *la linéarisation produit*. Ces trois méthodes de linéarisation ont en commun le fait que le programme linéaire équivalent qu'elles construisent a un nombre polynomial de variables et de contraintes. Ces méthodes n'ont pas fait l'objet d'un traitement égal dans la littérature. Plusieurs travaux ont utilisé la linéarisation classique et la linéarisation produit mais très peu ont appliqué la linéarisation compacte. D'autres méthodes de linéarisation ont été proposées dans la littérature. Elles conduisent à des programmes linéaires ayant un nombre exponentiel de variables ou de contraintes, par exemple par Granot et Hammer [35] ou Balas et Mazzola [2]. Deux linéarisations différentes peuvent être comparées d'une part selon la qualité de la borne qu'elles induisent et d'autre part selon le temps de calcul qu'elles nécessitent et qui doit être aussi petit que possible. Un autre intérêt des méthodes de linéarisation est qu'on peut appliquer les outils de la programmation linéaire en nombres entiers au programme équivalent obtenu pour essayer de résoudre le problème de départ.

La relaxation lagrangienne est une approche différente de calcul de bornes inférieures. Guignard [38] affirme qu'elle constitue une méthode pratique pour la résolution des problèmes linéaires en nombres entiers. La résolution du problème dual lagrangien fournit d'une part une borne de la valeur optimale du problème primal. D'autre part, une solution du dual lagrangien peut souvent constituer un point de départ d'une recherche heuristique, permettant ainsi d'encadrer la valeur optimale. Dans le contexte de la programmation linéaire en nombres entiers, la relaxation lagrangienne a naturellement été comparée à la relaxation continue et Geoffrion [33] a montré que si le polyèdre associé au problème dual lagrangien est entier (c'est à dire que tous ses points extrêmes sont entiers) alors la valeur optimale du problème dual lagrangien est égale à celle du programme linéaire continu. Sinon, la relaxation lagrangienne de tout ou partie des contraintes fournit des bornes au moins aussi bonnes que la relaxation continue. Dans le contexte de la programmation non linéaire en nombres entiers, cette comparaison ne peut plus se faire directement, essentiellement parce que la relaxation continue ne nous ramène plus immédiatement dans le domaine bien connu de la programmation linéaire. La relaxation lagrangienne peut par contre être comparée à d'autres méthodes de relaxation, par exemple à une méthode de linéarisation suivie d'une relaxation continue.
Les méthodes de décomposition représentent un prolongement des techniques lagrangiennes. Guignard et Kim [39] ont introduit la décomposition lagrangienne pour les programmes linéaires en nombres entiers et ont montré qu'elle permettait de calculer de meilleures bornes que la relaxation lagrangienne. Michelon et Maculan [51] ont par la suite étendu la décomposition lagrangienne pour les programmes non linéaires avec contraintes linéaires. Le but de la décomposition lagrangienne dans ce cas est de séparer la fonction objectif non linéaire des contraintes linéaires. Pour ce faire, une copie de chaque variable est ajoutée. Des contraintes de copie (imposant l'égalité des variables aux copies) sont également ajoutées au programme initial, puis dualisées par le mécanisme de la dualité lagrangienne. Chardaire et Sutter [19] ont introduit une nouvelle méthode de décomposition pour les programmes quadratiques en 0-1, sans contraintes. Cette méthode peut être vue comme une généralisation de la décomposition lagrangienne, où plusieurs copies de chacune des variables sont ajoutées. L'ensemble de variables est partitionné en p sous-ensembles et la fonction objectif est décomposée en p sous-fonctions. Le but est que la fonction duale obtenue par la dualisation des contraintes de copie soit la somme de p sous-problèmes

indépendants, "faciles" à résoudre. Cette nouvelle méthode de décomposition a été généralisée et appliquée par Faye [30] à un problème de placement de tâches dans les systèmes distribués et par Billionnet et al. [14] au problème du sac-à-dos quadratique. La généralisation de cette méthode au problème (QP) est ensuite faite par Elloumi et al. [26].

Bien que nous ne l'utilisions pas dans le cadre de ce chapitre, nous pouvons citer un autre type d'approche de calcul d'une borne inférieure pour (QP), qui consiste à re-formuler (QP) ou une relaxation de (QP) par un programme semidéfini (SDP). Un (SDP) peut être vu comme un programme linéaire où l'espace des variables n'est plus R_+^n mais S_n^+, le cône des matrices semi-définies positives. Récemment, la programmation semidéfinie a été appliquée à plusieurs problèmes d'Optimisation Combinatoire. A ce jour, cette approche semble aboutir à des bornes dont on peut souvent prouver qu'elles sont en théorie plus fines que celles des autres approches mais qui sont plus coûteuses en temps de calcul. Le lecteur peut se reporter à [43] pour une récente synthèse à ce sujet.

Le but de ce chapitre est de présenter la contribution de l'auteur dans l'application et la comparaison de ces différentes techniques, pour le problème (QP) ou pour certains de ses cas particuliers. La Section 1.2 présente des notations et définitions utilisées dans la suite du chapitre. La Section 1.3 résume la publication [26] dans laquelle nous généralisons la méthode de décomposition de Chardaire et Sutter au problème (QP), nous la comparons à la méthode de linéarisation produit et enfin nous présentons les résultats d'une application expérimentale de ces deux méthodes au problème du sac-à-dos quadratique. Dans la Section 1.4 nous nous intéressons, comme dans [11] au problème de semi-affectation quadratique, pour lequel nous montrons que la linéarisation produit fournit une réduction optimale. De plus, nous discutons l'intérêt théorique et expérimental de ce résultat. La Section 1.5 présente une synthèse de résultats de comparaison théorique et expérimentales de méthodes de calcul de bornes inférieures pour les problèmes de placement de tâches. Enfin, nous consacrons la dernière section à une énumération des conclusions et des perspectives de prolongements de ces travaux.

1.2 Notations et définitions

Rappelons que quand on travaille sur un vecteur $x = (x_1,\ldots,x_n)$ de variables binaires x_i, les valeurs 0 et 1 de ces variables jouent un rôle symétrique et il est souvent naturel d'utiliser à la fois les variables x_i et leur *complément* $\bar{x}_i$. Les variables x_i et $\bar{x}_i$ sont appelées des *littéraux* et on note $\bar{x} = (\bar{x}_1,\ldots,\bar{x}_n)$.

Il est bien connu que n'importe quelle fonction pseudo-Booléenne de n variables possède une représentation unique sous forme d'un polynôme de x, appelé *représentation polynomiale* de f. Une f.p.B.q. est représentée par un polynôme de degré 2.

Définition 1.1 Réécriture

Soients f une f.p.B.q., c une constante et ϕ une posiforme, c'est-à-dire un polynôme de x et $\bar{x}$ sans terme de degré 0 et avec des coefficients tous positifs ou nuls. On dit que le couple (c,ϕ) est une réécriture de f ssi:

$$\forall x \in \{0,1\}^n \quad f(x) = c + \phi(x,\bar{x}) \tag{1.2}$$

■

On sait qu'une telle réécriture existe toujours et qu'en général, plusieurs réécritures sont possibles pour une même f.p.B.q. f. Si on s'intéresse à la minimisation de f, une réécriture (c,ϕ) fournit à la fois une borne inférieure de la valeur optimale, la constante c, et un problème équivalent, la minimisation de la posiforme ϕ. La *Hauteur* d'une f.p.B.q. f, notée $H(f)$, est la plus grande constante c telle qu'il existe une posiforme *quadratique* ϕ vérifiant (1.2). Cette notion de Hauteur d'une f.p.B.q. est aussi connue dans la littérature sous le nom de *valeur du roof dual*. Le calcul de $H(f)$ est un problème polynomial, qui peut se faire par différentes méthodes. Le lecteur peut se reporter à [40].

La notion de réécriture a été généralisée par Billionnet et Faye [13] au cas où on s'intéresse à l'optimisation d'une f.p.B.q. sur un sous-ensembe $X \subset \{0,1\}^n$.

Définition 1.2 Réécriture sur X

Soient f une f.p.B.q. et $X \subset \{0,1\}^n$, c une constante et ϕ une posiforme. On dit que le couple (c,ϕ) est une réécriture de f sur X ssi:

$$\forall x \in X \quad f(x) = c + \phi(x,\bar{x}) \tag{1.3}$$

■

On note $H_X(f)$ la plus grande constante c_X telle qu'il existe une posiforme quadratique ϕ_X telle que (c_X,ϕ_X) est une réécriture de f sur X. Il est clair que $H(f) = H_{\{0,1\}^n}(f)$ et que si $Y \subset X$ alors $H_Y(f) \geq H_X(f)$.

Billionnet et Faye [13] ont démontré le Lemme suivant, qui donne une façon de caractériser la Hauteur sur un ensemble donné X.

Lemme 1.1 *[13] Soient f une f.p.B.q. et X un sous-ensemble quelconque de $\{0,1\}^n$,*

$$H_X(f) = max\{H(f+\gamma) : \gamma \in \mathcal{F}_0(n,X)\} \tag{1.4}$$

où $\mathcal{F}_0(n,X)$ est l'ensemble des f.p.B.q. qui sont nulles pour tout $x \in X$. ■

Définition 1.3 Réduction sur X

Soients f une f.p.B.q., X un sous-ensemble de $\{0,1\}^n$, c' une constante et f' une f.p.B.q.. La fonction f' est écrite sous sa forme polynomiale, n'a pas de terme de degré 0 et a des coefficients tous positifs ou nuls (c'est à dire que f' est une posiforme qui ne contient pas de variables sous forme comlémentée $\bar{x}_i$). On dit que le couple (c',f') est une réduction de f sur X ssi:

$$\forall x \in X \quad f(x) = c' + f'(x) \tag{1.5}$$

■

Cette fois, f' est un polynôme de x, de degré 2, sans terme de degré 0 et avec des coefficients tous positifs ou nuls. Notons qu'une telle réduction n'existe pas toujours pour toute f.p.B.q et pour tout ensemble X. Cette notion n'a d'ailleurs de sens que dans le cas où X est strictement inclus dans $\{0,1\}^n$, à cause de l'unicité de la représentation polynomiale d'une fonction pseudo-Booléenne. Une conséquence directe de ces définitions est que toute réduction (c',f') de f est telle que $c' \leq H_X(f)$. Une réduction (c',f') est *optimale* s'il n'existe pas une autre réduction $(c",f")$ avec $c" > c'$.

Ici, nous rappelons la linéarisation classique, initialement introduite par Dantzig [22].

Définition 1.4 Linéarisation classique

Nous appelons linéarisation classique d'un programme quadratique en 0-1 la technique qui consiste à remplacer chaque produit de deux littéraux ab par une variable continue unique c et à ajouter les contraintes $c \leq a$, $c \leq b$, $c \geq a + b - 1$ et $c \geq 0$. ■

La linéarisation classique d'un programme quadratique en 0-1 aboutit à un programme linéaire en variables mixtes, équivalent (c'est à dire qu'à partir d'une solution admissible de l'un, on peut déduire une solution admissible de l'autre, de même valeur pour la fonction objectif).

Cette technique de linéarisation a été améliorée par Adams et Sherali [1] par ce que nous appelons ici la *linéarisation produit.*

Définition 1.5 Linéarisation produit

Nous appelons linéarisation produit d'un programme quadratique en 0-1 la technique qui consiste à:

- *ajouter au programme quadratique les contraintes obtenues en multipliant chaque contrainte par chaque littéral puis,*
- *opérer une linéarisation classique.*

■

Dans ce chapitre, nous utiliserons aussi une autre technique de linéarisation, initialement introduite par Glover [34]. Elle a la particularité, par rapport aux deux méthodes ci-dessus, d'utiliser un nombre réduit de variables et contraintes supplémentaires pour aboutir également à un programme linéaire en variables mixtes, équivalent.

Définition 1.6 Linéarisation compacte

Nous appelons linéarisation compacte d'un programme quadratique en 0-1 la technique qui consiste à remplacer le polynome $x_i f_i(x)$, où x_i est une variable binaire et $f_i(x)$ une fonction linéaire en x, par une variable unique h_i, en prenant soin d'ajouter les contraintes :

$$\underline{\lambda}_i x_i \leq h_i \leq \bar{\lambda}_i x_i$$
$$f_i(x) - \bar{\lambda}_i(1 - x_i) \leq h_i \leq f_i(x) - \underline{\lambda}_i(1 - x_i)$$

où $\underline{\lambda}_i$ (resp. $\bar{\lambda}_i$) est une borne inférieure (resp. supérieure) de $f_i(x)$. ■

1.3 Décomposition et Linéarisation produit d'un programme quadratique en 0-1

Dans cette section, nous présentons un travail publié dans [26]. Nous nous intéressons au problème général (QP) défini dans la Section 1.1 et nous généralisons la méthode de décomposition de Chardaire et Sutter [19] et précisons certaines propriétés de cette méthode. Ensuite, nous montrone qu'elle donne en théorie, et dans le cas général, des bornes plus fines que la linéarisation produit. Enfin, nous présentons un travail expérimental sur le problème du sac-à-dos quadratique, pour illustrer cette comparaison.

1.3.1 Linéarisation produit

Appliquée au programme (QP), la linéarisation produit (cf. Définition 1.5) donne un programme linéaire en nombres entiers, dont la relaxation continue est le programme linéaire suivant:

$$(LPP) \ : \ \min \quad g(x,w) = \sum_{i=1}^{n} \left(c_{ii}x_i + \sum_{j=1}^{i-1} c_{ji}w_{ji} + \sum_{j=i+1}^{n} c_{ij}w_{ij} \right) \tag{1.6}$$

sous les contraintes :

$$\sum_{j=1}^{i-1} A_j w_{ji} + \sum_{j=i+1}^{n} A_j w_{ij} \le (b - A_i)x_i \qquad i = 1,\ldots,n \tag{1.7}$$

$$\sum_{j=1}^{i-1} A_j(x_j - w_{ji}) + \sum_{j=i+1}^{n} A_j(x_j - w_{ij}) \le b(1 - x_i) \qquad i = 1,\ldots,n \tag{1.8}$$

$$w_{ij} \ge 0,\ w_{ij} \le x_i,\ w_{ij} \le x_j,\ w_{ij} \ge x_i + x_j - 1 \qquad 1 \le i < j \le n \tag{1.9}$$

Le vecteur A_j est la colonne j de la matrice A. Les contraintes (1.9) sont les contraintes de linéarisation. Elles impliquent $w_{ij} = x_i x_j$ si x_i et x_j sont dans $\{0,1\}$. Les contraintes (1.7) (resp. (1.8)) sont obtenues en multipliant $\sum_{j=1}^{n} A_j x_j \le b$ par x_i (resp. $(1 - x_i)$). Si m est le nombre de lignes de la matrice A, (LPP) possède $(\frac{3}{2}n(n-1) + 2mn)$ contraintes et $(\frac{1}{2}n(n+1))$ variables.

1.3.2 Principe et propriétés de la méthode de décomposition

Soit $2 \le p \le n$ et soit une partition $I_1,\ldots,I_p$ de l'ensemble des indices des variables $N = \{1,\ldots,n\}$. La méthode de décomposition consiste d'abord à reformuler le programme (QP) de la façon suivante:

$$(DQP)\ :\ \min\quad f'(x,y)=\sum_{k=1}^{p}\left(\sum_{i\in I_k}c_{ii}x_i+\sum_{i\in I_k}\sum_{j\in I_k\ j>i}c_{ij}x_ix_j+\sum_{i\in I_k}\sum_{j\in N-I_k\ j>i}c_{ij}x_iy_j^k\right) \tag{1.10}$$

sous les contraintes :

$$x_j=y_j^k \qquad k=1,\ldots,p;\ j\in N-I_k \tag{1.11}$$

$$x_iy_j^k=x_jy_i^{k'} \qquad k=1,\ldots,p;\ k'=1,\ldots,p,k'\neq k;\ i\in I_k;\ j\in I_{k'} \tag{1.12}$$

$$(x_{I_k},y^k_{N-I_k})\in X \qquad k=1,\ldots,p \tag{1.13}$$

$$\tag{1.14}$$

Pour obtenir cette reformulation, on ajoute d'abord $(p-1)$ copies de chaque variable. L'indice j d'une variable x_j appartient nécessairement à un sous-ensemble $I_{k'}$. On crée alors une copie y_j^k pour chaque $k\neq k'$, que nous appelons *copie d'ordre k de x_j*. Les contraintes (1.11) sont les contraintes habituelles de copie et les contraintes (1.12) sont des contraintes quadratiques, redondantes en 0-1. Dans cette formulation, les différentes copies des variables sont utilisées de telle sorte que le produit de deux variables fasse intervenir soit deux variables d'un même sous-ensemble I_k, soit une variable d'un sous-ensemble I_k et une copie d'ordre k d'une variable n'appartenant pas à I_k.

La deuxième phase de cette méthode consiste à dualiser les contraintes (1.11) et les contraintes (1.12) en les munissant respectivement des multiplicateurs de Lagrange
λ_j^k, $k=1,\ldots,p;\ j\in N-I_k$ et μ_{ij}, $k=1,\ldots,p;\ k'=1,\ldots,p,k'\neq k;\ i\in I_k;\ j\in I_{k'}$.
On posant, pour $k=1,\ldots,p$,

$$\begin{aligned}L_k(x_{I_k},y^k_{N-I_k},\lambda,\mu)=\ &\sum_{i\in I_k}\left(c_{ii}+\sum_{k'\neq k}\lambda_i^{k'}\right)x_i-\sum_{j\in N-I_k}\lambda_j^ky_j^k+\sum_{i\in I_k}\sum_{j\in I_k,j>i}c_{ij}x_ix_j\\&+\sum_{i\in I_k}\sum_{j\in N-I_k\ j>i}(c_{ij}+\mu_{ij}-\mu_{ji})\,x_iy_j^k\end{aligned}$$

on constate aisément que la fonction duale $\omega(\lambda,\mu)$ se décompose en $\sum_{k=1}^{p}v(P_k(\lambda,\mu))$ où $P_k(\lambda,\mu)$ est le problème suivant:

$$(P_k(\lambda,\mu))\ :\ \min\quad L_k(x_{I_k},y^k_{N-I_k},\lambda,\mu) \tag{1.15}$$

sous les contraintes :

$$(x_{I_k},y^k_{N-I_k})\in X \tag{1.16}$$

$$\tag{1.17}$$

et le problème dual lagrangien (D) s'écrit:

$$(D)\ :\ \max_{\lambda,\mu}\quad\sum_{k=1}^{p}v(P_k(\lambda,\mu)) \tag{1.18}$$

Pour chaque k, le problème $(P_k(\lambda,\mu))$ est de même nature que le problème primal (QP) puisqu'il consiste à minimiser, sur l'ensemble X, une f.p.B.q.. Mais, pour une valeur fixée $\tilde{x}_{I_k}$ des variables x_{I_k}, L_k est linéaire en $y^k_{N-I_k}$ et donc le problème $(P_k(\lambda,\mu)(\tilde{x}_{I_k}))$ ainsi obtenu est un programme linéaire en nombres entiers. Ainsi, par énumération sur les variables x_{I_k}, on peut soit résoudre les problèmes $(P_k(\lambda,\mu)(\tilde{x}_{I_k}))$ si cela peut se faire rapidement, soit calculer une borne inférieure de ces problèmes, par exemple en résolvant leur relaxation linéaire.

On a ainsi un moyen pratique de calculer une borne inférieure pour le problème (QP) puisque si on peut calculer $v(P_k(\lambda,\mu))$ (ou une approximation), on peut aussi calculer la valeur de la fonction duale $\omega(\lambda,\mu)$ pour toute valeur des multiplicateurs de Lagrange. Une méthode classique de sous-gradient suffit ensuite à calculer une approximation da la valeur optimale du dual (D), soit une borne inférieure de la valeur optimale de (QP).

Bien sûr, la complexité du calcul de cette borne inférieure est étroitement lié à la complexité de la résolution des sous-problèmes $(P_k(\lambda,\mu)(\tilde{x}_{I_k}))$. Elle dépend également beaucoup du nombre de sous-problèmes à résoudre à chaque pas de la méthode du sous-gradient, qui lui-même dépend de la partition $I_1,\ldots,I_p$ opérée au départ sur l'ensemble N des indices des variables. La proposition suivante permet de caractériser les partitions qui réduisent le nombre de sous-problèmes à résoudre.

Proposition 1.1 *Soit*

$$n(p) = \min_{\substack{I_1,\ldots,I_p \\ \text{est une partition de } N}} \left(\sum_{k=1}^{p} 2^{|I^k|} \right)$$

$n(p)$ représente, pour toutes les partitions possibles de N en p sous-ensembles, le nombre minimum de problèmes $(P_k(\lambda,\mu)(\tilde{x}_{I_k}))$ à résoudre, pour calculer $\omega(\lambda,\mu)$ (à chaque étape de la méthode de sous-gradient). Alors,

(i) $n(p)$ est obtenu pour une partition équilibrée, c'est-à-dire telle que $||I_k|-|I_{k'}|| \leq 1 \ \forall (k,k') \in \{1,\ldots,p\}^2$.

(ii) $n(p)$ est une fonction décroissante en p. Elle atteint son maximum pour $p = 1$ et son minimum pour $p = n$.

■

La qualité de la borne inférieure calculée par la méthode de décomposition dépend, elle aussi, de la partition. Intuitivement, on peut penser que, plus le nombre de sous-ensembles p dans la partition est grand, plus on fait de copies de variables et donc on détériore la qualité de la relaxation. Mais le nombre de sous-ensembles de la partition ne suffit pas pour comparer des partitions entre elles et n'est probablement pas le seul critère qui intervient dans la qualité de la borne. La proposition suivante permet de comparer les qualités des bornes obtenues par deux partitions telles qu'un sous-ensemble de l'une est l'union de deux sous-ensembles de l'autre.

Proposition 1.2 *Soit p tel que $2 \leq p \leq n$, $I_1,\ldots,I_p$ une partition de N en p sous-ensembles, et $I'_1,\ldots,I'_{p-1}$ une partition de N en $p-1$ sous-ensembles tels que $I'_1 = I_1$, $\ldots$, $I'_{p-2} = I_{p-2}$ et $I'_{p-1} = I_{p-1} \cup I_p$. Avec la partition $I_1,\ldots,I_p$ on obtient le problème dual (D) et la borne inférieure $v(D)$ et avec la partition $I'_1,\ldots,I'_{p-1}$ on obtient le problème dual (D') et la borne inférieure $v(D')$. Alors la partition $I'_1,\ldots,I'_{p-1}$ permet de calculer une meilleure borne inférieure: $v(D') \geq v(D)$.*

■

1.3.3 Comparaison théorique de la méthode de décomposition et de la linéarisation produit

Les propositions 1.1 et 1.2 nous conduisent à nous intéresser à une partition particulière, celle où on partitionne l'ensemble des indices des variables N en n singletons. Soit M1 la méthode de décomposition basée sur cette partition où $p = n$. D'après la proposition 1.1, la méthode M1 est celle qui a la plus faible complexité parmi toutes les autres. Elle occasionne la résolution de $2n$ sous-problèmes linéaires à chaque étape du sous-gradient. La méthode M1 est aussi celle qui conduit à la plus faible borne inférieure, puisque toute autre partition peut-être vue comme une suite d'agrégations de singletons, et donc fournit une borne au moins aussi bonne que celle de M1, compte tenu de la proposition 1.2.

Nous montrons dans le Théorème suivant que la borne inférieure calculée par la méthode M1 est au moins aussi bonne que celle calculée par la relaxation de la linéarisation produit $v(LPP)$.

Théorème 1.1 *Soit (D_1) le problème dual Lagrangien obtenu pour la partition en singletons alors:*

$$v(LPP) \leq v(D_1).$$

■

Une conséquence immédiale du Théorème 1.1 et de la Proposition 1.2 est qu'une méthode de décomposition utilisant une partition quelconque calcule une borne au moins aussi bonne que $v(LPP)$.

1.3.4 Comparaison expérimentale pour le problème du sac-à-dos quadratique

Nous avons considéré, comme cas particulier du problème (QP), le problème du sac-à-dos quadratique (QKP). Ce problème peut être formulé par un programme quadratique en 0-1, contenant une seule contrainte $\sum_{i=1}^{n} a_i x_i \leq b$ où les coefficients a_i et b sont tous positifs. Il a de nombreuses applications et a fait l'objets de plusieurs publications récentes, dont par exemple [14]. Nous avons voulu comparer les résultats des trois méthodes suivantes:

- La linéarisation produit, qui conduit au calcul d'une borne inférieure de (QKP) par la résolution du programme linéaire $(LPP_{(QKP)})$.
- La méthode de décomposition en partant de la partition en n singletons de l'ensemble des variables, soit la méthode M1 citée ci-dessus. Le calcul de la fonction duale à chaque pas de la méthode de sous-gradient nécessite la résolution de $2n$ sous-problèmes. Dans ce cas, les sous-problèmes sont des problèmes de sac-à-dos linéaires (ayant chacun de l'ordre de n variables), dont nous calculons seulement la valeur optimale de la relaxation continue. Nous notons $v(\tilde{D_1})$ la borne inférieure ainsi obtenue.
- la méthode de décomposition en partant de la partition de l'ensemble des variables en sous-ensembles de cardinal 5, que nous désignons par méthode M5. Ici aussi, nous calculons la valeur de la relaxation continue de $\frac{32}{5}n$ (obtenu par $\sum_1^{\frac{n}{5}} 2^5$) sous-problèmes, et nous notons $v(\tilde{D_5})$ la borne inférieure ainsi obtenue.

Nous utilisons le même solveur de programmes linéaires (Cplex 4.0) et la même machine (HP 9000) pour calculer toutes les bornes. Nous considérons le même type d'instances que ceux de la littérature, avec n variant de 50 à 200 et une densité (c-à-d pourcentage des termes non nuls dans la fonction objectf) variant entre 25% et 100%. Nous constatons que:

- La borne $v(LPP_{(QKP)})$ est excellente. Son écart de l'optimum est de moins de 2,6% pour toutes les instances testées et il est de l'ordre de 1% en moyenne. Mais elle est très coûteuse en temps de calcul: on passe d'environ 2 minutes pour les instances à 50 variables à environ 5 heures pour les instances à 150 variables. La taille du programme linéaire correspondant aux instances à 200 variables dépasse les capacité du solveur.
- La borne $v(\tilde{D}_1)$ est légèrement moins bonne que $v(LPP_{(QKP)})$. Rappelons que $v(\tilde{D}_1)$ est une approximation de la borne $v(D1)$ qui, comme nous le montrons ci-dessus, est au moins aussi bonne que $v(LPP_{(QKP)})$. Néanmoins, elle se calcule très vite. Il faut environ 22 secondes pour les instances à 50 variables, 4 minutes pour les instances à 150 variables et 7 minutes pour les instances à 200 variables.
- La borne $v(\tilde{D}_5)$ est en général strictement meilleure que les deux autres. Sur les 20 instances testées, son écart moyen de la valeur optimale est de 0.56% contre 0.75% pour $v(\tilde{D}_1)$. Elle est en revanche moins rapide. Il faut 1 minute pour les instances à 50 variables et 15 minutes pour les instances à 200 variables. Comparée à $v(LPP_{(QKP)})$, $v(\tilde{D}_5)$ est à la fois de meilleure qualité et beaucoup plus rapide.
- Nous avons voulu voir ce que donne la linéarisation classique (cf. Définition 1.4) sur ces instances et constaté un écart moyen de l'optimum égal à 10%, pour un temps de calcul supérieur à celui de $v(\tilde{D}_1)$.

La Figure 1.1 illustre les résultats comportement des trois bornes pour les instances de densité égale à 50%. L'axe des ordonnées, indique l'erreur relative des bornes, c-à-d leur écart relatif de la valeur de la solution optimale, ou à défaut de la meilleure solution connue, calculée par l'heuristique de Billionnet et Calmels [6]. Nous avons omis la valeur $v(LPP_{(QKP)})$ pour $n = 150$ pour une raison d'échelle (erreur =0.5% et temps de calcul égal à 15770 secondes). Pour $n = 200$, 7 heures de temps CPU n'ont pas suffi au calcul de $v(LPP_{(QKP)})$. Notons que le comportement de ces bornes est très similaires pour les autres valeurs de la densité.

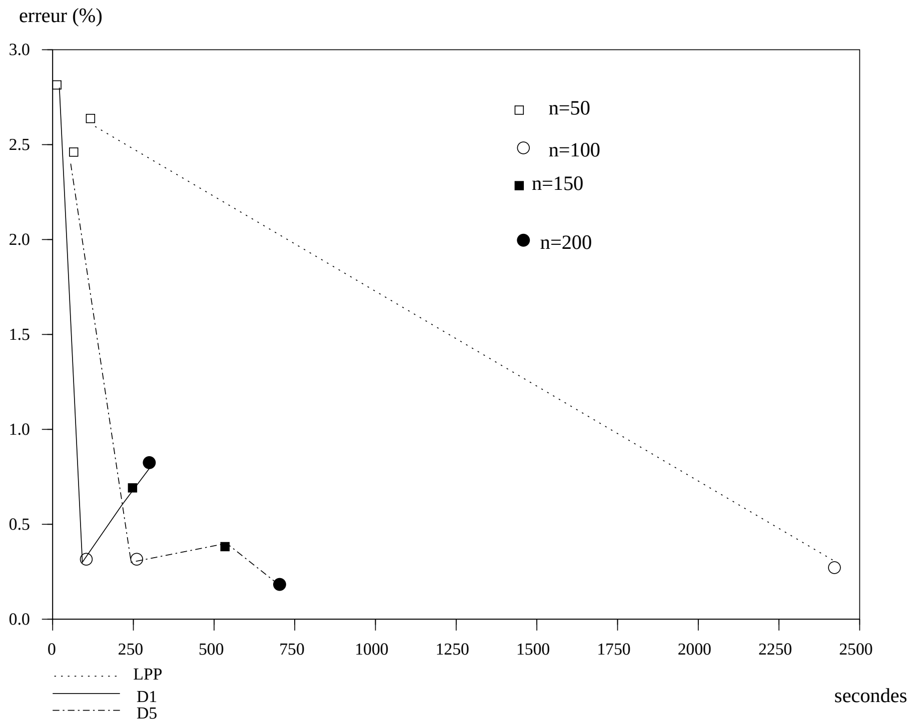


FIG. 1.1 – *Comparaison des bornes pour les instances du sac-à-dos quadratique de densité 50%*

1.4 Réduction optimale du problème de semi-affectation quadratique

Nous nous intéressons dans [11] au problème de la *semi-affectation quadratique*. Ce problème est un cas particulier de programmation quadratique en 0-1 et a de nombreuses applications pratiques. D'un point de vue théorique, comme le soulignent Pardalos et al. [55], il peut être vu comme une généralisation du problème très étudié de l'affectation quadratique. On peut aussi le considérer comme une unification de plusieurs problèmes d'optimisation combinatoire comme certains problèmes de classification, de partition pour minimiser la variance ou de coloration de graphes.

Le résultat principal de ce travail est que la linéarisation produit (cf. Définition 1.3) conduit à une réduction (cf. Définition 1.5) *optimale* du problème de la semi-affectation quadratique. Ce résultat est présenté dans la Section 1.4.1. Dans la Section 1.4.2, nous donnons un exemple et discutons l'intérêt de ce résultat. La Section 1.4.3 est consacrée à la présentation d'un travail expérimental qui illustre l'intérêt pratique de cette réduction.

1.4.1 Réduction optimale par la linéarisation produit

Précisons d'abord que nous travaillons sur le problème suivant :

$$(SAQ) \quad : \; min \;\; F(x) = \sum_{i=1}^{N}\sum_{j=1}^{M} e_{ij}x_{ij} + \sum_{i=1}^{N-1}\sum_{j=1}^{M}\sum_{k=i+1}^{N}\sum_{\ell=1}^{M} c_{ijk\ell}x_{ij}x_{k\ell} \tag{1.19}$$

sous les contraintes :

$$\sum_{j=1}^{M} x_{ij} = 1 \qquad i = 1,\ldots,N \tag{1.20}$$

$$x \in \{0,1\}^{N\times M} \tag{1.21}$$

et définissons $\mathcal{A}$ comme l'ensemble des solutions réalisables de ce problème. Les coefficients e_{ij} et $c_{ijk\ell}$ sont de signe quelconque.

Proposition 1.3 *Soit (LP) le programme linéaire suivant:*

$$(LP) \quad : \; min \;\; \mathcal{F}(x,y) = \sum_{i=1}^{N}\sum_{j=1}^{M} e_{ij}x_{ij} + \sum_{i=1}^{N-1}\sum_{j=1}^{M}\sum_{k=i+1}^{N}\sum_{\ell=1}^{M} c_{ijk\ell}y_{ijk\ell} \tag{1.22}$$

sous les contraintes :

$$\sum_{j=1}^{M} x_{ij} = 1 \qquad (1 \le i \le N) \tag{1.23}$$

$$\sum_{j=1}^{M} y_{ijk\ell} = x_{k\ell} \qquad 1 \le i,k \le N, i \ne k; 1 \le j,\ell \le M \tag{1.24}$$

$$y_{ijk\ell} = y_{k\ell ij} \qquad 1 \le i < k \le N, i \ne k; 1 \le j,\ell \le M \tag{1.25}$$

$$y_{ijk\ell} \ge 0 \qquad 1 \le i,k \le N, i \ne k; 1 \le j,\ell \le M \tag{1.26}$$

et soient δ_{ij} $(1 \le i \le N; 1 \le j \le M$ et $\delta_{ijk\ell}$ $(1 \le i,k \le N, i \ne k; 1 \le j,\ell \le M)$ les coûts réduits non négatifs associés à une solution de base réalisable et optimale $(\tilde{x}_{ij},\tilde{y}_{ijk\ell})$ de (LP) alors,

$$\forall x \in \mathcal{A} \;\; F(x) = v(LP) + \sum_{i=1}^{N}\sum_{j=1}^{M} \delta_{ij}x_{ij} + \sum_{i=1}^{N-1}\sum_{j=1}^{M}\sum_{k=i+1}^{N}\sum_{\ell=1}^{M} \delta_{ijk\ell}x_{ij}x_{k\ell}. \tag{1.27}$$

■

Revenons sur la façon dont est construit le programme (LP):

- On ajoute des contraintes en multipliant chaque contrainte $\sum_{j=1}^{M} x_{ij} = 1 \;\; (1 \le i \le N)$ par chaque variable $x_{k\ell}$. Ces contraintes sont redondantes quand les variables sont entières, elles donnent lieu aux contraintes (1.24) après linéarisation,
- On linéarise le programme quadratique obtenu, en remplaçant chaque produit de variables $x_{ij}x_{k\ell}$ par une variable $y_{ijk\ell} \ge 0$, d'où les contraintes de symétrie (1.25),
- On relâche les contraintes d'intégrité des variables x,

- On montre alors facilement que les contraintes classiques de linéarisation $y_{ijk\ell} \leq x_{ij}$, $y_{ijk\ell} \leq x_{k\ell}$ et $y_{ijk\ell} \geq x_{ij} + x_{k\ell} - 1$, ainsi que les contraintes $x_{ij} \geq 0$ sont redondantes.

On peut aussi montrer que ce programme est équivalent au programme obtenu par la Linéarisation produit (cf. Définition 1.5). A cause des contraintes (1.20), les contraintes qu'on rajouterait encore en multipliant chaque contrainte par chaque littéral $\bar{x}_{k\ell}$ seraient redondantes par rapport aux contraintes (1.24).

La Proposition 1.3 montre que la résolution du programme linéaire (LP) fournit une réduction de la fonction F sur l'ensemble $\mathcal{A}$ (cf. Définition 1.3). Nous montrons ensuite que cette réduction est optimale parce que $v(LP)$ est égale à la hauteur de F sur $\mathcal{A}$, $H_{\mathcal{A}}(F)$. Le point crucial était, comme dans le Lemme 1.1, de caractériser les f.p.B.q. nulles sur tout le domaine $\mathcal{A}$. Pour ce faire, nous avons démontré le Lemme 1.2 qui dit que toute fonction nulle sur $\mathcal{A}$ peut s'écrire comme une somme:

- d'une combinaison linéaire de fonctions linéaires nulles, immédiatement déduites des contraintes linéaires qui définissent le domaine $\mathcal{A}$,
- d'une combinaison linéaire de fonctions quadratiques obtenues par la multiplication d'une variable quelconque par une des fonctions linéaires ci-dessus et,
- d'une combinaison linéaire de termes quadratiques $x_{ij}x_{i\ell}$, évidemment tous nuls sur le domaine $\mathcal{A}$.

Lemme 1.2 *Soit Γ une f.p.B.q. de NM variables, dont la représentation polynomiale est:*

$$\Gamma(x) = q_0 + \sum_{i=1}^{N}\sum_{j=1}^{M} q_{ij}x_{ij} + \sum_{i=1}^{N-1}\sum_{j=1}^{M}\sum_{k=i+1}^{N}\sum_{\ell=1}^{M} q_{ijk\ell}x_{ij}x_{k\ell} + \sum_{i=1}^{N}\sum_{j=1}^{M-1}\sum_{\ell=j+1}^{M} \mu_{ij\ell}x_{ij}x_{i\ell} \tag{1.28}$$

alors $\Gamma(x) = 0$ *pour tout* $x \in \mathcal{A} \iff$

$\exists \lambda_i \in \mathbb{R}(i = 1, \ldots, N); \exists \delta_{ikj} \in \mathbb{R}(i = 1, \ldots, N; k = 1, \ldots, N; k \neq i; j = 1, \ldots, M)$ *tels que:*

$$\Gamma(x) = \sum_{i=1}^{N} \lambda_i g_i(x) + \sum_{i=1}^{N}\sum_{k=1;k\neq i}^{N}\sum_{j=1}^{M} \delta_{ikj}x_{kj}g_i(x) + \sum_{i=1}^{N}\sum_{j=1}^{M-1}\sum_{\ell=j+1}^{M} \mu_{ij\ell}x_{ij}x_{i\ell} \tag{1.29}$$

où $g_i(x) = (\sum_{j=1}^{M} x_{ij} - 1)$, $i = 1, \ldots, N$. ■

Ainsi, le calcul de la hauteur de la fonction F sur $\mathcal{A}$, $H_{\mathcal{A}}(F)$, se ramène d'après le Lemme 1.1 et le Lemme 1.2, à trouver les coefficients λ_i, δ_{ikj} et $\mu_{ij\ell}$ qui maximisent la hauteur de $(F + \Gamma)$ sur $\{0,1\}^{NM}$, $H(F + \Gamma)$. Nous montrons, en utilisant la dualité en programmation linéaire et les résultats de Hammer et al. [40] sur le calcul de la Hauteur d'une f.p.B.q. par linéarisation, le théorème suivant:

Théorème 1.2 $H_{\mathcal{A}}(F) = v(LP)$. ■

1.4.2 Exemple et Discussion

Considérons l'instance suivante de $(QSAP)$:

Instance

$N = M = 3$

$F(x) = -3x_{11} - 3x_{12} - 5x_{21} + 4x_{22} - 4x_{33} + 2x_{11}x_{21} + 5x_{12}x_{21} + 6x_{12}x_{33} + 5x_{21}x_{33}$

La résolution du programme linéaire (LP) correspondant à cette instance mène à la réduction suivante:

$F(x) = -7 + x_{21} + 4x_{22} + 3x_{12}x_{21} + 6x_{12}x_{33} + x_{21}x_{33} + x_{13}x_{21} + 3x_{13}x_{22} + 3x_{13}x_{23} + 4x_{22}x_{31} + 4x_{23}x_{31} + 4x_{22}x_{32} + 4x_{23}x_{32}$

On peut aisément vérifier que cette réécriture n'est pas valide pour tout vecteur x de $\{0,1\}^{3\times 3}$, par exemple pour $\hat{x} = 0$. En revanche, elle est valide pour tout x solution admissible pour le problème de semi-affectation quadratique. Soit $\tilde{x}$ tel que $\tilde{x}_{11} = \tilde{x}_{23} = \tilde{x}_{33} = 1$ et toutes les autres composantes nulles. $F(\tilde{x}) = -7$ par les deux expressions et on peut alors affirmer que $\tilde{x}$ est une solution optimale puisque la réduction nous dit aussi que -7 est une borne inférieure de la valeur optimale.
On peut également vérifier sur cette instance que la hauteur sur $\mathcal{A}$, $H_{\mathcal{A}}(F)$, peut être strictement supérieure à des bornes fournies par d'autres réductions dont par exemple celle qu'on obtient par le calcul de $H(F)$ ou encore celle qui consiste à intégrer les contraintes dans la fonction objectif en les élevant au carré et les munissant d'une pénalité suffisamment grande, puis à calculer la hauteur de la nouvelle fonction objectif.

La contribution de ce travail réside en deux points:

1- Le fait d'avoir montré que la résolution du programme (LP), obtenu par relaxation continue de la linéarisation produit du problème, fournissait une réduction du problème de semi-affectation quadratique. L'intérêt étant qu'on peut travailler ensuite sur le problème réduit et utiliser d'autres méthodes d'optimisation.

2- Le fait d'avoir montré que cette réduction était optimale dans le sens où la borne inférieure qu'elle fournit est au moins aussi bonne que n'importe quelle autre méthode de calcul de borne inférieure qui conduit à une réécriture de la fonction objectif. Cela donne un critère de comparaison des différentes méthodes de calcul de borne inférieure:

  - Pour une méthode qui conduit à une écriture $(c_{\mathcal{A}},\phi_{\mathcal{A}}(x,\bar{x}))$ de la fonction objectif par une posiforme de degré 2 (réécriture), on peut affirmer que la borne inférieure $c_{\mathcal{A}}$ qu'elle fournit est inférieure ou égale à la borne $H_{\mathcal{A}}(F)$ calculée par la réduction optimale,
  - Pour une méthode $\mathcal{M}$ qui calcule une borne inférieure strictement supérieure à $H_{\mathcal{A}}(F)$ pour au moins une instance du problème, on peut affirmer que la méthode $\mathcal{M}$ ne peut pas conduire à une réécriture de F sur le domaine $\mathcal{A}$. Si une reformulation de F par une posiforme existe avec cette borne inférieure, alors la posiforme est de degré au moins 3.

1.4.3 Intérêt pratique de la réduction optimale

Afin de mesurer l'intérêt pratique de cette méthode de réduction, nous avons utilisé un solveur de programmes linéaires en nombres entiers et comparé les deux méthodes suivantes:

Méthode1 : Linéarisation classique (cf. Définition 1.4) de (SAQ) et résolution en 0-1 du programme linéaire obtenu.

Méthode2 : Réduction puis linéarisation classique du problème réduit et résolution en 0-1 du programme linéaire obtenu.

Nous avons constitué des instances de la façon suivante: des valeurs sont choisies pour N et M puis les coefficients e_{ij} et c_{ijkl} sont tirés *aléatoirement* dans l'intervalle $[-50{,}50]$. Nous avons utilisé les logiciels AMPL [32] pour la modélisation et Cplex 6.5 [21] pour la résolution des programmes linéaires et des programmes linéaires en nombres entiers. Le solveur est interrompu au bout de 10 minutes de temps CPU si un critère d'arrêt habituel n'est pas rencontré auparavant. On constate que:

- La Méthode 2 est entre 2 et 52 fois plus rapide que la Méthode 1 sur l'ensemble des instances testées,
- En 10 minutes de temps CPU, la Méthode 2 résout des instances de plus grande taille: jusqu'à $N = 18$ et $M = 4$ contre $N = 15$ et $M = 2$ pour la Méthode 1.

1.5 Comparaison théorique et expérimentale de différentes méthodes de calcul de bornes inférieures pour le problème de placement de tâches

Nous nous intéressons ici aux problèmes de placement de tâches dans les sytèmes distribués. Dans la Section 1.5.1, nous présentons cette classe de problèmes. Dans la Section 1.5.2, nous décrivons les propriétés de la relaxation lagrangienne appliquée à un cas particulier de ce problème, puis dans la Section 1.5.3, nous appliquons et comparons différentes méthodes de calcul de bornes pour le même cas particulier.

1.5.1 Présentation des problèmes de placement de tâches

Nous considérons un système distribué formé de sites et d'un réseau de communications totalement maillé (deux sites distincts peuvent toujours échanger des informations). Chaque site est muni d'un processeur disposant de ressources propres, dont une mémoire de taille limitée. Les processeurs des différents sites sont hétérogènes et peuvent communiquer à travers le réseau. Un programme distribué est composé de tâches qui peuvent s'exécuter sur les sites (d'une manière parallèle ou séquentielle). Chaque tâche doit être affectée à un processeur unique mais un site peut exécuter plusieurs tâches, dans la limite de la mémoire dont il dispose. Certaines tâches doivent se communiquer des informations par le réseau si elles n'ont pas été affectées au même site.

A chaque affectation de l'ensemble des tâches aux processeurs correspond un coût d'exécution (somme des coûts d'exécution des tâches sur les processeurs hétérogènes auxquels elles sont affectées) et un coût de communication (somme des coûts de communication entre les couples de tâches non affectées au même processeur et devant échanger des informations). Le problème posé ici est la recherche d'une affectation des tâches aux processeurs qui tient compte des contraintes de capacité mémoire des processeurs et qui minimise la somme des coûts d'exécution et de communication.

D'une façon plus formelle, soit $P = \{p_1,p_2,\dots,p_P\}$ un ensemble de P processeurs et $T = \{t_1,t_2,\dots,t_T\}$ un ensemble de T tâches. Nous disposons des données suivantes :

- q_{tp} : le coût d'exécution de la tâche t quand elle est affectée au processeur p;
- $c_{tpt'p'}$ $(t < t')$: le coût de communication entre la tâche t, affectée au processeur p, et la tâche t' , affectée au processeur p'. Quand ce coût est indépendant des processeurs, il est dit *uniforme* et nous le noterons $c_{tt'}$;
- s_t: la quantité de mémoire requise par la tâche t;
- n_p: la quantité de mémoire dont dispose le processeur p. On supposera qu'il existe au moins un placement possible compte tenu des valeurs des s_t et des n_p. Une hypothèse minimale dans ce cas est $\sum_{t=1}^{T} s_t \leq \sum_{p=1}^{P} n_p$.

le problème se formule comme le programme non linéaire en 0-1 suivant :

$$\min \quad F(x) = \sum_{t=1}^{T}\sum_{p=1}^{P} \ell_{tp}x_{tp} + \sum_{t=1}^{T-1}\sum_{t'=t+1}^{T}\sum_{p=1}^{P}\sum_{p'=1}^{P} c_{tpt'p'}x_{tp}x_{t'p'} \tag{1.30}$$

sous les contraintes :

$$\sum_{p=1}^{P} x_{tp} = 1 \qquad t = 1,\ldots,T \tag{1.31}$$

$$\sum_{t=1}^{T} s_t x_{tp} \leq n_p \qquad p = 1,\ldots,P \tag{1.32}$$

$$x \in \{0,1\}^{T\times P} \tag{1.33}$$

La variable de décision binaire x_{tp} vaut 1 si et seulement si la tâche t_i est affectée au processeur p. Les contraintes (1.31) assurent que chaque tâche soit affectée à exactement un processeur et les contraintes (1.32) que la somme des demandes des tâches affectées à un processeur ne dépasse pas la capacité de ce processeur.

Ce problème est intéressant car plusieurs de ses cas particuliers sont des problèmes "génériques" en Optimisation Combinatoire :

- Si tous les coûts de communication sont nuls, on retrouve un problème d'*affectation linéaire généralisée*.
- Si tous les coûts d'exécution sont nuls et si les coûts de communication ne dépendent pas des processeurs, ce problème peut être vu comme un problème de *partitionnement d'un graphe* de N sommets en au plus M ensembles, de façon à respecter des contraintes de capacité des ensembles et à minimiser la somme des poids des arêtes dont les extrémités appartiennent à deux ensembles différents.
- Si on relâche les contraintes de capacité (1.32), le problème résultant peut être vu comme un problème de *semi-affectation quadratique*.
- Le cas particulier où le nombre de tâches est égal au nombre de processeurs et où les contraintes de capacité font que l'on doit placer exactement une tâche sur chaque processeur est le problème largement étudié de l'*affectation quadratique*.

Bien sûr, en dehors de quelques cas particuliers polynomiaux, ce problème est NP-dur.

1.5.2 Relaxation lagrangienne appliquée au placement de tâches avec coûts de communication uniformes

Dans la référence [8], nous considérons le problème de placement de tâches dans le cas où le coûts de communication sont uniformes. Nous nous intéressons au dual lagrangien obtenu par relaxation de toutes les contraintes. Le point de départ de ce travail est un autre travail de Billionnet, Costa et Sutter [7] qui montrent que, pour le problème de placement de tâches à coûts de communications uniformes mais sans contraintes de capacité (1.32), la borne inférieure que donne la valeur optimale du problème dual lagrangien est d'excellente qualité. Mais nos premières expérimentations concernant le cas avec contraintes de capacité ont montré une grande détérioration de la qualité de cette borne inférieure. Nous nous sommes alors attachés à essayer de mettre en évidence des propriétés du problème dual lagrangien et expliquer le comportement de sa valeur optimale.

Rappelons que $c_{tt'}$ est le coût de communication positif entre les tâches t et t' si elles sont placées sur deux processeurs différents. Le fait que ces coûts soient uniformes et les contraintes d'affectation (1.31) font que la fonction objectif peut s'écrire sous forme d'une fonction *sous-modulaire*.

En posant:

- $\ell_{Tp} = q_{Tp}$ pour $p = 1,\ldots,P$ et
- $\ell_{tp} = q_{tp} + \sum_{t'=t+1}^{T} c_{tt'}$ pour $t = 1,\ldots,T-1$ et $p = 1,\ldots,P$,

le problème se transforme en:

$$(PU) \ : \quad \min \ \ F(x) = \sum_{t=1}^{T}\sum_{p=1}^{P} \ell_{tp}x_{tp} - \sum_{t=1}^{T-1}\sum_{t'=t+1}^{T}\sum_{p=1}^{P} c_{tt'}x_{tp}x_{t'p} \tag{1.34}$$

sous les contraintes:

(1.31),(1.32),(1.33)

En associant le vecteur de multiplicateurs de Lagrange $\alpha = (\alpha_t \in R)_{(t=1,\ldots,T)}$ aux contraintes (1.31) et $\beta = (\beta_p \geq 0)_{(p=1,\ldots,P)}$ aux contraintes (1.32), la fonction duale s'écrit:

$$L(\alpha,\beta) = \underset{x\in\{0,1\}^{TP}}{Min} \left(-\sum_{t=1}^{T}\alpha_t - \sum_{p=1}^{P} n_p\beta_p + \sum_{t=1}^{T}\sum_{p=1}^{P}(\ell_{tp} + \alpha_t + s_t\beta_p)x_{tp} - \sum_{t=1}^{T-1}\sum_{t'=t+1}^{T}\sum_{p=1}^{P} c_{tt'}x_{tp}x_{t'p} \right) \tag{1.35}$$

et le problème dual est:

$$\max \quad L(\alpha,\beta) \tag{1.36}$$

sous les contraintes:

$$\alpha_t \in R \qquad t = 1,\ldots,T \tag{1.37}$$

$$\beta_p \geq 0 \qquad p = 1,\ldots,P \tag{1.38}$$

Pour une valeur donnée des multiplicateurs de Lagrange (α,β), le calcul de la fonction duale (1.35) peut se faire en un temps polynomial. En effet, les coefficients des termes quadratiques étant tous négatifs, la fonction duale est une f.p.B.q. sous-modulaire dont une des façons efficaces de calculer le minimum est de résoudre un problème de flot maximal (voir par exemple [16]). On peut donc calculer facilement, par la méthode classique de sous-gradient, une solution (α^*,β^*) quasi-optimale pour le problème dual On obtient ainsi $L(\alpha^*,\beta^*)$ qui est une borne inférieure de la valeur optimale du problème primal.

Dans la référence [8], nous montrons les résultats suivants:

- La borne $L(\alpha^*,\beta^*)$ est aussi la valeur optimale obtenue par la relaxation continue de la linéarisation classique (cf. Définition 1.4) du problème (1.34)-(1.33).

- La borne $L(\alpha^*,\beta^*)$ est comprise entre deux valeurs, L_{min} et L_{max}, qu'elle peut atteindre. Les valeurs L_{min} et L_{max} sont indépendantes des coûts de communication entre les tâches et se calculent facilement à partir des autres données du problème. Par exemple, quand les coûts d'exécution des tâches sur les processeurs sont tous nuls, $L_{min} = L_{max} = 0$ et donc $L(\alpha^*,\beta^*) = 0$. On peut alors construire des instances où le saut de dualité est plus grand que n'importe quelle valeur Δ donnée. Ceci n'exclut pas l'existence de classes d'instances où il n'y a pas de saut de dualité. Dans le cas où il n'y a pas de contraintes de capacité (i.e., chaque processeur peut exécuter l'ensemble des tâches) les auteurs de [7] observent que les sauts de dualité sont très rares. Dans ce cas, nous montrons que L_{max} dépasse la valeur optimale du problème de placement de tâches.
- Le calcul de la solution (α^*,β^*) est intéressant, même si la borne $L(\alpha^*,\beta^*)$ n'est pas assez efficace pour être intégrée à un algorithme de type *Branch and Bound*, car il peut être considéré comme une première phase de calcul d'une meilleure borne inférieure. En effet, la solution (α^*,β^*) permet d'écrire, pour toute solution admissible x:

$$F(x) \geq L(\alpha^*,\beta^*) + \phi(x,\bar{x}) \tag{1.39}$$

  où $\phi(x,\bar{x})$ est une posiforme quadratique dont les coefficients se calculent à partir de α^* et β^*.

1.5.3 Comparaison expérimentale de différentes bornes pour le problème de placement de tâches avec coûts de communication uniformes

Nous allons présenter [29] l'adaptation et les résultats expérimentaux de quatre méthodes différentes de calcul d'une borne inférieure pour la valeur optimale du problème (PU).

<u>**Méthode M1**</u>: Linéarisation classique
Cette méthode de calcul de borne inférieure consite à résoudre la relaxation continue de la linéarisation classique du problème (cf. Définition 1.4).

Plus précisément, la Méthode M1 calcule la borne inférieure $b_1 = v(LP_1)$ où (LP_1) est le programme linéaire suivant :

$$(LP_1) \;:\; \min \;\; F_1(x,z) = \sum_{t=1}^{T}\sum_{p=1}^{P} \ell_{tp}x_{tp} - \sum_{t=1}^{T-1}\sum_{t'=t+1}^{T}\sum_{p=1}^{P} c_{tt'}z_{tpt'} \tag{1.40}$$

sous les contraintes :
(1.31), (1.32)

$$z_{tpt'} \leq x_{tp} \qquad 1 \leq t < t' \leq T; p = 1,\ldots,P \tag{1.41}$$

$$z_{tpt'} \leq x_{t'p} \qquad 1 \leq t < t' \leq T; p = 1,\ldots,P \tag{1.42}$$

$$x_{tp} \geq 0 \qquad t = 1,\ldots,T; p = 1,\ldots,P \tag{1.43}$$

Dans une solution optimale de (LP_1), la valeur de $z_{tpt'}$ $(t < t')$ est égale au minimum des valeurs de x_{tp} et $x_{t'p}$, ce qui est aussi le produit $x_{tp}x_{t'p}$ si les variables x sont entières.

Nous montrons dans la sous-section 1.5.2 que la borne b_1 est égale à la valeur optimale du problème dual lagrangien obtenu en relâchant toutes les contraintes du problème (PU).

Pour le calcul pratique de b_1, il suffit d'utiliser un solveur de programmes linéaires.

Méthode M2: Linéarisation produit
Cette méthode de calcul de borne inférieure consite à résoudre la relaxation continue de la linéarisation produit (cf. Définition 1.5) du problème (PU).

On résout ici précisément le programme linéaire (LP_2) suivant :

$$(LP_2) \;:\; \min \; F_2(x,r) = \sum_{t=1}^{T}\sum_{p=1}^{P} \ell_{tp}x_{tp} - \sum_{t=1}^{T-1}\sum_{t'=t+1}^{T}\sum_{p=1}^{P} c_{tt'}r_{tpt'p} \tag{1.44}$$

sous les contraintes :
(1.31), (1.32)

$$\sum_{p=1}^{P} r_{tpt'p'} = x_{t'p'} \qquad 1 \leq t < t' \leq T; p' = 1,\ldots,P \tag{1.45}$$

$$\sum_{p=1}^{P} r_{t'p'tp} = x_{t'p'} \qquad 1 \leq t' < t \leq T; p' = 1,\ldots,P \tag{1.46}$$

$$\sum_{t=1}^{t'-1} s_t r_{tpt'p} + \sum_{t=t'+1}^{T} s_t r_{t'ptp} + \leq (n_p - s'_t)x_{t'p} \qquad t' = 1,\ldots,T; p = 1,\ldots,P \tag{1.47}$$

$$\sum_{t=1}^{t'-1} s_t r_{tpt'p'} + \sum_{t=t'+1}^{T} s_t r_{t'p'tp} + \leq n_p x_{t'p'} \qquad t' = 1,\ldots,T; p = 1,\ldots,P; p' = 1,\ldots,P, p' \neq p \tag{1.48}$$

$$x_{tp} \geq 0 \qquad t = 1,\ldots,T; p = 1,\ldots,P \tag{1.49}$$

$$r_{tpt'p'} \geq 0 \qquad 1 \leq t < t' \leq T; p = 1,\ldots,P; p' = 1,\ldots,P \tag{1.50}$$

Pour chaque tâche t, on multiplie la contrainte d'affectation correspondante $\sum\limits_{p=1}^{P} x_{tp} = 1$ par chaque variable $x_{t'p'}$ telle que $t' \neq t$. Cela donne les contraintes (1.45) pour $t < t'$ et les contraintes (1.46) pour $t' < t$. De même, pour chaque processeur p, on multiplie la contrainte de capacité correspondante $\sum\limits_{t=1}^{T} s_t x_{tp} \leq n_p$ par chaque variable $x_{t'p}$, ce qui donne les contraintes (1.47) et par chaque variable $x_{t'p'}$ avec $p' \neq p$, ce qui donne les contraintes (1.48). L'écriture de ces contraintes tient compte du fait que $x_{tp}x_{tp} = x_{tp}$ et $x_{tp}x_{t'p} = 0$ pour $t' \neq t$. On peut par ailleurs aisément montrer que les contraintes (1.31), (1.32) et (1.45)-(1.50) impliquent les contraintes de linéarisation qui, quand x_{tp} et $x_{t'p'}$ ont des valeurs dans $\{0,1\}$, obligent $r_{tpt'p'}$ à valoir $x_{tp}x_{t'p'}$. Notons de plus que les contraintes (1.31) font que les contraintes obtenues par multiplication des contraintes (1.31) et (1.32) par $(1 - x_{tp})$ sont alors redondantes.

La borne b_2 calculée par cette méthode est égale à $v(LP_2)$. On peut facilement montrer que $b_2 \geq b_1$ et voir que le calcul de b_2 est plus coûteux en temps puisque (LP_2) contient beaucoup plus de variables et contraintes que (LP_1).

Dans le cas où il n'y a pas de contraintes de capacité (1.32), Decocq [25] montre que $b_1 = b_2$.

Méthode M3: Linéarisation compacte
Nous utilisons ici la linéarisation compacte (cf. Définition 1.6), qui consiste à remplacer des sous-fonctions quadratiques de la fonction objectif, par des variables continues et de nouvelles contraintes.

Réécrivons d'abord la fonction objectif du programme (PU)

$$F(x) = \sum_{t=1}^{T}\sum_{p=1}^{P} \ell_{tp}x_{tp} - \sum_{t=1}^{T}\sum_{t'=1,\ t'\neq t}^{T}\sum_{p=1}^{P} \tfrac{1}{2}c_{tt'}x_{tp}x_{t'p}$$

$$= \sum_{t=1}^{T} \sum_{p=1}^{P} \ell_{tp} x_{tp} - \sum_{t=1}^{T} \sum_{p=1}^{P} \left(\sum_{t'=1,\ t' \neq t}^{T} \frac{1}{2} c_{tt'} x_{tp} x_{t'p} \right)$$

On veut ensuite remplacer l'expression $\left(\sum_{t'=1,\ t' \neq t}^{T} \frac{1}{2} c_{tt'} x_{tp} x_{t'p} \right)$ par une variable unique h_{tp} pour $t = 1, \ldots, T-1; p = 1, \ldots, P$. Suivant la Définition 1.6, nous devons disposer d'une borne inférieure $\underline{\alpha}_{tp}$ et d'une borne supérieure $\bar{\alpha}_{tp}$ sur la quantité $\left(\sum_{t'=1,\ t' \neq t}^{T} \frac{1}{2} c_{tt'} x_{tp} x_{t'p} \right)$. Ici, nous fixons $\underline{\alpha}_{tp}$ à 0 et choisissons dans un premier temps des valeurs arbitrairement grandes pour des constantes $\bar{\alpha}_{tp}$. On peut facilement voir que la résolution du programme linéaire $(L_{\bar{\alpha}})$ fournit une solution optimale pour (PU).

$$\min \delta(x,h) = \sum_{t=1}^{T} \sum_{p=1}^{P} \ell_{tp} x_{tp} - \sum_{t=1}^{T} \sum_{p=1}^{P} h_{tp} \tag{1.51}$$

sous les contraintes:
(1.31), (1.32)

$$h_{tp} \leq \sum_{t'=1,\ t' \neq t}^{T} \frac{1}{2} c_{tt'} x_{t'p} \qquad t = 1,...,T; p = 1,...,P \tag{1.52}$$

$$h_{tp} \leq \bar{\alpha}_{tp} x_{tp} \qquad t = 1,...,T; p = 1,...,P \tag{1.53}$$

$$h_{tp} \geq 0 \qquad t = 1,...,T; p = 1,...,P \tag{1.54}$$

$$x_{tp} \in \{0,1\} \qquad t = 1,...,T; p = 1,...,P \tag{1.55}$$

En effet, si le choix des constantes $\bar{\alpha}_{tp}$ est correct, alors pour tout (x,h) solution optimale de $(L_{\bar{\alpha}})$, $h_{tp} = x_{tp} \sum_{t'=1,\ t' \neq t}^{T} \frac{1}{2} c_{tt'} x_{t'p}$.Pour $x_{tp} = 0$, $h_{tp} = 0$ découle des contraintes (1.53) et, pour $x_{tp} = 1$, h_{tp} prend nécessairement la plus grande valeur possible, soit $\sum_{t'=1,\ t' \neq t}^{T} \frac{1}{2} c_{tt'} x_{t'p}$ d'après les contraintes (1.52).

Ainsi, pour un choix donné de $\bar{\alpha}$, la résolution de la relaxation continue de $(L_{\bar{\alpha}})$ fournit une borne inférieure pour le problème (PU) et, pour que cette borne soit la plus fine possible, on a intérêt à choisir des valeurs les plus petites possibles pour $\bar{\alpha}_{tp}$. Ici, nous calculons $\bar{\alpha}_{tp}$ comme la valeur optimale (notons qu'une borne supérieure suffirait) du programme linéaire en nombres entiers suivant:

$$\bar{\alpha}^*_{tp} = \max \sum_{t'=1,\ t'\neq t}^{T} \frac{1}{2} c_{tt'} x_{t'p} \tag{1.56}$$

sous les contraintes :

$$x_{tp} = 1 \tag{1.57}$$

$$\sum_{p'=1}^{P} x_{t'p'} = 1 \qquad t' = 1,...,T \tag{1.58}$$

$$\sum_{t'=1}^{T} s_t x_{t'p'} \leq n_{p'} \qquad p' = 1,...,P \tag{1.59}$$

$$x \in \{0,1\}^{T\times P} \tag{1.60}$$

Une fois calculées des valeurs optimales $\bar{\alpha}^*_{tp}$, on peut déduire le programme linéaire en nombres entiers $(L_{\bar{\alpha}^*})$, dont la valeur optimale est égale à celle de (PU). Notons que des bornes supérieures sur les valeurs de $\bar{\alpha}^*_{tp}$ suffiraient. Mais notre expérience de calcul montre que le calcul des $\bar{\alpha}^*_{tp}$ est très rapide et donc il n'est pas forcément rentable d'introduire une approximation à ce niveau.

La relaxation continue du problème $(L_{\bar{\alpha}^*})$ est un programme linéaire que nous appelons (LP_3). Nous appelons b_3 la valeur optimale de (LP_3). C'est la borne inférieure calculée par la méthode M3.

Méthode M4: Décomposition

Ici, nous utilisons la méthode de décomposition présentée dans la Section 1.3.2.

On partitionne l'ensemble des variables en T sous-ensembles $S_1 \ldots S_T$ tels que, pour chaque tâche t, $S_t = \{x_{tp}\}_{p=1\ldots P}$.

A chaque variable x_{tp} $(t = 1,...,T; p = 1,...,P)$ on associe $(T-1)$ copies $y^{t'}_{tp}$ $(t' = 1,...,T, t' \neq t)$. On construit le programme quadratique suivant, qui a la même valeur optimale que (P):

$$\min \sum_{t=1}^{T} \left(\sum_{p=1}^{P} \ell_{tp} x_{tp} - \sum_{t'=1,t'\neq t}^{T} \frac{1}{2} c_{tt'} \sum_{p=1}^{P} x_{tp} y^{t}_{t'p} \right) \tag{1.61}$$

sous les contraintes :

$$x_{tp} = y^{t'}_{tp} \qquad 1 \leq t \neq t' \leq T;\ p = 1,\ldots,P \tag{1.62}$$

$$x_{tp} y^{t}_{t'p'} = x_{t'p'} y^{t'}_{tp} \qquad 1 \leq t < t' \leq T,\ 1 \leq p \neq p' \leq P \tag{1.63}$$

$$\sum_{p=1}^{P} x_{tp} = 1 \qquad t = 1,\ldots,T \tag{1.64}$$

$$\sum_{p=1}^{P} y^{t}_{t'p} = 1 \qquad 1 \leq t \neq t' \leq T \tag{1.65}$$

$$s_t x_{tp} + \sum_{t'=1,t'\neq t}^{T} s_{t'} y^{t}_{t'p} \leq n_p \qquad t = 1,\ldots,T; p = 1,\ldots,P \tag{1.66}$$

$$x_{tp} \in \{0,1\} \qquad t = 1,\ldots,T; p = 1,\ldots,P \tag{1.67}$$

$$y^{t'}_{tp} \in \{0,1\} \qquad 1 \leq t \neq t' \leq T;\ p = 1,\ldots,P \tag{1.68}$$

Les contraintes (1.62) sont des contraintes habituelles de copie. Les contraintes (1.63) sont des contraintes redondantes de copie. Dans cette nouvelle écriture de la fonction objectif (1.61), elle se décompose en T sous-fonctions telles que, pour t donné, la sous-fonction correspondante utilise exlusivement les variables x_{tp} et les copies *indexées* t, $y^t_{t'p'}$, des variables $x_{t'p'}$. Cette propriété est aussi vraie pour les contraintes (1.64)- (1.66) qui se décomposent, pour chaque t, en groupes de contraintes portant sur les mêmes variables. Ainsi vient naturellement l'idée d'effectuer une relaxation lagrangienne en "dualisant" les contraintes (1.62) et (1.63). Soient:

- $\lambda_{tpt'}$, $1 \leq t \neq t' \leq T$; $p = 1,\ldots,P$ les coefficients de Lagrange (libres) associés aux contraintes (1.62) et
- $\mu_{tpt'p'}$, $1 \leq t < t' \leq T$, $1 \leq p \neq p' \leq P$ les coefficients de Lagrange (libres) associés aux contraintes (1.63).

On constate facilement que la fonction duale $\omega(\lambda,\mu)$ se décompose en :

$$\omega(\lambda,\mu) = \sum_{t=1}^{T} v\left(\mathcal{P}_t(\lambda,\mu)\right) \tag{1.69}$$

où $(\mathcal{P}_t(\lambda,\mu))$ est le programme linéaire en nombres entiers suivant:

$$\begin{aligned} \min \quad & \sum_{p=1}^{P}\left(\ell_{tp} + \sum_{t'=1,t'\neq t}^{T}\lambda_{tpt'}\right)x_{tp} - \sum_{p'=1}^{P}\sum_{t'=1,t'\neq t}^{T}\lambda_{t'p't}y^t_{t'p'} - \sum_{t'=1,t'\neq t}^{T}\tfrac{1}{2}c_{tt'}\sum_{p=1}^{P}x_{tp}y^t_{t'p} \\ & + \sum_{t'=1,t'>t}^{T}\sum_{p=1}^{P}\sum_{p'=1}^{P}\mu_{tpt'p'}x_{tp}y^t_{t'p'} - \sum_{t'=1,t'<t}^{T}\sum_{p=1}^{P}\sum_{p'=1}^{P}\mu_{t'p'tp}x_{tp}y^t_{t'p'} \end{aligned} \tag{1.70}$$

sous les contraintes : (1.64)-(1.68)

On peut maintenant remarquer que, pour une valeur fixée des variables x_{tp}, $(p = 1,\ldots,P)$, le programme $(\mathcal{P}_t(\lambda,\mu))$ devient linéaire. De plus, il est particulièrement aisé de faire l'énumération de toutes les valeurs possibles des x_{tp}, pour t fixé:

- $x_{t1} = 1$ implique $x_{tp} = 0$, $p = 1,\ldots,P, p \neq 1$ ou
- $x_{t2} = 1$ implique $x_{tp} = 0$, $p = 1,\ldots,P, p \neq 2$ ou
- ...
- $x_{tP} = 1$ implique $x_{tp} = 0$, $p = 1,\ldots,P, p \neq P$.

Ainsi, la valeur optimale de $(\mathcal{P}_t(\lambda,\mu))$ peut être calculée par:

$$v\left(\mathcal{P}_t(\lambda,\mu)\right) = \min_{p=1,\ldots,P} v\left(\mathcal{P}^p_t(\lambda,\mu)\right) \tag{1.71}$$

où $\mathcal{P}^p_t(\lambda,\mu)$ est obtenu à partir de $(\mathcal{P}_t(\lambda,\mu))$ en fixant x_{tp} à 1. C'est donc le programme linéaire en nombres entiers:

$$
\begin{aligned}
\min \quad & \ell_{tp} + \sum_{t'=1,t'\neq t}^{T} \lambda_{tpt'} - \sum_{p'=1}^{P} \sum_{t'=1,t'\neq t}^{T} \lambda_{t'p't} y^t_{t'p'} - \sum_{t'=1,t'\neq t}^{T} \tfrac{1}{2} c_{tt'} y^t_{t'p} \\
+ & \sum_{t'=1,t'>t}^{T} \sum_{p'=1}^{P} \mu_{tpt'p'} y^t_{t'p'} - \sum_{t'=1,t'<t}^{T} \sum_{p'=1}^{P} \mu_{t'p'tp} y^t_{t'p'}
\end{aligned}
\tag{1.72}
$$

sous les contraintes :

$$
\sum_{p'=1}^{P} y^t_{t'p'} = 1 \qquad t' = 1,\ldots,T,\ t' \neq t \tag{1.73}
$$

$$
\sum_{t'=1,t'\neq t}^{T} s_{t'} y^t_{t'p} \leq n_p - s_t \tag{1.74}
$$

$$
\sum_{t'=1,t'\neq t}^{T} s_{t'} y^t_{t'p'} \leq n_{p'} \qquad p' = 1,\ldots,P, p' \neq p \tag{1.75}
$$

$$
y^t_{t'p'} \in \{0,1\} \qquad t' = 1,\ldots,T,\ t' \neq t; p' = 1,\ldots,P \tag{1.76}
$$

Nous avons alors une méthode de calcul de la fonction duale pour une valeur donnée des multiplicateurs de Lagrange, utilisant la programmation linéaire en nombres entiers:

Calcul de $\omega(\lambda,\mu)$

- Pour chaque t, calculer $v\left(\mathcal{P}_t(\lambda,\mu)\right)$:
  - * Pour chaque p, résoudre $\mathcal{P}_t^p(\lambda,\mu)$,
  - * Déduire $v\left(\mathcal{P}_t(\lambda,\mu)\right)$ par (1.71)
- Déduire $\omega(\lambda,\mu)$ par (1.69)

Ensuite, on peut appliquer une méthode classique de sous-gradient pour calculer une valeur approchée du problème dual lagrangien. Cette valeur est une borne inférieure pour le problème (PU).

Le programme $(\mathcal{P}_t^p(\lambda,\mu))$ contient $(T-1)P$ variables et $(T-1)+P$ contraintes. Le calcul de $\omega(\lambda,\mu)$ nécessite la résolution de TP programmes $(\mathcal{P}_t^p(\lambda,\mu))$. Nous appelons b_4 la borne inférieure obtenue par la méthode M4 où les problèmes $(\mathcal{P}_t^p(\lambda,\mu))$ sont résolus de façon exacte.

Nous considérons aussi le cas où on résout seulement la relaxation continue des problèmes $(\mathcal{P}_t^p(\lambda,\mu))$, on obtient alors une borne inférieure de b_4 et donc une borne inférieure pour le problème primal (PU), de moins bonne qualité que b_4. Nous notons $\tilde{b}_4$ cette borne plus faible mais plus rapide à calculer.

D'après les résultats de Elloumi et al. [26], on peut déduire que $b_4 \geq b_2$.

Résultats

Nous avons généré plusieurs instances du problème de placement de tâches et effectué des tests pour mesurer la qualité des différentes bornes, et leur comportement suivant certains paramètres. Nous avons ainsi sélectionné 4 configurations de jeu d'essai et, pour chacune de ces 4 configurations, nous considérons le cas où le graphe de communication entre les tâches est complet puis le cas où la densité de ce graphe est égale à 1/2. Les tailles de nos jeux d'essai sont soit 10 tâches et 3 processeurs soit 20 tâches et 5 processeurs.

	config1	config2	config3	config4
int1	100	10	100	0
int2	100	100	10	100

TAB. 1.1 – *Différentes configurations de jeux d'essai pour le problème de placement de tâches*

Les coûts d'exécution sont tirés aléatoirement dans l'intervalle [0, int1]. Les coûts de communication dans l'intervalle [0, int2].

Les tailles des tâches (s_t, $t = 1,...,T$) sont tirées dans l'intervalle [1, 10] et les capacités des processeurs (n_p, $p = 1,...P$) dans l'intervalle $[S/P,\ 2 * S/P]$ où S est la somme des tailles de toutes les tâches.

Toutes les expériences ont été effectuées sur une machine de type ordinateur personnel, muni d'un processeur Pentium II à 400 MHz et d'une mémoire de 384 MB. Pour le résolution des programmes linéaires, nous avouns utilisé le logiciel CPLEX version 7.0. Les Tables 1.2 et 1.3, ainsi que la Figure 1.2 résument et illustrent les résultats que nous avons obtenus.

	T	P	$\frac{(opt-b_1)}{opt}$	cpu(s)	$\frac{(opt-b_2)}{opt}$	cpu(s)	$\frac{(opt-b_3)}{opt}$	cpu(s)	$\frac{(opt-\bar{b}_4)}{opt}$	cpu(s)	$\frac{(opt-b_4)}{opt}$	cpu(s)
Config 1	10	3	68%	0.13	12%	0.75	23%	0.60	15%	30	10%	60
	20	5	85%	0.68	13%	82	27%	4	19%	238	17%	648
Config 2	10	3	94%	0.13	29%	0.70	37%	0.56	33%	25	24%	54
	20	5	98%	0.60	17%	74.4	32%	4	25%	223	23%	576
Config 3	10	3	4%	0.12	1%	0.61	4%	0.54	1%	19	0%	14
	20	5	19%	0.70	1%	77	10%	3.8	1%	240	1%	419
Config 4	10	3	100%	0.13	25%	0.63	39%	0.56	32%	27	22%	49
	20	5	100%	0.62	18%	65	31%	4	24%	205	22%	624

TAB. 1.2 – *Comparaison des bornes : Cas où le graphe de communication est complet*

	T	P	$\frac{(opt-b_1)}{opt}$	cpu(s)	$\frac{(opt-b_2)}{opt}$	cpu(s)	$\frac{(opt-b_3)}{opt}$	cpu(s)	$\frac{(opt-\bar{b}_4)}{opt}$	cpu(s)	$\frac{(opt-b_4)}{opt}$	cpu(s)
Config 1	10	3	37%	0.07	21%	0.71	27%	0.54	25%	26	19%	48
	20	5	69%	0.35	29%	71	40%	4	35%	214	32%	520
Config 2	10	3	85%	0.08	48%	0.61	57%	0.53	57%	24	47%	45
	20	5	96%	0.36	51%	80	61%	3.57	65%	156	58%	432
Config 3	10	3	2%	0.06	1%	0.69	3%	0.55	1%	19	0%	10
	20	5	2%	0.36	4%	69	7%	4	1%	228	1%	420
Config 4	10	3	100%	0.06	43%	0.67	52%	0.52	55%	25	37%	55
	20	5	100%	0.34	56%	69	63%	4	66%	127	61%	374

TAB. 1.3 – *Comparaison des bornes : Cas où le graphe de communication est à moitié dense*

1.6 Conclusion

Nous pouvons dire que le fil conducteur des différents travaux présentés dans ce chapitre est la comparaison de différentes méthodes de linéarisation des programmes quadratiques en 0-1, dans le but à la fois de calculer une borne inférieure de leur valeur optimale, et/ou de les résoudre.

La Section 1.3 montre que la méthode de décomposition donne en théorie des bornes au moins aussi fines que la linéarisation produit. Cette dernière linéarisation est supérieure à plusieurs autres méthodes de linéarisation et donne déjà une très bonne borne inférieure pour le

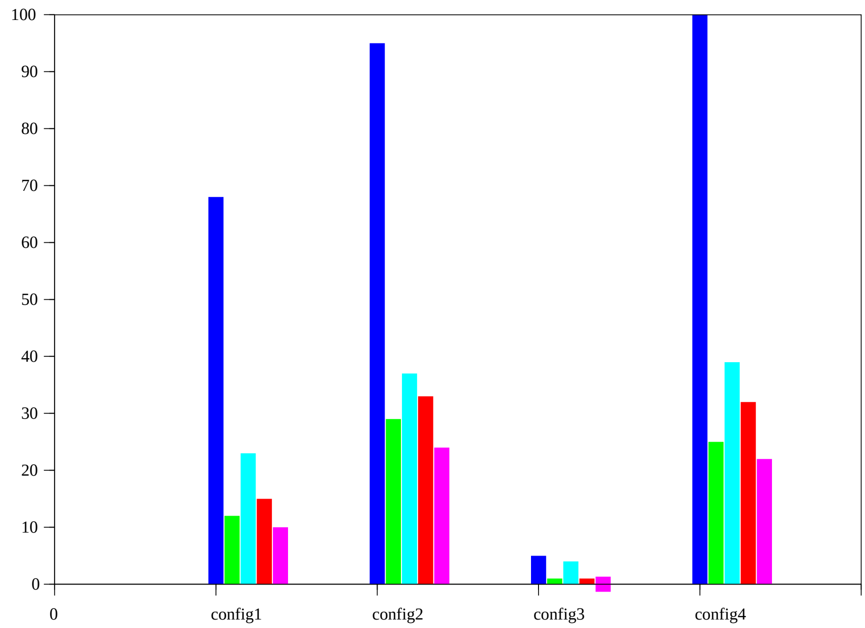


FIG. 1.2 – *Comparaison de la qualité des bornes pour le problème de placement de tâches*
Instances de 10 tâches et 3 processeurs avec un graphe de communication complet
Pour chacune des configurations, le graphique indique l'erreur relative respectivement des bornes b_1, b_2, b_3, $\tilde{b}_4$ et b_4

problème du sac-à-dos quadratique. De plus, la méthode de décomposition, ou plus précisément une approximation de cette méthode, est beaucoup plus rapide car elle nécessite la résolution de plusieurs "petits" programmes linéaires, alors que la linéarisation produit resout un programme linéaire de grande taille. La mise en œuvre pratique de cette méthode est plus délicate car, en plus de l'affinage de la méthode de sous-gradient, on doit faire des choix au niveau de la partition de l'ensemble des variables, qui peuvent être importants pour le rapport qualité de la borne sur son temps de calcul. Nous sommes cependant convaincus que c'est une méthode intéressante, qui mérite d'être envisagée pour d'autres problèmes quadratiques.

La Section 1.4 montre une propriété intéressante de la linéarisation produit, pour le problème de la semi-affectation quadratique. Nous avons réservé une part importante de cette section à prouver l'intérêt pratique du fait de savoir que la réduction obtenue par la résolution du programme linéaire (LP) est optimale parmi toutes les réductions qui conservent le caractère quadratique de la fonction objectif. On peut aisément voir que cette méthode peut être appliquée au problème de l'Affectation Quadratique (QAP). Le programme linéaire (LP_{QAP}) obtenu par relaxation continue du programme obtenu par linéarisation produit de (QAP) fournit aussi une réduction. Il est donc naturel de conjecturer que cette réduction est également optimale au vu du même critère. A ce jour, nous n'avons pas réussi à prouver cette conjecture. Des travaux de Karisch et al. [46] vont dans le sens de cette conjecture puisque, considérant le problème dual (DLP_{QAP}) de (LP_{QAP}), ils montrent que la plupart les méthodes connues de calcul de borne inférieure ou de réduction de (QAP) reviennent à calculer des solutions réalisables de (DLP_{QAP}) et donc fournissent des réductions moins bonnes que celles qui utilisent une solution optimale de (DLP_{QAP}) ou de (LP_{QAP}). Karisch et al. [46] proposent une façon plus efficace de résoudre (DLP_{QAP}), qui profite de le structure particulière de ce problème dual. Nous pensons que leurs idées peuvent être utilisées pour améliorer la résolution du programme (LP) ou de son dual, pour le problème de la semi-affectation quadratique. La taille de ce programme linéaire augmente en effet rapidement en fonction des données et il est certainement très utile d'exploiter sa structure pour espérer appliquer cette méthode à des instances de taille importante. Un autre avantage du fait d'utiliser le problème dual est qu'on peut se contenter d'une bonne solution admissible puisque toute solution du dual donne une réduction.

Dans la Section 1.5, nous comparons les résultats de la linéarisation classique, la linéarisation produit, la linéarisation compacte et la méthode de décomposition pour un problème de placement de tâches avec coûts de communication uniformes et contraintes de capacité. Ce travail met en évidence l'intérêt de la linéarisation compacte. La borne ainsi obtenue n'est certes pas toujours de bonne qualité mais elle est presque toujours meilleure que celle de la linéarisation classique et elle est surtout très rapide à calculer. Une conclusion similaire avait été dégagée des travaux de Soutif [60] pour le problème de sac-à-dos quadratique. Une voie de recherche probablement intéressante consite à essayer d'améliorer la linéarisation compacte. D'ailleurs, des travaux très récents ([18], [37]) utilisent, généralisent et améliorent cette idée de linéarisation initialement dûe à Glover [34].

Chapitre 2

Le problème du p-Centre

2.1 Introduction

Le problème du p-Centre consiste à localiser p centres parmi un ensemble de localisations possibles puis à affecter chaque client au centre qui lui est le plus proche, de façon à minimiser le rayon, c'est à dire la distance maximale entre un client et le centre qui le dessert. Ce problème a plusieurs applications dont les plus connues sont les problèmes de localisation de casernes de pompiers ou de dépôts d'ambulances.

Soit N le nombre de clients notés C_1,C_2,...,C_N; M le nombre de sites potentiels pour les centres F_1,F_2,...,F_M; et d_{ij} la distance de C_i à F_j. Le problème est de trouver un sous-ensemble S de $\{1,...,M\}$ tel que:

- $|S| \leq p$
- $rayon(S) = max_{i=1,...,N}(min_{j \in S} d_{ij})$ est minimum

Ce problème inclut le cas où un poids non négatif w_i est associé à chaque client C_i et où le rayon d'une solution S est $max_{i=1,...,N}(w_i min_{j \in S} d_{ij})$. Pour tenir compte de ces poids, il suffit de modifier les distances en posant $d'_{ij} = w_i d_{ij}$.

Ce problème est introduit dans la littérature dans les années 1970 ([59], [47]) où il a d'abord fait l'objet de quelques développements algorithmiques qui ont immédiatement mis l'accent sur son lien avec le problème de recouvrement: le rayon optimal est aussi la plus petite distance telle qu'on puisse couvrir tous les clients par p centres, à l'intérieur de cette distance. Plus tard, on s'est intéressé à des algorithmes d'approximation de ce problème, c'est à dire des algorithmes polynomiaux qui fournissent une solution admissible dont l'écart relatif par rapport à la valeur optimale est limité par une constante ou par une fonction des données du problème. Hochbaum et Shmoys [44] présentent un algorithme 2-approché dans le cas où les distances sont Euclidiennes et montrent que sous l'hypothèse $P \neq NP$, il ne peut pas exister un algorithme ϵ-approché avec $\epsilon < 2$. Plusieurs travaux très récents proposent des méthodes de résolution exacte ou approchée de ce problème. Mladenovic et al. [54] implémentent et comparent une méthode de recherche par voisinage variable et une méthode taboue et calculent des solutions approchées pour des instances ayant de l'ordre de 3000 clients et sites potentiels. Daskin [24] d'une part et Ilhan et Pinar [45] d'autre part présentent des méthodes de résolution exacte et les testent sur un sous-ensemble des instances considérées par Mladenovic et al. [54]. Ils arrivent à résoudre de façon exacte des instances de taille maximale $N = M = 900$.

Dans la publication [27]:

1. Nous présentons une nouvelle formulation du problème du p-Centre, qui tient mieux compte de sa relation avec le problème de recouvrement. Nous montrons la supériorité de cette nouvelle formulation par rapport à la formulation "classique", jusqu'ici la seule connue dans la littérature. En effet, nous montrons que la relaxation continue de cette nouvelle formulation fournit une borne au moins aussi bonne que la formulation classique et que cette inégalité est très souvent stricte.
2. Nous décrivons une méthode polynomiale de calcul d'une borne inférieure LB^* pour la valeur optimale du problème du p-Centre. Nous montrons que cette borne peut être vue comme la valeur optimale du programme linéaire en variables mixtes obtenu par relaxation des contraintes d'intégrité d'une partie des variables de la formulation nouvelle. Néanmoins, nous proposons une méthode "combinatoire" pour calculer cette borne LB^*. Par ailleurs, en comparant LB^* à l'algorithme 2-approché de Hochbaum et Shmoys [44], nous déduisons que, dans le cas où les distances sont Euclidiennes, LB^* est au moins égale à la moitié de la valeur optimale du problème du p-Centre.
3. Nous menons un important travail expérimental qui montre que la borne LB^* est de très bonne qualité pour toutes les instances déjà testées dans la littérature. De plus, elle se calcule très vite. La qualité de cette borne ouvre des perspectives intéressantes quant à la résolution exacte du problème du p-Centre. Nous avons implémenté une méthode de résolution en deux phases. La première phase consiste à calculer la borne LB^*, ce qui aboutit aussi à une réduction du problème. Ensuite, partant de LB^*, la deuxième phase cherche la plus petite distance telle que p centres suffisent à couvrir la totalité des clients à l'intérieur de cette distance. Cette méthode nous permet de résoudre plus rapidement toutes les instances déjà résolues par Daskin [24] et Ilhan et Pinar [45] et de résoudre des instances de taille plus importante, atteignant $N = M = 1817$.

Dans la suite de ce chapitre, nous étayons chacun des résultats ci-dessus puis nous consacrons la dernière section à des prolongements en cours ou potentiels de ce travail.

2.2 La formulation classique et la nouvelle formulation

Dans cette section, nous rappelons la formulation classique du problème du p-Centre par un programme linéaire en 0-1, que nous appelons (PC). Puis, nous proposons une nouvelle formulation, toujours par un programme linéaire en 0-1, que nous appelons $(PC - SC)$. Par le Théorème 2.1, nous prouvons que la formulation $(PC - SC)$ est en théorie supérieure à la formulation (PC). Enfin, nous présentons une comparaison expérimentale qui montre que si on dispose uniquement d'un solveur de programmes linéaires en nombres entiers pour résoudre pratiquement des instances de ce problème, on a nettement intérêt à choisir la formulation $(PC - SC)$ plutôt que la formulation $(PC - SC)$.

Le problème du p-Centre a toujours été formulé par (2.1)-(2.6) (voir par exemple [23]). Dans toute solution admissible (x,y,z): $y_j = 1$ ssi le site F_j est équipé, $x_{ij} = 1$ ssi le client C_i est affecté au centre situé en F_j, et z est une borne supérieure du rayon de cette solution admissible. Dans une solution optimale, $z = max_{i=1,\dots,N} \sum_{j=1}^{M} d_{ij}x_{ij}$ est le rayon optimal.

$$(PC) \quad : \min \quad z \tag{2.1}$$

sous les contraintes

$$\sum_{j=1}^{M} y_j \leq p \tag{2.2}$$

$$\sum_{j=1}^{M} x_{ij} = 1 \qquad i = 1,...,N \tag{2.3}$$

$$x_{ij} \leq y_j \qquad i = 1,...,N; j = 1,...,M \tag{2.4}$$

$$\sum_{j=1}^{M} d_{ij} x_{ij} \leq z \qquad i = 1,...,N \tag{2.5}$$

$$x_{ij}, y_j \in \{0,1\} \qquad i = 1,...,N; j = 1,...,M \tag{2.6}$$

La contrainte (2.2) limite le nombre de centres à au plus p, les contraintes (2.3) affectent chaque client à exactement un site, les contraintes (2.4) interdisent l'affectation d'un client à un site non équipé, et les contraintes (2.5) obligent z à être au moins égal à la distance entre n'importe quel client et le centre auquel il est affecté.

Notre nouvelle formulation du problème du p-Centre est basée sur sa relation bien connue avec le problème de recouvrement. Soit D_{min} (resp. D_{max}) la plus petite (resp. plus grande) valeur dans la matrice des distances, et soient $D_{min} = D^0 < D^1 < D^2 < ... < D^K = D_{max}$ les différentes valeurs des distances comprises entre D_{min} et D_{max}. La formulation $(PC - SC)$ utilise les variables $y_j, j = 1,...,M$, et $z^k, k = 1,...,K$. Pour toute solution réalisable (y,z), $y_j = 1$ ssi le site F_j est équipé et $z^k = 0$ s'il est possible de localiser p centres et couvrir tous les clients à l'intérieur du rayon D^{k-1}. Ainsi, $z^k = 0$ implique $z^{k+1} = ... = z^K = 0$, et la fonction objectif est strictement inférieure à D^k.

$$(PC - SC) \quad : \min \quad D^0 + \sum_{k=1}^{K} (D^k - D^{k-1}) z^k \tag{2.7}$$

sous les contraintes

$$\sum_{j=1}^{M} y_j \geq 1 \tag{2.8}$$

$$\sum_{j=1}^{M} y_j \leq p \tag{2.9}$$

$$z^k + \sum_{j : d_{ij} < D^k} y_j \geq 1 \qquad i = 1,...,N, k = 1,...,K \tag{2.10}$$

$$z^k \in \{0,1\} \qquad k = 1,...,K \tag{2.11}$$

$$y_j \in \{0,1\} \qquad j = 1,...,M \tag{2.12}$$

Les contraintes (2.8) interdisent les solutions n'ayant aucun site équipé et la contrainte (2.9) est la même que (2.2). Les coefficients dans la fonction objectif (2.7) sont tous positifs. Ainsi, dans toute solution optimale, les contraintes (2.10) disent que, pour une valeur donnée de k, $z^k = 0$, si et seulement si tous les clients peuvent être desservis par des centres situés à une distance strictement inférieure à D^k. Quand $z^k = 1$, une distance égale à $(D^k - D^{k-1})$ est ajoutée au rayon dans la fonction objectif (2.7). Notons que la formulation $(PC - SC)$ ne donne pas explicitement l'affectation des clients aux sites. Une fois calculée une solution $(\tilde{z},\tilde{y})$ de $(PC - SC)$, il suffit d'affecter chaque client C_i au centre en F_{j_0}tel que $d_{ij_0} = min_{j:\tilde{y}_j=1} d_{ij}$.

Les variables z^k $(k = 1,\ldots,K)$ explicitent également la relation qui existe entre le problème du p-Centre et le problème de recouvrement. Précisément, $z^k = 0$ dans une solution optimale de $(PC - SC)$ si et seulement s'il existe une couverture des N clients par au plus p centres dans un rayon égal à D^{k-1}. Autrement dit, si et seulement si la valeur optimale du problème de recouvrement $SC_{D^{k-1}}$ est inférieure ou égale p.

$$(SC_\delta) \quad : \min \quad \sum_{j=1}^{M} y_j \tag{2.13}$$

sous les contraintes

$$\sum_{j:d_{ij}\leq\delta} y_j \geq 1 \qquad i = 1,...,N \tag{2.14}$$

$$y_j \in \{0,1\} \qquad j = 1,...,M \tag{2.15}$$

Soit $v(SC_\delta)$ la valeur d'une solution optimale de SC_δ, avec $v(SC_\delta) = \infty$ si SC_δ ne contient aucune solution admissible. Il est clair que $v(SC_{D^0}) \geq v(SC_{D^1})... \geq v(SC_{D^K})$. De plus, soit k^* le plus petit k dans $0,...,K$ tel que $v(SC_{D^k}) \leq p$ et soit $\tilde{y}$ une solution optimale de $SC_{D^{k^*}}$. Alors une solution $(\tilde{y},\tilde{z})$ existe pour $(PC - SC)$ où $\tilde{z}^1 = ... = \tilde{z}^{k^*} = 1$ et $\tilde{z}^{k^*+1} = ... = \tilde{z}^K = 0$. Ainsi, la résolution de $(PC - SC)$ peut se faire par la résolution d'au plus $(K + 1)$ problème de recouvrement.

Pour comparer les deux formulations (PC) et $(PC - SC)$, nous comparons d'abord leurs relaxations linéaires.

Théorème 2.1 *Soient (LPC) et $(LPC - SC)$ les deux programmes linéaires obtenus quand on relâche les contraintes d'intégrité de toutes les variables des programmes (PC) et $(PC - SC)$. Alors,*

$$v(LPC - SC) \geq v(LPC).$$

■

L'exemple suivant montre que la domination de (LPC-SC) sur (LPC) peut être stricte. Soit l'instance symétrique telle que $N = M = 3$, $p = 2$, $d_{ii} = 0$ $(i = 1,\ldots,3)$, $d_{12} = d_{13} = 2$, et $d_{23} = 1$. Considérons les bornes triviales $LB = 0$ et $UB = 2$. Ici, $D^0 = 0$, $D^1 = 1$, et $D^2 = 2$. Une solution optimale pour (LPC) est $\hat{z} = \frac{2}{5}$, $\hat{y}_1 = \frac{4}{5}$, $\hat{y}_2 = \hat{y}_3 = \frac{3}{5}$, $\hat{x}_{11} = \frac{4}{5}$, $\hat{x}_{13} = \frac{1}{5}$, $\hat{x}_{22} = \hat{x}_{33} = \frac{3}{5}$, et $\hat{x}_{23} = \hat{x}_{32} = \frac{2}{5}$. Ainsi, $v(LPC) = \frac{2}{5}$. Une solution optimale pour $(LPC - SC)$ est $\tilde{z}^1 = \frac{1}{2}$,

$\tilde{z}^2 = 0$, $\tilde{y}_1 = 1$, et $\tilde{y}_2 = \tilde{y}_3 = \frac{1}{2}$. Cela donne $v(LPC - SC) = \frac{1}{2} > v(LPC)$. Remarquons que pour cet exemple, la valeur optimale en nombres entiers est 1. C'est aussi la valeur de la borne inférieure LB^* que nous décrivons ci-dessous.

On peut aussi comparer ces formulations pour le critère de temps de calcul. La formulation (PC) contient $M(N+1)$ variables binaires, 1 variable continue et $(1+N(M+2))$ contraintes. La taille de la formulation $(PC-SC)$ dépend du nombre de distances différentes dans la matrice des distances, mais ce nombre est limité par le nombre de cases de cette matrice, soit $K \leq (NM-1)$. Le problème $(PC-SC)$ contient $K+M$ variables binaires et $(2+KN)$ contraintes. Ainsi, dans le cas extrême où toutes les distances sont différentes, $(PC-SC)$ et (PC) contiennent sensiblement le même nombre de variables mais $(PC - SC)$ contient de l'ordre de N fois plus de contraintes. Cependant, en pratique, d'une part le nombre de distances différentes n'est pas aussi important, et d'autre part, la taille du programme $(PC - SC)$ peut être considérablement réduite par la connaissance d'une borne inférieure ou d'une borne supérieure. En effet, si $\underline{r}$ (resp. $\bar{r}$) est la valeur d'une borne inférieure (resp. supérieure) sur le rayon optimal alors toute variable z^k telle que $D^k < \underline{r}$ (resp. $D^k > \bar{r}$) peut être fixée à 1 (resp. 0) et les N contraintes correspondantes (2.9) éliminées. La connaissance des mêmes bornes $\underline{r}$ et $\bar{r}$ ont un effet plus modeste sur le programme (PC) où on peut seulement fixer à 0 toute variable x_{ij} telle que $d_{ij} > \bar{r}$. Pour être totalement persuadé de la supériorité de la formulation $(PC-SC)$, nous montrons que, utilisant directement un solveur de programmes linéaires en nombres entiers, et limitant le temps de calcul à 1h, on arrive à résoudre à l'optimum des instances non résolues par la formulation (PC) et à aller plus vite pour d'autres instances. Voici plus précisément les conditions de ces expérimentations:

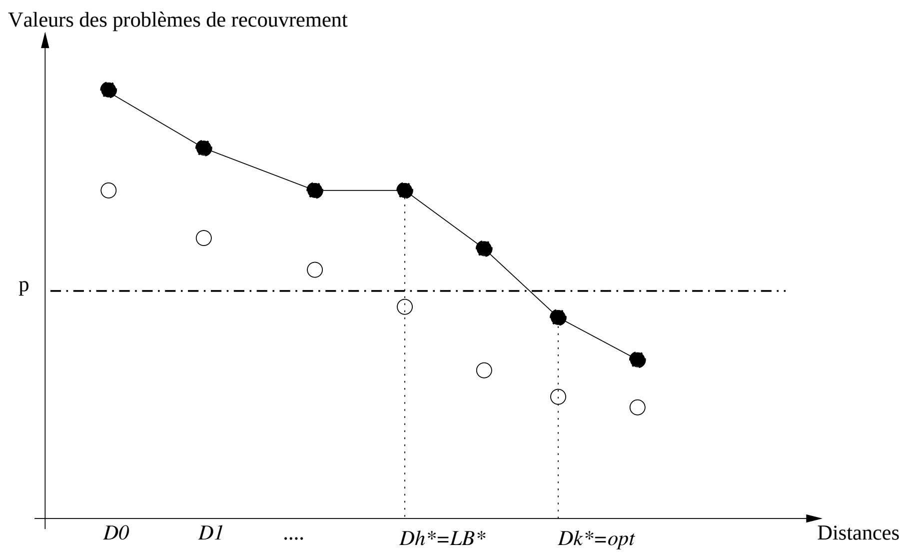
- nous décrivons un nouvel algorithme de calcul d'une borne inférieure $\underline{r}_1$, basé sur une notion de regret associé à la non-ouverture d'un centre,
- nous utilisons la borne supérieure $\bar{r}_1$ donnée par la solution d'une méthode gourmande, utilisée par Mladénovic et al. [54],
- les outils logiciels que nous utilisons sont AMPL [32] pour la modélisation et Cplex7.0 [21] pour la résolution des programmes linéaires en nombres entiers, sur une machine de type ordinateur personnel,
- nous utilisons 5 instances de ORlib [4], crées à l'origine pour le problème de la p-médiane mais déjà considérées par d'autres auteurs pour le problème du p-Centre ([54], [24], [45]). Pour toutes ces instances, $N = M = 100$ et $p = 5$, 10, 20 ou 33 et les distances sont euclidiennes.

Notre expérimentation montre qu'en utilisant la formulation (PC), le solveur s'arrête au bout d'une heure de calcul avec une erreur (écart relatif entre la meilleure solution et la meilleure borne inférieure trouvées) de 22% pour une instance et 32% pour une autre instance. Pour les 3 autres instances, une solution optimale est trouvée après, en moyenne, 4330 nœuds et 2000 secondes. La formulation $(PC - SC)$ permet de résoudre les 5 instances après, en moyenne, 350 nœuds et 1240 secondes.

2.3 Un algorithme polynomial pour calculer une borne inférieure LB^*

Dans cette section, nous définissons la borne inférieure LB^* et proposons une méthode polynomiale pour la calculer. Puis, dans la Proposition 2.1, nous donnons un résultat de performance au pire de cette borne.

Soit $(PC-SC-R)$ le programme linéaire en variables mixtes obtenu à partir de $(PC-SC)$ en relâchant les contraintes d'intégrité sur les variables y seulement, c'est à dire le problème (2.7)-(2.11) avec $0 \leq y_j \leq 1$ pour $j = 1,...,M$. Soit LB^* la valeur optimale de $(PC-SC-R)$. Nous montrons qu'il existe h^* tel que $LB^* = D^{h^*}$. En effet, de même que le rayon optimal D^{k^*} peut être vu comme la plus petite distance D^k telle que $v(SC_{D^k}) \leq p$, une borne inférieure sur D^{k^*} peut être obtenue en calculant la plus petite distance D^h telle que $v(LSC_{D^h}) \leq p$ où le problème (LSC_δ) est obtenu en relâchant les contraintes d'intégrité dans (SC_δ). La Figure 2.1 illustre les définitions de h^* et k^* et la relation entre eux. Quand les deux valeurs coïncident pour une instance, cela veut dire qu'il n'y a pas de saut d'intégrité entre les problèmes $(PC-SC-R)$ et $(PC-SC)$.



$\bullet$: $v(SC_{D^k})$ $\circ$: $v(LSC_{D^k})$.

FIG. 2.1 – *Illustration de la borne $LB^* = D^{h^*}$ et de la valeur optimale D^{k^*} pour le problème du p-Centre*

Le calcul de LB^* peut se faire en temps polynomial. En effet, il nécessite la résolution d'au plus $K+1$ programmes linéaires, avec $K \leq (NM-1)$. Chacun de ces programmes linéaires contient M variables et N contraintes. On commence à $h = 0$, on résout successivement des problèmes (LSC_{D^h}) et on s'arrête dès que $v(LSC_{D^h}) \leq p$. Mieux, on peut tenir compte du fait que les valeurs $v(LSC_{D^h})$ sont ordonnées et utiliser la dichotomie pour résoudre au maximum $log_2(K+1)$ programmes linéaires.

Performance au pire de la borne LB^*

Considérant le cas où les centres sont à placer sur les sites des clients ($N = M$ et $d_{ii} = 0\ \forall i$) et où les distances sont Euclidiennes, Hochbaum et Shmoys [44] proposent une heuristique 2-approchée pour ce problème. Ce résultat d'approximation est basé sur la relation du p-Centre avec le problème de l'ensemble stable de cardinal maximal. Nous proposons ci-dessous une extension des résultats de Hochbaum et Shmoys.

Pour une distance D^k, soit $G_{D^k} = (V,V',E)$ le graphe biparti où V est l'ensemble des clients, V' l'ensemble des sites potentiels des centres, et une arête de E relie un clients et un site si est seulement si la distance entre eux est inférieure ou égale à D^k. Ce graphe est associé au problème de recouvrement SC_{D^k}. Soit par ailleurs le "graphe de voisinage" $\Delta_{D^k} = (V,E')$ où deux clients C_i et $C_{i'}$ sont reliés ss'ils peuvent être servis par le même centre à l'intérieur de la distance D^k, ou encore s'il existe un site potentiel $F_j \in V'$ tel que $(C_i,F_j) \in E$ et $(C_{i'},F_j) \in E$. On montre aisément que le problème de la recherche d'un ensemble stable de cardinal maximal dans le graphe Δ_{D^k} est le dual (faible) du problème (SC_{D^k}) de recherche d'un recouvrement de cardinal minimal dans le graphe G_{D^k}. La Figure 2.2 illustre ces deux graphes pour un exemple avec 5 clients et 4 sites potentiels.

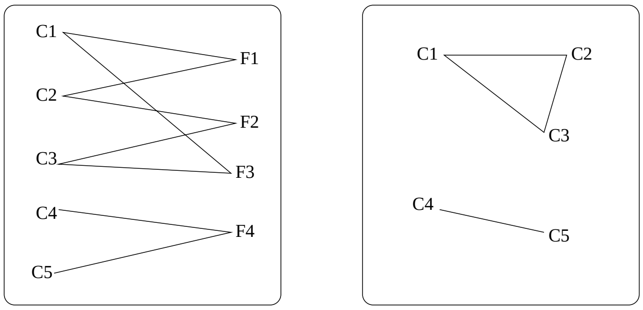


FIG. 2.2 – *Illustration des graphes G_{D^k} et Δ_{D^k}*

La proposition suivante montre la relation entre le cardinal d'un ensemble stable de Δ_{D^k}, la borne LB^* et la valeur optimale du problème du p-Centre.

Proposition 2.1 *Pour une distance donnée D^k, soit $S \subset V$ un sous-ensemble de clients qui forment un ensemble stable de cardinal maximal pour le graphe Δ_{D^k}.*

(i) si $|S| > p$ alors $D^{k+1} \leq LB^* \leq D^{k^*}$ (D^{k+1} est une borne inférieure du rayon optimal pour le problème du p-Centre).

(ii) si $|S| \leq p$ alors $D^{k^*} \leq 3D^k$ (le rayon optimal est au plus $3D^k$). ■

Preuve
Pour $s \in S$, soit $\phi(s) \in V'$ un site potentiel tel que $(s,\phi(s)) \in E$. Par définition de Δ_{D^k} et puisque S est un ensemble stable pour le graphe Δ_{D^k} alors $\phi(s') \neq \phi(s)\ \forall s' \neq s$ et il faut au moins $|S|$ centres pour servir les clients de S, à l'intérieur de la distance D^k.

(i) si $|S| > p$ alors $D^{k+1} \leq LB^*$ découle de la dualité faible.

(ii) si $|S| \leq p$ alors on construit l'ensemble de sites $\Phi(S) = \{\phi(s) : s \in S\}$. Ainsi, $|\Phi(S)| = |S|$. Deux cas sont alors possibles :

1. $\Phi(S)$ est une solution pour le problème (SC_{D^k}), i.e. chaque client C_i est relié à un site de $\Phi(S)$, dans le graphe G_{D^k}. Dans ce cas, $\Phi(S)$ est une solution optimale pour le problème (SC_{D^k}) et c'est une solution pour le problème du p-Centre, de rayon au plus égal à D^k. Donc $D^{k^*} \leq D^k$.
2. $\Phi(S)$ n'est pas pas une solution pour le problème SC_{D^k}, i.e. il existe au moins un client C_i qui n'est relié à aucun site de $\Phi(S)$, dans le graphe G_{D^k}. Mais dans ce cas, puisque S est stable de cardinal maximal, C_i est obligatoirement relié, dans le graphe Δ_{D^k}, à un client $C_{i'}$ de S. Ainsi, il existe un site $F_j \neq \phi(i')$ tel que $(i,F_j) \in E$ et $(i',F_j) \in E$.

$$i \overset{\leq D^k}{\text{———}} F_j \overset{\leq D^k}{\text{———}} i' \overset{\leq D^k}{\text{———}} \phi(i')$$

Ainsi, si les distances vérifient les inégalités triangulaires alors $\Phi(S)$ constitue une solution pour le problème du p-Centre, de rayon inférieur ou égal à $3D^k$.

■

Corollaire 2.1 *Si les distances vérifient les inégalités triangulaires alors* $LB^* \geq \frac{1}{3}D^{k^*}$. ■

Preuve
Ce corollaire découle du fait qu'en général, il existe k tel que :

- le cardinal maximal d'un ensemble stable de Δ_{D^k} est $\leq p$
- le cardinal maximal d'un ensemble stable de $\Delta_{D^{k-1}}$ est $> p$

La Propriété ci-dessus nous dit alors que d'une part $D^k \leq LB^*$ et d'autre part $D^{k^*} \leq 3D^k$. ■

Remarque
Dans le cas où les centres sont à placer sur les sites des clients ($N = M$ et $d_{ii} = 0\ \forall i$), il est facile de vérifier que la propriété *(ii)* de la Proposition 2.1 se spécialise en $D^{k^*} \leq 2D^k$, ce qui a déjà été démontré par Hochbaum et Shmoys et nous permet d'affirmer que, dans ce cas, $LB^* \geq \frac{1}{2}D^{k^*}$.

2.4 Travail expérimental

Dans cette section, nous décrivons les algorithmes de calcul de la borne LB^* et de résolution exacte que nous avons implémentés et nous présentons les résultats de plusieurs expérimentations effectuées sur des instances de la littérature. Nos résultats montrent la très bonne qualité de cette borne inférieure pour toutes les instances Euclidiennes. Notre méthode exacte résout des instances de taille environ deux fois plus importante que celles décrites dans la littérature très récente. Enfin, nous montrons aussi que la qualité de cette borne est sensible au caractère Euclidien des distances. Elle se dégrade nettement quand les distances sont tirées aléatoirement.

Calcul de la borne inférieure LB^* par l'algorithme *Bsearch*

Rappelons que $LB^* = D^{h^*}$ où D^{h^*} est la plus petite distance telle que la valeur optimale du programme linéaire $(LSC_{D^{h^*}})$ est inférieure ou égale à p. Nous avons implémenté *Bsearch*, un algorithme basé sur la dichotomie, pour calculer h^* et donc LB^*. En plus de LB^*, notre algorithme fournit aussi la valeur d'une solution admissible UB^*. Cette valeur UB^* est initialisée à la valeur de la solution de la méthode gourmande pour le problème du p-Centre (voir [54]), puis mise à jour à chaque fois qu'une meilleure solution admissible est trouvée. A chaque fois qu'on résout un problème (LSC_{D^h}), on essaie aussi de trouver une solution entière, gourmande ou par arrondi de (SC_{D^h}), et d'en déduire une solution admissible pour le problème du p-Centre.

La Table 2.1 donne les résultats d'exécution de notre algorithme *Bsearch* sur les 40 instances d'OR-Lib [5] initialement prévues pour le problème de la p-médiane. Les trois premières colonnes donnent les caractéristiques de chaque instance, *Opt* est la valeur optimale de chaque instance, les colonnes suivantes contiennent les valeurs calculées par l'algorithme *Bsearch*, ainsi que le temps de calcul qu'il nécessite. On peut observer que $LB^* < Opt$ uniquement pour les 4 instances *Pmed1, Pmed6, Pmed26, et Pmed32*. Pour 29 sur les 36 instances restantes, $LB^* = UB^*$, c'est à dire que l'algorithme *Bsearch* tout seul résout chacune de ces instances. Le temps moyen de l'algorithme pour les 40 instances est de 4.5 secondes. Pour l'ensemble de ces 40 instances, on peut dire que la borne LB^* est de très bonne qualité et que l'algorithme *Bsearch* est très efficace pour la calculer. Pour affiner la comparaison, nous avons tenté de calculer LB^* en utilisant un solveur de programmes linéaires en nombres entiers pour calculer la valeur optimale de $(PC - SC - R)$. Pour l'instance *Pmed1*, $(PC - SC - R)$ contient 137 variables 0-1, 100 variables continues et 13901 contraintes. Cplex7.0 a besoin de 40 minutes et 178 nœuds de *Branch & Bound* pour calculer LB^*, soit environ 6000 fois plus de temps que *Bsearch*.

<u>Résolution exacte par l'algorithme *Bsearch-Ex*</u>

Une fois calculée la borne inférieure LB^* et la borne supérieure UB^* par l'algorithme *Bsearch*, on peut continuer à rechercher la valeur optimale du problème du p-Centre, dans l'intervalle $[LB^*,UB^*]$. Pour ce faire, nous avons implémenté l'algorithme *Bsearch-Ex*, qui est très semblable à *Bsearch*, mais qui résout des programmes linéaires en nombres entiers (SC_{D^h}) plutôt que leur relaxation continue (LSC_{D^h}). Rappelons que la valeur optimale du problème du p-Centre est D^{k^*}, la plus petite distance telle que $v(LSC_{D^{k^*}}) \leq p$. L'algorithme *Bsearch-Ex* procède donc à une recherche dichotomique de k^* dans l'intervalle $[h^*,k']$ où k' est tel que $D^{k'} = UB^*$.

A chaque itération de sa recherche dichotomique, l'algorithme *Bsearch-Ex* résout un programme linéaire en variables 0-1. En fait, il n'a pas réellement besoin de résoudre ce problème à l'optimalité mais de répondre à la question: "A-t-il une solution de valeur inférieure ou égale à p?". Pour ce faire, nous utilisons Cplex7.0 et changeons certains de ses paramètres par défaut:

- *mip tolerances uppercutoff* est mis à $(p + 0.001)$, pour éliminer les solutions de valeur $> p$,
- *mip limits solutions* est mis à 1, pour que la résolution s'arrête dès qu'une solution entière (de valeur au plus p) est trouvée,
- *timelimit* est mis à 3600, pour que la résolution s'arrête au bout d'une heure de temps CPU. Si cela se produit, l'algorithme *Bsearch-Ex* fait comme s'il n'existait pas de solution de valeur inférieure à p pour le problème en cours et donc on ne peut plus garantir que la valeur de la solution admissible qu'il calcule au final soit optimale.

Nous avons appliqué l'algorithme *BsearchEx* aux 11 instances de la Table 2.1 pour lesquelles $LB^* < UB^*$. Pour l'ensemble de ces 11 instances, *BsearchEx* a dû résoudre 14 problèmes de recouvrement (SC_δ) et a nécessité 123 secondes de temps CPU. La Table 2.2 donne d'abord les résultats concernant ces 11 instances puis pour des instances initialement prévues pour le problème du voyageur du commerce (TSP) prises dans TSPLIB [58]. Ces instances ont déjà été

Instance	N=M	P	Opt	LB*	UB*	T(s)
Pmed1	100	5	127	121	127	0.2
Pmed2	100	10	98	98	98	0.2
Pmed3	100	10	93	93	93	0.1
Pmed4	100	20	74	74	74	0.1
Pmed5	100	33	48	48	48	0.1
Pmed6	200	5	84	83	84	0.8
Pmed7	200	10	64	64	64	0.5
Pmed8	200	20	55	55	55	0.4
Pmed9	200	40	37	37	37	0.1
Pmed10	200	67	20	20	20	0.3
Pmed11	300	5	59	59	61	1.1
Pmed12	300	10	51	51	51	1.3
Pmed13	300	30	36	36	36	0.8
Pmed14	300	60	26	26	26	0.9
Pmed15	300	100	18	18	18	1.0
Pmed16	400	5	47	47	47	1.6
Pmed17	400	10	39	39	39	2.1
Pmed18	400	40	28	28	28	1.4
Pmed19	400	80	18	18	19	0.4
Pmed20	400	133	13	13	13	1.8
Pmed21	500	5	40	40	40	5.2
Pmed22	500	10	38	38	39	6.9
Pmed23	500	50	22	22	23	2.1
Pmed24	500	100	15	15	15	4.5
Pmed25	500	167	11	11	11	2.7
Pmed26	600	5	38	37	39	8.8
Pmed27	600	10	32	32	32	8.2
Pmed28	600	60	18	18	18	2.1
Pmed29	600	120	13	13	13	5.1
Pmed30	600	200	9	9	9	5.4
Pmed31	700	5	30	30	30	8.1
Pmed32	700	10	29	28	30	13.2
Pmed33	700	70	15	15	16	4.3
Pmed34	700	140	11	11	11	6.5
Pmed35	800	5	30	30	30	13.7
Pmed36	800	10	27	27	29	21.2
Pmed37	800	80	15	15	15	2.0
Pmed38	900	5	29	29	29	18.5
Pmed39	900	10	23	23	24	21.2
Pmed40	900	90	13	13	13	7.8
Average						4.5

TAB. 2.1 – *Résultats de l'algorithme Bsearch pour les 40 instances d'ORlib*

utilisées par Mladenovic et al. [54]. Si n est le nombre de villes, on crée des instances du problème du p-Centre en prenant $N = M = n$ et en considérant différentes valeurs de p, typiquement 10, 20, 30, etc.... Ainsi, toutes les instances de la Table 2.2 sont telles que $N = M$, les distances sont symétriques et Euclidiennes. Dans cette table, les trois premières colonnes caractérisent l'instance. La colonne suivante donne soit la valeur optimale soit la valeur de la meilleure solution connue. Les colonnes 5 à 8 donnent les résultats de l'algorithme *Bsearch*: $\#LSC_\delta$ est le nombre de problèmes (LSC_δ) qui ont été résolus et *cpu1* est le temps CPU de l'algorithme *Bsearch*. La colonne 9 contient $\#SC_\delta$, le nombre de problèmes (SC_δ) que l'algorithme *BsearchEx* a dû résoudre. La dernière colonne, *cpu2* est le temps CPU total mis pour la résolution de l'instance, c'est à dire pour exécuter la méthode gourmande, trier les distances et exécuter les algorithmes *Bsearch* et *BsearchEx*.

Daskin [24] présente les résultats de sa méthode de résolution exacte pour les 40 instances d'ORlib. Même s'il n'est pas très correct de comparer des temps de calcul sur des machines différentes, on peut donner comme indication le fait que nous utilisons le même type de machines, à environ une année d'intervalle. Notre temps moyen de calcul pour ces 40 instances est d'environ 9 secondes avec un maximum de l'ordre d'une minute alors que les résultats de Daskin montrent une moyenne d'environ 90 seconds avec un maximum à environ 23 minutes.

Une autre comparaison peut être faite avec le travail de Mladenovic et al. [54] qui ont récemment conçu et testé des heuristiques pour le problème du p-Centre. Nous pouvons, à posteriori, calculer l'erreur moyenne de leur méthode de recherche par voisinage variable *VNS*, la meilleure parmi celles qu'ils ont testées. Cette erreur moyenne est de 2.37% pour les 40 instances d'ORlib (temps CPU moyen de 133.6 secondes pour la méthode *VNS*) et de 0.02% (temps CPU moyen de 300 secondes) pour les 15 instances u1060.tsp de TSPLIB.

Nous avons également voulu tester la qualité de la borne LB^* quand les distances ne vérifient pas l'inégalité triangulaire. Pour ce faire, nous avons généré des instances *Rand100* avec $N = M = 100$ et les distances sont générées aléatoirement dans l'intervalle [0,100[, avec $d_{ii} = 0$ et $d_{ji} = d_{ij}$. La Table 2.3 donne les résultats obtenus pour ces instances. On observe que la borne LB^* est nettement moins fine pour ces instances non-Euclidiennes. L'erreur relative par rapport à la valeur optimale est de 28% pour $p = 5$, 30% pour $p = 10$ et 14% pour $p = 15$, alors que cette erreur ne dépasse jamais 5% pour l'ensemble des instances Euclidiennes que nous avons testées.

Instance	$N = M$	p	**Opt**	LB^*	UB^*	$\#LSC_\delta$	*cpu1*	$\#SC_\delta$	*cpu2*
Pmed1	100	5	127	121	127	7	0.2	3	0.7
Pmed6	200	5	84	83	84	6	0.8	1	0.3
Pmed11	300	5	59	59	61	4	1.1	1	1.0
Pmed19	400	80	18	18	19	4	0.4	1	0.4
Pmed22	500	10	38	38	39	5	6.9	1	4.3
Pmed23	500	50	22	22	23	4	2.1	1	1.2
Pmed26	600	5	38	37	39	6	8.8	1	6.1
Pmed32	700	10	29	28	30	5	13.2	2	45.2
Pmed33	700	70	15	15	16	3	4.3	1	3.1
Pmed36	800	10	27	27	29	5	21.2	1	34.5
Pmed39	900	10	23	23	24	5	21.2	1	27.3
u1060	1060	10	2273	2273	2273	6	27	0	53
u1060	1060	20	1581	1556	1768	9	63	6	2778
u1060	1060	30	1208	1205	1275	8	50	4	298
u1060	1060	40	1021	1013	1079	9	35	4	366
u1060	1060	50	905	895	963	6	21	4	383
u1060	1060	60	781	765	807	8	21	4	233
u1060	1060	70	711	707	761	6	17	3	135
u1060	1060	80	652	652	711	6	18	2	60
u1060	1060	90	608	604	636	6	19	3	38
u1060	1060	100	570	570	570	5	18	0	29
u1060	1060	110	539	539	539	5	18	0	30
u1060	1060	120	510	510	538	7	29	1	44
u1060	1060	130	500	495	510	6	28	1	44
u1060	1060	140	452	452	500	5	28	3	46
u1060	1060	150	447	430	447	6	34	3	50
rl1323	1323	10	3077	3062	3329	11	106	8	1380
rl1323	1323	20	2016	2008	2152	11	115	5	480
rl1323	1323	30	1632	1611	1797	11	99	7	900
rl1323	1323	40	1352	1334	1521	10	76	7	3000
rl1323	1323	50	1187	1165	1300	9	61	7	8580
rl1323	1323	60	1063	1047	1194	11	55	8	9120
rl1323	1323	70	972	959	1040	10	42	7	1740
rl1323	1323	80	895	889	948	10	37	5	420
rl1323	1323	90	832	830	857	10	30	4	120
rl1323	1323	100	787	777	803	9	26	5	120
u1817	1817	10	458	455	467	8	611	4	2700
u1817	1817	20	310?	306	342	9	660	5	4920
u1817	1817	30	250?	240	287	8	355	6	16500
u1817	1817	40	210?	205	234	6	247	4	6420
u1817	1817	50	187?	180	205	8	242	4	9840
u1817	1817	60	163	163	183	6	177	4	1260
u1817	1817	70	148	148	152	6	166	1	420
u1817	1817	80	137	137	148	5	150	3	1140
u1817	1817	90	130?	127	148	6	161	3	7202
u1817	1817	100	127	127	130	6	159	1	300
u1817	1817	110	109	108	127	5	119	4	420
u1817	1817	120	108	108	108	6	131	0	120
u1817	1817	130	108?	105	109	4	121	1	3720
u1817	1817	140	105?	102	108	5	121	2	4020
u1817	1817	150	94?	92	108	5	144	3	5640

TAB. 2.2 – *Résultats de notre méthode exacte pour des instances de OR-Lib et de TSPLIB*

? : Opt est la valeur de la meilleure solution connue pour cette instance

Instance	$N = M$	p	**Opt**	LB^*	UB^*	$\#LSC_\delta$	*cpu1*	$\#SC_\delta$	*cpu2*
Rand100	100	5	28	20	39	6	0.4	4	37.2
Rand100	100	10	13	10	17	5	0.2	3	42.5
Rand100	100	15	7	6	10	4	0.2	2	4.9

TAB. 2.3 – *Résultats de notre méthode exacte pour des instances non Euclidiennes*

2.5 Prolongements à des généralisations du p-Centre

Des publications récentes [48] généralisent le problème du p-Centre pour introduire une notion de tolérance aux pannes ou pour tenir compte de contraintes de capacité sur les centres. Notre façon de traiter le problème du p-Centre se généralise également.

Prolongement pour le problème du p-Centre α-tolérant

Dans ce problème, il faut ouvrir p centres et affecter chaque client aux α centres ouverts les plus proches, de façon à minimiser la distance maximale entre un client et un centre auquel il est affecté. Notre formulation $(PC - SC)$ s'adapte aisément à ce nouveau problème :

$$(PC - SC^\alpha) \quad : \ \min \quad D^0 + \sum_{k=1}^{K}(D^k - D^{k-1})z^k \tag{2.16}$$

sous les contraintes

$$\sum_{j=1}^{M} y_j \geq 1 \tag{2.17}$$

$$\sum_{j=1}^{M} y_j \leq p \tag{2.18}$$

$$\alpha z^k + \sum_{j:d_{ij}<D^k} y_j \geq \alpha \qquad i = 1,...,N; \ k = 1,...,K \tag{2.19}$$

$$z^k \in \{0,1\} \qquad k = 1,...,K \tag{2.20}$$

$$y_j \in \{0,1\} \qquad j = 1,...,M \tag{2.21}$$

Nous montrons dans [28] que notre méthode de calcul de borne et notre méthode de résolution exacte se généralisent également. Le sous-problème que l'on résout à chaque itération de la recherche dichotomique (autrement dit, pour une distance donnée) peut être vu comme une généralisation du problème de recouvrement :

$$(SC_\delta^\alpha) \quad : \ \min \quad \sum_{j=1}^{M} y_j \tag{2.22}$$

sous les contraintes

$$\sum_{j:d_{ij}\leq\delta} y_j \geq \alpha \qquad i = 1,...,N \tag{2.23}$$

$$y_j \in \{0,1\} \qquad j = 1,...,M \tag{2.24}$$

Nos résulats de calcul sur les 40 instances d'ORlib montrent que la borne LB^*_α ainsi obtenue est d'aussi bonne qualité que LB^* pour $\alpha = 2$ et $\alpha = 3$.

Prolongement pour le problème du p-Centre avec capacités

Cette fois, p centres sont ouverts et un client est affecté à exactement un centre mais un centre ne peut pas desservir plus de L clients.

Pour généraliser la formulation $(PC - SC)$, nous sommes cette fois amenés à considérer les variables x^{ij} d'affectation des clients aux centres. On obtient la formulation suivante :

$$(PC - SC^L) \quad : \quad \min \quad D^0 + \sum_{k=1}^{K} (D^k - D^{k-1}) z^k \tag{2.25}$$

sous les contraintes

$$\sum_{j=1}^{M} y_j \geq 1 \tag{2.26}$$

$$\sum_{j=1}^{M} y_j \leq p \tag{2.27}$$

$$z^k + \sum_{j: d_{ij} < D^k} x_{ij} \geq 1 \qquad i = 1,\ldots,N; \; k = 1,\ldots,K \tag{2.28}$$

$$x_{ij} \leq y_j \qquad i = 1,\ldots,N; \; j = 1,\ldots,M \tag{2.29}$$

$$\sum_{i=1}^{N} x_{ij} \leq L y_j \qquad j = 1,\ldots,M \tag{2.30}$$

$$\sum_{j=1}^{M} x_{ij} = 1 \qquad i = 1,\ldots,N \tag{2.31}$$

$$z^k \in \{0,1\} \qquad k = 1,\ldots,K \tag{2.32}$$

$$x_{ij} \in \{0,1\} \quad y_j \in \{0,1\} \qquad i = 1,\ldots,N; \; j = 1,\ldots,M \tag{2.33}$$

Notons que les contraintes (2.29) sont redondantes si les variables sont entières mais ne le sont pas si les variables sont continues. Cette fois, le sous-problème que l'on résout à chaque itération de la dichotomie est :

$$(SC^L_\delta) \quad : \quad \min \quad \sum_{j=1}^{M} y_j \tag{2.34}$$

sous les contraintes

$$\sum_{j: d_{ij} \leq \delta} x_{ij} = 1 \qquad i = 1,\ldots,N \tag{2.35}$$

$$x_{ij} \leq y_j \qquad i = 1,\ldots,N; \; j = 1,\ldots,M \;\; : d_{ij} \leq \delta \tag{2.36}$$

$$\sum_{i: d_{ij} \leq \delta} x_{ij} \leq L y_j \qquad j = 1,\ldots,M \tag{2.37}$$

$$x_{ij} \in \{0,1\} \quad y_j \in \{0,1\} \qquad i = 1,\ldots,N; \; j = 1,\ldots,M \tag{2.38}$$

Il s'agit d'un problème de localisation simple avec capacités. Une étude expérimentale est en cours, dont le but est d'évaluer la qualité de la borne LB_L^*.

2.6 Extension à un problème de codage avec un critère min-max

Un autre prolongement notre travail sur le problème du p-Centre consiste à étendre notre démarche à d'autres problèmes où l'objectif est de type *Min-Max*. Par exemple, à des problèmes "génériques" comme le partitionnement d'ensembles en minimisant la dissimilarité entre objets. Dans un premier temps, nous nous intéressons à un problème de codage, qui se pose en Théorie de l'Information. Le problème est, connaissant N données et une certaine distance entre elles, d'attribuer une abscisse et une ordonnée à chacune des données de façon à minimiser l'erreur potentielle maximale. Cette erreur, pour une solution donnée, se mesure comme la distance maximale entre deux données ayant la même abscisse ou la même ordonnée (deux données différentes ne peuvent pas avoir une même abscisse et une même ordonnée). Il s'agit de nouveau d'un problème de Min-Max auquel nous avons réussi à appliquer les mêmes idées qu'au problème du p-Centre.

Description du problème de codage

Etant donnés :

- N points P_1, P_2, ..., P_N qui représentent des vecteurs d'information. Ces points seront transmis via deux canaux de communication,
- d_{ij} $(i < j)$ la distance entre P_i et, P_j,
- n_1 limite du premier canal de communication,
- n_2 limite du deuxième canal de communication (autrement dit, le premier canal transmet $log n_1$ bits et le deuxième $log n_2$ bits.

Une solution admissible consiste à affecter à chaque point P_i un code $a(i)$ dans le premier canal de communication $(1 \leq a(i) \leq n_1)$ et un code $b(i)$ dans le deuxième canal de communication $(1 \leq b(i) \leq n_2)$. Les couples $(a(i),\ b(i))$ doivent être tous différents pour qu'un tel couple caractérise un point mais plusieurs points peuvent avoir le même code $a(i)$ ou le même code $b(i)$. L'erreur potentielle associée à une solution est égale à la distance maximale entre 2 points ayant le même code $a(i)$ ou le même code $b(i)$. Le but est de minimiser l'erreur potentielle maximale, ou, plus formellement, trouver e^* telle que :

$$e^* = \underset{a,\ b}{\mathrm{Min}} \left(\underset{i,j:\ a(i)=a(j)\ ou\ b(i)=b(j)}{\mathrm{Max}} d_{ij} \right)$$

Un exemple pour le problème de codage

Voici un exemple avec $N = 9$:

d	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9
P_1	0	93	75	41	37	96	70	29	30
P_2	93	0	38	58	97	48	92	70	9
P_3	75	38	0	70	52	29	30	3	22
P_4	41	58	70	0	15	59	55	56	82
P_5	37	97	52	15	0	92	40	76	64
P_6	96	48	29	59	92	0	24	16	99
P_7	70	92	30	55	40	24	0	81	56
P_8	29	70	3	56	76	16	81	0	70
P_9	30	9	22	82	64	99	56	70	0

et une solution avec $n_1 = n_2 = 3$:

	$b=1$	$b=2$	$b=3$
$a=1$	P_1	P_4	P_5
$a=2$	P_8	P_2	P_9
$a=3$	P_3	P_6	P_7

L'erreur associée à cette solution est la distance entre les points P_1 et P_3, soit 75.

Formulation proposée

On fait ici l'hypothèse que $N = n_1 n_2$.

Soit D_{min} (resp. D_{max}) la plus petite (resp. plus grande) valeur dans la matrice des distances, et soient $D_{min} = D^0 < D^1 < D^2 < ... < D^K = D_{max}$ les différentes valeurs des distances comprises entre D_{min} et D_{max}.
On définit les variables de décision suivantes :

- $z^k = 0$ s'il est possible de trouver une solution avec une erreur au plus égale à D^{k-1}, $z^k = 1$ sinon,
- $r_{ij} = 1$ si les points P_i et P_j ont le même codage a et 0 sinon,
- $c_{ij} = 1$ si les points P_i et P_j ont le même codage b et 0 sinon.

Bien sûr, les variables r_{ij} et c_{ij} ne donnent qu'une description partielle de la solution mais on peut aisément montrer qu'à partir d'une valeur admissible de ces variables, il est possible de construire une solution avec les valeurs $a(i)$ et $b(i)$.

La formulation que nous proposons est la suivante :

$$\min \quad D^0 + \sum_{k=1}^{K} (D^k - D^{k-1}) z^k \tag{2.39}$$

sous les contraintes

$$r_{ij} + r_{jl} - r_{il} \leq 1 \qquad 1 \leq i \neq j \neq l \leq N \tag{2.40}$$

$$c_{ij} + c_{jl} - c_{il} \leq 1 \qquad 1 \leq i \neq j \neq l \leq N \tag{2.41}$$

$$\sum_{j=1}^{N} r_{ij} = n_2 \qquad i = 1,...,N \tag{2.42}$$

$$\sum_{i=1}^{N} c_{ij} = n_1 \qquad j = 1,...,N \tag{2.43}$$

$$r_{ij} + c_{ij} \leq z^k \qquad k = 1,...,K;\ 1 \leq i \neq j \leq N : d_{ij} \geq D^k \tag{2.44}$$

$$r_{ii} = 1 \text{ et } c_{ii} = 1 \qquad i = 1,...,N \tag{2.45}$$

$$r_{ij} = r_{ji} \text{ et } c_{ij} = c_{ji} \qquad i = 1,...,N-1; j = i+1,...,N \tag{2.46}$$

$$z^k \in \{0,1\},\ r_{ij} \in \{0,1\} \text{ et } c_{ij} \in \{0,1\} \tag{2.47}$$

Les contraintes (2.40) sont des contraintes de transitivité : si les points P_i et P_j ont le même codage a et que les points P_j et P_l ont le même codage a alors les points P_i et P_l ont nécessairement le même codage a. Les contraintes (2.41) sont les contraintes analogues pour le codage b. Les contraintes (2.42) (resp. (2.43)) imposent que chaque point i partage le même codage a (resp. b) que $n_2 - 1$ (resp. $n_1 - 1$) autres points, à cause de l'hypothèse $N = n_1 n_2$. Les contraintes (2.44) font que si $z^k = 0$, c'est à dire qu'il existe une solution avec une erreur d'au plus D^{k-1}, alors toutes les paires de points (P_i, P_j) telles que $d_{ij} \geq D^k$ n'ont ni le même codage a ni le même codage b. Par contre, si $z^k = 1$ alors toute paire de points peut avoir le même codage a ou le même codage b mais pas les deux.

Approche de résolution On peut ramener la recherche d'un codage minimisant l'erreur à celui de la recherche de la plus petite distance telle que le sous-problème de décision F_δ suivant, associé à une distance δ admette une solution :

Problème F_δ :

$$\begin{aligned}
r_{ij} + r_{jl} - r_{il} \leq 1 & \qquad 1 \leq i \neq j \neq l \leq N \\
c_{ij} + c_{jl} - c_{il} \leq 1 & \qquad 1 \leq i \neq j \neq l \leq N \\
\sum_{j=1}^{N} r_{ij} = n_2 & \qquad i = 1,\ldots,N \\
\sum_{i=1}^{N} c_{ij} = n_1 & \qquad j = 1,\ldots,N \\
r_{ij} = c_{ij} = 0 & \qquad 1 \leq i \neq j \leq N : d_{ij} > \delta \\
r_{ij} + c_{ij} \leq 1 & \qquad 1 \leq i \neq j \leq N : d_{ij} \leq \delta \\
r_{ii} = 1 \text{ et } c_{ii} = 1 & \qquad i = 1,\ldots,N \\
r_{ij} = r_{ji} \text{ et } c_{ij} = c_{ji} & \qquad i = 1,\ldots,N-1; j = i+1,\ldots,N \\
r_{ij} \in \{0,1\} \text{ et } c_{ij} \in \{0,1\} &
\end{aligned}$$

Et, de la même façon que pour le problème du p-Centre, on peut calculer une borne inférieure définie comme la plus petite distance telle que la relaxation continue LF_δ du problème F_δ soit admissible, puis continuer la dichotomie en résolvant les sous-problèmes F_δ en 0-1. Cette méthode est en cours de test.

2.7 Conclusion

Dans ce chapitre, nous avons présenté une nouvelle formulation du problème du p-Centre, ainsi qu'une méthode très efficace de calcul d'une borne inférieure de la valeur optimale de ce problème, que nous avons appelée LB^*. Nous avons mené un travail expérimental qui montre la très bonne qualité de cette borne inférieure, tout au moins pour les instances Euclidiennes, les seules envisagées par plusieurs autres auteurs. Notre comparaison entre la formulation classique et la nouvelle formulation indique clairement la supériorité de la nouvelle formulation. Pour ce qui est du calcul de la borne inférieure et de la résolution exacte, même si nous résolvons des instances de taille 2 fois plus importante que celles jusqu'ici reportées dans la littérature, des voies d'amélioration nous semblent se dégager immédiatement de notre travail :

- La dégradation de la qualité de la borne LB^* pour les instances non-Euclidienne indique qu'il faut trouver des idées d'amélioration de cette borne, par exemple par l'introduction de coupes, connues ou à trouver.
- Notre méthode de résolution exacte s'appuie sur la résolution d'un certain nombre de problèmes de recouvrement, que nous effectuons à l'aide d'un solveur de programmes linéaires en nombres entiers, avec assez peu d'efforts supplémentaires pour tenir compte des particularités du problème de recouvrement. Ce problème a fait d'objet de développements récents et importants dans la littérature, dont il serait intéressant de profiter pour mieux résoudre le problème du p-Centre.
- Nous avons fait le choix d'une méthode de résolution exacte parmi d'autres possibles. Un autre choix pourrait être la conception d'une méthode de *Branch & Bound* utilisant la borne LB^* dans sa phase d'évaluation. Pour les instances Euclidiennes où la borne LB^* est excellente, il n'est pas exclu d'obtenir de meilleures performances en adoptant cette dernière méthode.

Chapitre 3

Un problème de Localisation et Routage dans la conception d'un réseau de télécommunications

3.1 Introduction

Un système de radiotéléphonie ([49], Chapitres 1 et 3) a pour premier objectif de permettre l'accès au réseau téléphonique à partir d'un terminal portatif (notre téléphone portable) sur un territoire étendu. Ce service utilise une liaison radioélectrique entre le terminal et le réseau. Pour que le service soit disponible, il faut que la liaison soit de qualité suffisante, ce qui peut nécessiter une puissance importante des émetteurs. Afin de limiter cette puissance, l'opérateur du réseau radiomobile place un ensemble de *stations de base* sur le territoire à couvrir pour que le terminal soit toujours à moins de quelques kilomètres de l'une d'entre elles. Un réseau de radiotéléphonie est composé de plusieurs sous-ensembles, dont un nous intéresse en particulier: le *sous-système radio* qui assure les transmissions radioélectriques et gère la ressource radio. C'est ce sous-système qui joue le rôle d'intermédiaire entre un terminal et le reste du réseau de radiotéléphonie, c'est à dire le sous-système d'acheminement et le sous-système d'exploitation et de maintenance. Le sous-système radio est composé:

- des stations de base, appelées BTS (*Base Transceiver Station*), qui sont des émetteurs-récepteurs ayant un minimum d'intelligence
- des contrôleurs de stations de base BSC (*Base Station Controller*) dont chacun contrôle un nombre limité de BTS et permet une première concentration des circuits.

La connexion BTS-BSC est assurée par des liaisons numériques, suivant une configuration chaînée ou en étoile dans les anciens réseaux de radiotéléphonie, et suivant une configuration en boucles dans les réseaux de nouvelle génération, basés sur la technologie SDH (*Synchronous Digital Hierarchy*).

Le problème que nous étudions ici concerne la définition d'architecture des sous-systèmes radio, dans la norme SDH. Connaissant les sites des BTS et un certain nombre de sites potentiels pour les BSC, il faut choisir les sites où vont être effectivement placés des BSC et décrire les boucles SDH gérées par chaque BSC. Une boucle SDH étant un sous-ensemble de BTS reliés entre-eux et à un BSC par des liaisons qui forment une boucle. Mais le problème est plus complexe car il faut aussi dimensionner les BSC, c'est à dire décider de leur capacité et donc choisir un équipement adéquat parmi des types de BSC disponibles. Il faut en outre tenir compte de plusieurs restrictions techniques. Enfin, on doit essayer de trouver une architecture de coût minimal. Une définition plus formelle du problème est donnée dans la Section 3.3. Pour donner un ordre de grandeur, on peut dire que le réseau de Paris intra-muros contenait, fin 1994, de l'ordre d'une centaine de BTS et quelques dizaines de BSC [49].

Le problème posé, que nous avons appelé *SDHNDP*, fait partie d'une classe de problèmes connue dans la littérature de l'Optimisation Combinatoire sous le nom de *problèmes de localisation et de routage*. En effet, dans le problème SDHNDP, localiser des équipements et leur allouer des BTS représente un sous-problème de *localisation discrète* et, définir les boucles SDH en supposant connus les emplacements des équipements représente un sous-problème de *routage*, plus précisément un problème de *tournées de véhicules multi-dépôts*. Le SDHNDP contient un sous-problème supplémentaire qui est le dimensionnement de la capacité de chaque équipement à installer. Chacun de ces sous-problèmes est difficile et le problème SDHNDP consiste à les considérer tous à la fois.

Dans la suite de ce chapitre, nous donnons d'abord, dans la Section 3.2 une définition précise du problème de Localisation et Routage, comme il a été défini par Perl et Daskin [56], puis nous consacrons la section suivante à une description formelle du problème (SDHNDP) et de nos approches de calcul d'une solution réalisable et d'une borne inférieure.

3.2 Un problème général de Localisation et Routage (LRP)

Il existe une grande variété de problèmes pratiques qui font intervenir à la fois des décisions de localisation et de routage. Une manière générale de décrire ces problèmes consiste à dire qu'il s'agit d'ouvrir des dépôts parmi un ensemble de sites potentiels et d'organiser des livraisons chez les clients par l'intermédiaire de véhicules qui effectuent des tournées. Dans la littérature traitant d'Optimisation Combinatoire, ces problèmes sont au carrefour de deux classes importantes de problèmes désignés respectivement par *les problèmes de localisation* et *les problèmes de tournées de véhicules*, eux-mêmes très largement étudiés. A la fin des années 80, Laporte [50] a été le premier à effectuer un travail de synthèse sur les problèmes de localisation et routage où il a fait l'inventaire des exemples et applications, des formulations mathématiques et des méthodes de résolution exacte ou approchée. Plus récemment, Min et al. [53] ont de plus essayé d'établir une taxonomie des différents modèles existants et de proposer des directions de recherche future pour cette classe de problèmes. En effet, plusieurs modèles existent et peuvent influencer le choix des méthodes qu'on peut être amené à choisir pour les traiter. Les données peuvent être déterministes ou incertaines. Il peut exister plusieurs niveaux de localisation: une source puis des dépôts de catégorie différente. La nature des contraintes et des coûts est différente suivant les modèles: nombre maximal ou minimal de dépôts à ouvrir, véhicules homogènes ou différents, nombre maximal de véhicules, un client peut être ou non desservi par des tournées différentes, existence ou non de coûts fixes par tournée, ...

Nous présentons ici le problème de localisation et routage initialement décrit par Perl et Daskin [56]. C'est le problème de la littérature qui, à notre connaissance, ressemble le plus au problème de conception de réseau radiomobile que nous traitons. Nous le désignons par (LRP). Dans ce problème, on dispose des données suivantes:

- N: nombre de sites clients, désignés par $S_1, \ldots, S_N$;

- M: nombre de sites de dépôts potentiels, désignés par $S_{N+1},\dots,S_{N+M}$;
- K: nombre de véhicules;
- C: capacité de chaque véhicule;
- T_j: disponibilité au dépôt S_j, s'il est ouvert $(N+1 \leq j \leq N+M)$;
- q_i: demande du client S_i $(1 \leq i \leq N)$;
- CM: coût de transport par unité de distance parcourue;
- d_{gh}: distance entre les sites S_g et S_h $(1 \leq g,h \leq N+M)$;
- FC_j: coût fixe d'ouverture d'un site en S_j $(N+1 \leq j \leq N+M)$;
- VC_j: coût de livraison d'une unité de marchandise depuis le site S_j $(N+1 \leq j \leq N+M)$.

Le problème est de trouver le nombre et le lieu des dépôts à ouvrir et à définir le trajet de chaque véhicule de sorte que:
- chaque véhicule parte d'un dépôt ouvert et y retourne, sans passer par un autre site de dépôt,
- les demandes des clients, les capacités des véhicules et les disponibilités aux dépôts soient respectées, un client étant desservi par exactement un véhicule,
- la somme des coûts d'ouverture des dépôts, de transport et de livraison soit minimale.

Ce problème a fait l'objet de quelques publications, dont celle de Hansen et al. [42] qui présentent une heuristique constructive et une autre récente de Tuzun et Burke [61] qui proposent une méthode de recherche tabou.

3.3 Le problème de conception d'un réseau radiomobile

3.3.1 Description du problème $(SDHNDP)$

On dispose des donnée suivantes:
- N: nombre de BTS, désignés par $BTS_1,\dots,BTS_N$. Soit $\mathcal{N} = \{1,\dots,N\}$ l'ensemble des indices des sites de BTS;
- M: nombre de sites potentiels de BSC, désignés par $BSC_{N+1},\dots,BSC_{N+M}$. Soit $\mathcal{M} = \{N+1,\dots,N+M\}$ l'ensemble des indices des sites de BSC;
- NS: nombre maximal d'équipements BSC par site, chaque site potentiel de BSC est ainsi divisé en NS sous-sites;
- NT: nombre de types d'équipements BSC disponibles;
- d_{gh}: $(g,h \in \mathcal{N} \cup \mathcal{M})$ distances entre les sites (BTS ou BSC) g et h. Ces distances vérifient l'inégalité triangulaire;
- q_i: $(i \in \mathcal{N})$ vecteur des demandes de BTS_i. Ce vecteur a 4 composantes $q_i = (IT_i,Erlang_i,BTS_i,TRX_i)$, qui désignent respectivement le volume de trafic, l'intensité du trafic, le nombre d'antennes et le nombre de terminaux qu'il peut gérer;
- Q_ω: $(\omega = 1,\dots,NT)$ vecteur des capacités d'un équipement BSC de type $TBSC_\omega$. De façon analogue au vecteur q_i, le vecteur Q_ω a 4 composantes $Q_\omega = (IT_\omega,Erlang_\omega,BTS_\omega,TRX_\omega)$;
- $\bar{n}$: nombre maximal de BTS dans une même boucle SDH ($\bar{n} = 14$);
- $\underline{n}$: nombre minimal de BTS dans une boucle SDH ($\underline{n} = 2$);
- $\bar{D}$: longueur maximale d'une boucle SDH ($\bar{D} = 100000$ m);

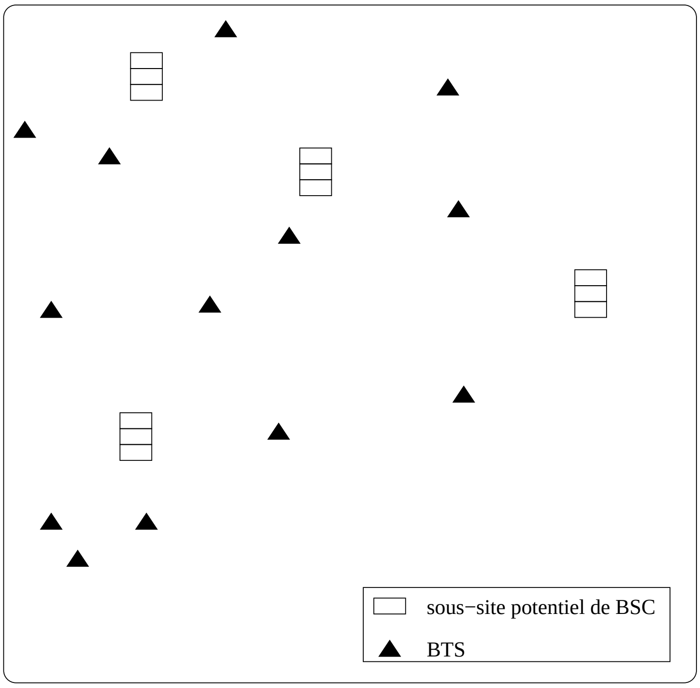


FIG. 3.1 – *Un exemple de SDHNDP*

- F_j: ($j \in \mathcal{M}$) coût d'ouverture d'un site BSC_j. Un site est ouvert dès qu'un de ses sous-sites est équipé;
- M_ω: ($\omega = 1,\ldots,NT$) coût d'acquisition d'un équipement BSC de type ω;
- CU: coût par kilomètre de longueur de boucle;
- CF: coût fixe par boucle.

Il est facile de voir les similitudes qui existent entre le problème ($SDHNDP$) et le problème (LRP) que nous décrivons dans la Section 3.2. L'équipement ou non d'un sous-site de BSC correspond à l'ouverture ou non d'un dépôt et une boucle SDH correspond au trajet d'un véhicule. Le problème ($SDHNDP$) est cependant différent par un certain nombre de points:

- Il n'existe pas de limite sur le nombre total de boucles.
- En plus de la décision d'équipement ou non d'un sous-site, il faut aussi dire par quel type parmi les NT possibles il faut le faire, et cela donne la capacité et le coût d'achat de cet équipement.
- La longueur et le nombre de BTS d'une boucle sont limités.

La Figure 3.1 illustre un exemple de problème SDHNDP avec 13 BTS, 4 sites potentiels de BSC contenant chacun 3 sous-sites. La Figure 3.2 illustre une solution pour cet exemple, où 3 sites sont ouverts, 4 équipements BSC de type 1 ou 2 sont installés et 5 boucles SDH relient les BTS aux équipements BSC.

3.3.2 Modélisation du problème

Soit $G = (V,A)$ un graphe simple, orienté et symétrique. L'ensemble des nœuds $V = \mathcal{N} \cup \mathcal{M}$ est composé des indices des sites de BTS et des sites potentiels de BSC. L'ensemble des arcs A

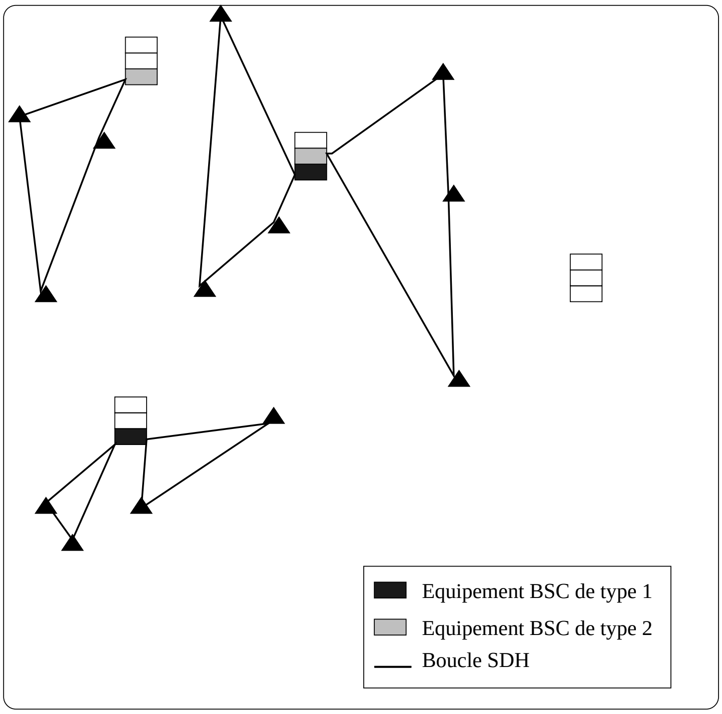


FIG. 3.2 – *Une solution avec 3 sites ouverts et 5 boucles SDH*
pour l'exemple de la Figure 3.1

est obtenu à partir du graphe complet sur V en supprimant:

– les arcs entre deux sites de BSC,
– les arcs (g,h) vérifiant $d_{gh} + \min_{j \in \mathcal{M}}(d_{jg} + d_{hj}) > \bar{D}$. En effet, ces arcs sont trop longs pour être inclus dans une boucle de longueur inférieure à $\bar{D}$. Notons que cette condition élimine aussi les arcs (j,i) reliant BSC_j à BTS_i tels que $d_{ji} > \frac{\bar{D}}{2}$.

Ayant défini le graphe G, nous pouvons maintenant dire qu'une boucle SDH correspond à un circuit élémentaire de ce graphe, passant par un et un seul site de BSC et respectant les contraintes de cardinalité, de longueur et de capacité des boucles SDH.

Nous considérons les variables suivantes:

- $v_{gh} = 1$ si l'arc (g,h) est sélectionné (i.e. inclus dans une boucle) et 0 sinon, pour $(g,h) \in A$.
- $u_{js\omega} = 1$ si le sous-site s du site BSC_j contient un équipement BSC de type ω et 0 sinon, pour $j \in \mathcal{M}$, $s = 1,\ldots,NS$ et $\omega = 1,\ldots,NT$
- $z_j = 1$ si le site BSC_j est ouvert (i.e. au moins un de ses sous-sites contient un équipement BSC) et 0 sinon, pour $j \in \mathcal{M}$.
- $t_{ijs} = 1$ si BTS_i est desservie par le sous-site s de BSC_j et 0 sinon, pour $i \in \mathcal{N}$, $j \in \mathcal{M}$ et $s = 1,\ldots,NS$.
- l_{gh} = longueur de la portion de boucle (chemin) depuis le site BSC qu'elle contient, jusqu'au nœud g, si l'arc (g,h) est sélectionné et 0 sinon, pour $(g,h) \in A$.
- n_i = nombre de sites de BTS visités avant BTS_i, dans la boucle qui contient BTS_i, pour $i \in \mathcal{N}$.

Notons que les variables de décision v_{gh} et $u_{js\omega}$ déterminent complètement une solution de SDHNDP. Mais elles ne sont pas suffisantes pour le formuler par un programme linéaire. Les autres variables sont donc introduites afin d'obtenir la formulation suivante qui est un programme linéaire en nombres entiers, contenant un nombre polynomial de variables et contraintes:

$$\textbf{(SDHNDP):}\quad \text{Min} \sum_{j\in\mathcal{M}} F_j z_j + \sum_{j\in\mathcal{M}}\sum_{s=1}^{NS}\sum_{\omega=1}^{NT} M_\omega u_{js\omega} + CU \sum_{(g,h)\in A} d_{gh} v_{gh} + CF \sum_{j\in\mathcal{M}}\sum_{i\in\mathcal{N}} v_{ji} \quad (3.1)$$

sous les contraintes:

$$\sum_{h:(h,i)\in A} v_{hi} = 1 \qquad i\in\mathcal{N} \quad (3.2)$$

$$\sum_{h:(i,h)\in A} v_{ih} = 1 \qquad i\in\mathcal{N} \quad (3.3)$$

$$\sum_{s=1}^{NS} t_{ijs} \geq (v_{ji} + v_{ij}) \qquad i\in\mathcal{N};\ j\in\mathcal{M} \quad (3.4)$$

$$1 + t_{i_2js} \geq t_{i_1js} + (v_{i_1i_2} + v_{i_2i_1}) \qquad i_1,i_2\in\mathcal{N} : i_1 \neq i_2;\ j\in\mathcal{M};\ s=1,\dots,NS \quad (3.5)$$

$$\sum_{j\in\mathcal{M}}\sum_{s=1}^{NS} t_{ijs} = 1 \qquad i\in\mathcal{N} \quad (3.6)$$

$$\sum_{i\in\mathcal{N}} q_i t_{ijs} \leq \sum_{\omega=1}^{NT} Q_\omega u_{js\omega} \qquad j\in\mathcal{M};\ s=1,\dots,NS \quad (3.7)$$

$$n_i \leq (\bar{n}-1)(1 - \sum_{j\in\mathcal{M}} v_{ji}) \qquad i\in\mathcal{N} \quad (3.8)$$

$$n_{i_1} - n_{i_2} + 1 \leq \bar{n}(1 - v_{i_1i_2}) \qquad i_1,i_2\in\mathcal{N} : i_1\neq i_2 \quad (3.9)$$

$$l_{ji} = 0 \qquad j\in\mathcal{M};\ i\in\mathcal{N} \quad (3.10)$$

$$l_{ih} \leq (\bar{D} - d_{ih} - \min_{j\in\mathcal{M}} d_{hj}) v_{ih} \qquad i\in\mathcal{N};\ h\in V (i\neq h) \quad (3.11)$$

$$\sum_{h\in V} (l_{hi} + d_{hi} v_{hi}) = \sum_{h\in V} l_{ih} \qquad i\in\mathcal{N} \quad (3.12)$$

$$\sum_{\omega=1}^{NT} u_{js\omega} \leq z_j \qquad j\in\mathcal{M};\ s=1,\dots,NS \quad (3.13)$$

$$\sum_{\omega=1}^{NT} u_{js\omega} \geq t_{ijs} \qquad i\in\mathcal{N};\ j\in\mathcal{M};\ s=1,\dots,NS \quad (3.14)$$

$$\sum_{j\in\mathcal{M}, i\in\mathcal{N}} v_{ji} \geq \beta \quad (3.15)$$

$$v_{gh}\in\{0,1\} \quad l_{gh}\geq 0 \qquad (g,h)\in A \quad (3.16)$$

$$u_{js\omega}\in\{0,1\} \qquad j\in\mathcal{M};\ s=1,\dots,NS;\ \omega=1,\dots,NT \quad (3.17)$$

$$n_i \geq 0 \quad 0\leq z_j\leq 1 \qquad i\in\mathcal{N};\ j\in\mathcal{M} \quad (3.18)$$

$$t_{ijs}\in\{0,1\} \qquad i\in\mathcal{N};\ j\in\mathcal{M};\ s=1,\dots,NS \quad (3.19)$$

L'objectif (3.1) est de minimiser la somme des coûts d'ouverture des sites BSC, d'acquisition des équipements BSC, d'installation des boucles (coût dû à la longueur totale des boucles plus coût fixe par boucle).

Les contraintes (3.2) et (3.3) imposent que chaque BTS soit desservie par exactement une boucle. Les contraintes (3.4) veulent dire que si BTS_i est liée à BSC_j par un arc sélectionné, alors BTS_i est desservie par un des sous-sites de BSC_j. Les contraintes (3.5) obligent BTS_{i_1} et BTS_{i_2} à être desservies par le même sous-site si l'arc (i_1,i_2) est sélectionné. Les contraintes (3.6) veulent dire que BTS_i est desservie par exactement un sous-site. Les contraintes (3.4), (3.5) et (3.6) obligent une boucle à commencer et à se terminer au même sous-site et garantissent toutes les BTS d'une boucle d'être desservies par le même sous-site. Elles éliminent aussi les boucles ayant un seul site BTS puisque les contraintes (3.4) et (3.6) impliquent $v_{ij} + v_{ji} \leq 1$. Les contraintes (3.7) sont des contraintes de capacité.

Les variables n_i représentent une numérotation des BTS à l'intérieur d'une boucle. Rappelons qu'une boucle part toujours d'un sous-site de BSC. La première BTS visitée ensuite est numérotée 0. Ceci est assuré par les contraintes (3.8). Les BTS suivantes dans la même boucle sont numérotées 1,2,...,etc. Pour ce faire, les contraintes (3.9) garantissent que si l'arc (i_1,i_2) est sélectionné alors $n_{i_2} \geq n_{i_1} + 1$. Les contraintes (3.8) interdisent aussi que le numéro d'une BTS soit plus grand que $(\bar{n} - 1)$. Ainsi, dans une solution admissible, une boucle ne contient pas plus de $\bar{n}$ BTS. Par ailleurs, les variables n_i servent aussi à éliminer les solutions contenant des sous-tours de sites BTS, de la même façon que la formulation de Miller, Tucker et Zemlin [52] pour le problème du voyageur du commerce.

Les contraintes (3.10), (3.11) et (3.12) concernent les variables l_{gh} et servent à respecter les contraintes de longueur des boucles. Rappelons que, si l'arc (g,h) est sélectionné, alors l_{gh} est la longueur de la portion de boucle depuis le site BSC de la boucle jusqu'au nœud g. Ainsi, $l_{gh} = 0$ si g correspond à un site de BSC ou si l'arc (g,h) n'est pas sélectionné. Ceci est imposé par les contraintes (3.10) et (3.11). De plus, les contraintes (3.12) donnent la relation entre ces variables, pour deux arcs consécutifs, sélectionnés. En effet, les contraintes (3.2) et (3.3) font qu'exactement un arc sélectionné (h_1,i) arrive au nœud i et un arc sélectionné (i,h_2) part du nœud i. Ensuite, les contraintes (3.12) font que $l_{ih_2} = l_{h_1 i} + d_{h_1 i}$. Finalement, la boucle passant par l'arc (i,h) a une longueur qui ne dépasse pas $l_{ih} + d_{ih} + \min_{j \in \mathcal{M}} d_{hj}$. Ainsi, les contraintes (3.11), avec les autres, représentent des conditions nécessaires et suffisantes pour que la longueur totale d'une boucle ne dépasse pas $\bar{D}$. Les contraintes (3.13) imposent, d'une part, qu'un site BSC soit ouvert si un de ses sous-sites est équipé et, d'autre part, que les sous-sites ne contiennent qu'un équipement au maximum.

Les contraintes (3.14) et (3.15) sont des contraintes redondantes, dont l'introduction vise à améliorer la relaxation continue du problèmes $(SDHNDP)$. Les contraintes (3.14) disent que si BTS_i est desservie par le sous-site s de BSC_j alors ce sous-site est équipé.

Pour un choix judicieux du paramètre β, les contraintes (3.15) sont également des contraintes redondantes qui imposent qu'une solution admissible contienne au moins β boucles. Il faut donc que β soit une borne inférieure sur le nombre de boucles dans toute solution admissible. A cause de la restriction sur le nombre de BTS dans une boucle, une valeur triviale de β est $\lceil \frac{N}{\bar{n}} \rceil$. Nous montrons ci-après comment calculer une meilleure valeur de β dans le but d'améliorer la résolution du problème SDHNDP.

3.3.3 Amélioration de la formulation

Le but est ici de calculer une borne inférieure sur le nombre de boucles dans une solution admissible de $(SDHNDP)$. Soit $\Delta = (V,E)$ le graphe non-orienté obtenu à partir du graphe

G. Etant donné un entier K, nous désignons par P_K le problème de partitionnement du graphe Δ en K sous-ensembles sous les contraintes que nous préciserons. Nous définissons β comme la plus petite valeur de K telle que le problème P_K admette une solution.

Le problème P_K consiste à partitionner l'ensemble des sommets de V en K sous-ensembles et sélectionner des arêtes de E de telle sorte que:

- chaque BTS appartienne à exactement un sous-ensemble
- le sous-graphe de Δ induit par chaque sous-ensemble soit une clique
- le nombre de BTS dans un sous-ensemble est dans $[\underline{n},\bar{n}]$
- chaque sous-ensemble contienne exactement un site de BSC
- chaque sommet d'un sous-ensemble soit relié à deux autres sommets du même sous-ensemble, par deux arêtes sélectionnées
- la somme des longueurs des arêtes sélectionnées à l'intérieur d'un même sous-ensemble n'excède par $\bar{D}$
- la somme des longueurs des arêtes sélectionnées soit minimale.

Le problème P_K peut être vu comme une relaxation de SDHNDP. A partir d'une solution de SDHNDP contenant ν boucles, on peut facilement déduire une solution de P_ν par correspondance un à un entre les boucles de SDHNDP et les sous-ensembles de P_ν. En d'autres termes, si P_K n'a pas de solution admissible pour un K donné alors $(K+1)$ est une borne inférieure du nombre de boucles pour toute solution admissible de SDHNDP. Par ailleurs, il est clair que $\beta \geq \lceil \frac{N}{\bar{n}} \rceil$.

Par ailleurs, soit Δ' le sous-graphe de Δ engendré par l'ensemble des BTS $\mathcal{N}$. Si $\mathcal{S} \subset \mathcal{N}$ est un ensemble stable du graphe Δ', alors les sommets de $\mathcal{S}$ appartiennent à des sous-ensembles différents dans toute partition de Δ. Ainsi, $\beta \geq \alpha(\Delta')$, où $\alpha(\Delta')$ est le nombre de stabilité de Δ'.

Nous utilisons l'algorithme suivant pour calculer β :

Pas 1. **Init.** $K = max(\lceil \frac{N}{\bar{n}} \rceil, \alpha(\Delta'))$;

Pas 2. Résoudre P_K;

Pas 3. si (P_K admet une solution) alors { $\beta = K$; **Arrêt.** } sinon { K=K+1; Retour à Pas 2.}

Au Pas 1., on a besoin de calculer $\alpha(\Delta')$. Pour ce faire, nous utilisons une formulation classique du problème du stable de cardinalité maximale par un programme linéaire en 0-1. Ensuite, nous faisons appel à un solveur pour calculer un ensemble stable $\mathcal{S}$ tel que $|\mathcal{S}| = \alpha(\Delta')$. Au Pas 2., on doit résoudre le problème P_K . Nous le formulons comme un programme linéaire en 0-1. Pour améliorer notre formulation, nous utilisons l'ensemble stable $\mathcal{S}$ trouvé au Pas. 1. Soit $\mathcal{S} = \{b_1, b_2, \ldots, b_{|\mathcal{S}|}\}$. Nous formulons le problème P_K comme suit:

Les variables de décision sont $x_{gk} = 1$ ssi le sommet g est dans le sous-ensemble k, et $y_{ghk} = 1$ ssi l'arête (g,h) est sélectionnée dans le sous-ensemble k.

$$(\mathbf{P_k}) : \quad \text{Min} \quad \sum_{k\in\{1,\ldots,K\}} \sum_{(g,h)\in E(g<h)} d_{gh} y_{ghk} \tag{3.20}$$

sous les contraintes:

$$\sum_{k\in\{1,\ldots,K\}} x_{ik} = 1 \qquad i \in I \tag{3.21}$$

$$\sum_{j\in J} x_{jk} = 1 \qquad k \in \{1,\ldots,K\} \tag{3.22}$$

$$\sum_{i\in I} x_{ik} \leq \bar{n} \qquad k \in \{1,\ldots,K\} \tag{3.23}$$

$$\sum_{i\in I} x_{ik} \geq \underline{n} \qquad k \in \{1,\ldots,K\} \tag{3.24}$$

$$x_{gk} + x_{hk} \leq 1 \qquad (g,h) \notin E, k \in \{1,\ldots,K\} \tag{3.25}$$

$$\sum_{h\in V(g<h)} y_{ghk} + \sum_{h\in V(h<g)} y_{hgk} = 2x_{gk} \qquad g \in V, k \in \{1,\ldots,K\} \tag{3.26}$$

$$\sum_{g\in V, h\in V(g<h)} d_{gh} y_{ghk} \leq D_{max} \qquad k \in \{1,\ldots,K\} \tag{3.27}$$

$$\sum_{k\in\{1,\ldots,K\}} y_{ghk} \leq 1 \qquad (g,h) \in E \ (g<h) \tag{3.28}$$

$$x_{b_k k} = 1 \qquad k \in \{1,\ldots,|\mathcal{S}|\} \tag{3.29}$$

$$y_{ghk} \in \{0,1\} \qquad (g,h) \in E, k \in \{1,\ldots,K\} \tag{3.30}$$

$$_{gk} \in \{0,1\} \qquad g \in V, k \in \{1,\ldots,K\} \tag{3.31}$$

$$\tag{3.32}$$

Les contraintes (3.21) affectent chaque BTS à exactement un sous-ensemble. Les contraintes (3.22) imposent l'appartenance d'un site de BSC à chaque sous-ensemble. Les contraintes (3.23) et (3.24) limitent la cardinalité de chaque sous-ensemble. Les contraintes (3.25) évitent l'affectation de deux sommets non adjacents dans E à un même sous-ensemble. Les contraintes (3.26) sont des contraintes de degré. La longueur totale des arcs sélectionnés dans un même sous-ensemble est limitée à D_{max} par les contraintes (3.27)). Les contraintes (3.28) sont des contraintes supplémentaires que nous ajoutons pour améliorer la relaxation linéaire de P_K. Les contraintes (3.29) affectent le sommet BTS_{b_k} de l'ensemble $\mathcal{S}$ au sous-ensemble d'indice k. Ce qui garantit l'appartenance des sommets de $\mathcal{S}$ à des sous-ensembles différents et, puisque toute numérotation des sous-ensembles est valide, la fixation de ces variables n'élimine pas des solutions admissibles de P_K. Finalement, la fonction objectif (3.20) minimise la longueur totale des arcs sélectionnés.

3.3.4 Heuristique proposée

La méthode heuristique que nous avons développée [36] est constituée de trois phases.

Phase 1:

Le but de cette phase est de relier chaque BTS à un site de BSC par l'intermédiaire d'un arbre. Les arbres ainsi obtenus doivent être des sous-graphes partiels du graphe Δ, dont chacun relie un ensemble de sites de BTS et un seul site de BSC. Nous appelons le sous-graphe partiel de G obtenu une *forêt couvrante des BTS*. De plus, parmi toutes les forêts possibles, on en cherche une qui minimise la somme des longueurs des arêtes qu'elle utilise, soit une forêt *de poids minimal*.

Pour calculer une forêt couvrante des BTS, de poids minimal, nous utilisons l'algorithme suivant: (1) ajouter au graphe Δ des arêtes de poids 0 pour relier toutes les paires de sites de BSC. On obtient un graphe Δ'. (2) appliquer l'algorithme de Prim [57] pour trouver un arbre couvrant de poids minimal sur Δ'. On obtient un arbre $\mathcal{A}'$. (3) Enlever de $\mathcal{A}'$ les arêtes qui relient deux sites de BSC. On obtient $\mathcal{F}$ une forêt couvrante des BTS, dont on peut aisément montrer qu'elle est de poids minimal. On peut aussi montrer que le poids de cette forêt est une borne inférieure de la longueur totale des boucles, dans une solution admissible quelconque de $(SDHNDP)$.

Mais l'algorithme que nous utilisons effectivement modifie l'étape (2) de l'algorithme ci-dessus pour essayer, dans une démarche heuristique, de mieux tenir compte des contraintes de $(SDHNDP)$:

- La demande de l'ensemble des BTS reliées à un site de BSC par un arbre ne doit pas dépasser la disponibilité maximale d'un site de BSC, c'est à dire celle qu'on obtient en équipant tous ses sous-sites par un type d'équipement de capacité maximale.
- On ne relie jamais une BTS au même arbre qu'un BSC qui est à une distance supérieure à $\alpha\frac{\bar{D}}{2}$ où $0 < \alpha \leq 1$ est un paramètre fixé par l'heuristique. Cela permet d'éviter d'obtenir un arbre où le nombre de BTS dans les différents arbres est déséquilibré.

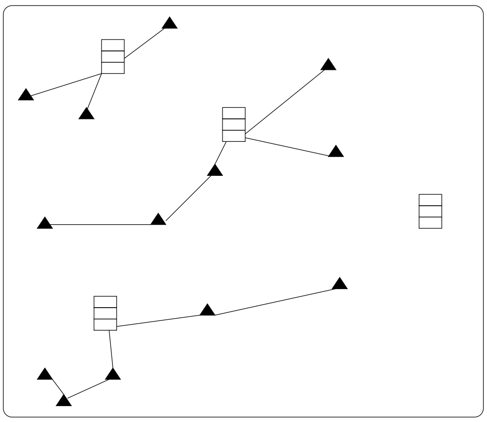


FIG. 3.3 – *Forêt obtenue à la fin de la Phase 1 de l'heuristique pour l'exemple de la Figure 3.1*

Phase 2:

La forêt couvrante des BTS, obtenue à la fin de la Phase 1 définit une affectation de chaque BTS à un BSC. En effet, chaque BTS appartient à un arbre et chaque arbre contient exactement un BSC. Cette phase consiste d'abord à décider d'ouvrir chaque site de BSC qui fait partie d'un arbre et à "fermer" les autres sites de BSC. Ensuite, on considère les arbres un par un et on forme des boucles SDH qui relient les BTS aux BSC en résolvant un problème de tournées de véhicule avec un seul dépôt et sous les contraintes qu'une boucle ne contient pas plus de $\bar{n}$ BTS et que sa longueur ne dépasse pas $\bar{D}$. La fonction objectif de ce problème vise à minimiser la somme des coûts dus au nombre de boucles et à leur longueur totale. Chacun des sous-problèmes est résolu de façon heuristique, par une adaptation de l'heuristique de Clarke et Wright [20].

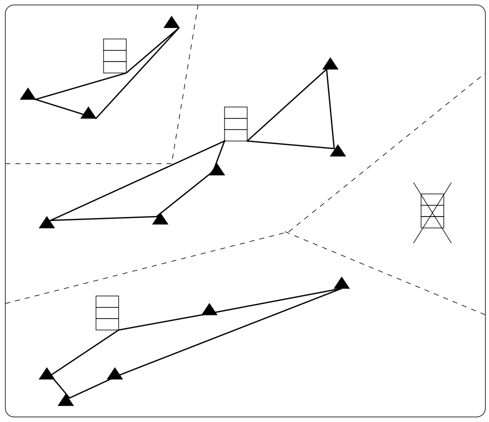


FIG. 3.4 – *Boucles SDH obtenues à la fin de la Phase 2 de l'heuristique pour l'exemple de la Figure 3.1*

Phase 3:

A la fin de la phase précédente, on a les boucles SDH et leurs sites BSC de rattachement. Il reste à décider du dimensionnement de chaque site BSC "ouvert". Cela revient à dire pour chaque sous-site s'il est équipé ou non, et s'il est équipé, par quel type d'équipement. Il faut aussi affecter les boucles à des sous-sites équipés. Il s'agit en fait, pour chaque site de BSC et les boucles qui lui sont affectées, d'un problème de localisation simple, avec capacités. Nous résolvons ces problèmes à l'aide d'un solveur de programmes linéaires en nombres entiers (Cplex MIP).

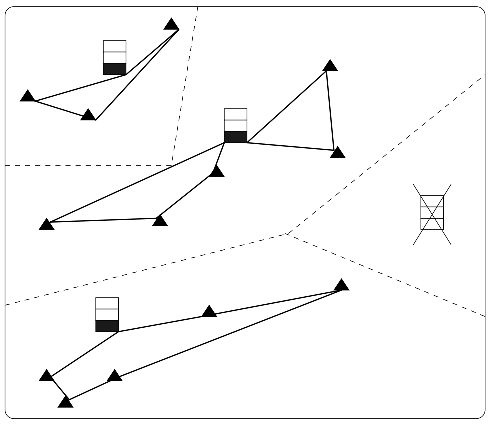


FIG. 3.5 – *Equipement des sous-sites BSC à la fin de la Phase 3 de l'heuristique pour l'exemple de la Figure 3.1*

3.3.5 Résultats expérimentaux

Nous considérons des instances réelles contenant jusqu'à 70 BTS 9 sites de BSC et 2 types d'équipements BSC. Pour toutes ces instances, le coût fixe par boucle est $CF = 10000$, ce qui est aussi le coût maximal dû à la longueur de la boucle ($CU = 1$ et $\bar{D} = 100000$). Rappelons par ailleurs que $\bar{n} = 14$ et $\underline{n} = 2$. La Table 3.1 donne des caractéristiques supplémentaires des instances que nous considérons. N est le nombre de BTS, M est le nombre de sites de BSC, NS est le nombre de sous-sites par site et NT est le nombre de types d'équipements BSC disponibles. La dernière colonne donne la densité du graphe G. Rappelons que $G = (V,A)$ est le graphe symétrique et orienté dont les sommets sont toutes les BTS et tous les sites de BSC et dont les arcs sont obtenus à partir du graphe complet en supprimant des arcs "impossibles" (voir début de la Section 3.3.2). La densité est donc $\frac{|A|}{(N+M)(N+M-1)}$. Les instances pb1 à pb5 correspondent à des réseaux de forte densité (en zone urbaine par exemple) et les instances pb6 à pb10 correspondent à des réseaux de densité moyenne. Dans ce dernier cas, les BTS et BSC étant plus éloignés, on élimine un plus grand nombre d'arcs "impossibles".

pb	N	M	NS	NT	densité
pb1	20	3	3	2	99%
pb2	30	3	3	2	99%
pb3	40	6	3	2	99%
pb4	50	6	3	2	99%
pb5	60	6	3	2	99%
pb6	30	4	3	2	50 %
pb7	40	4	3	2	49 %
pb8	50	5	3	2	48 %
pb9	60	5	3	2	48 %
pb10	70	5	3	2	50 %

TAB. 3.1 – *Quelques caractéristiques des instances de test du SDHNDP*

La Table 3.3.5 résume les résultats de l'heuristique et de différentes façons de calculer des bornes inférieures:

- s^* est le coût de la meilleure solution connue,
- $\bar{s}$ est le coût de la solution calculée par notre heuristique, décrite dans la Section 3.3.4, avec
$$\alpha = \frac{1}{2} + \frac{1}{\bar{D}} \underset{i \in \mathcal{N}}{max} \; \underset{j \in \mathcal{M}}{min} \; d_{ij}$$
- b_0 est la borne inférieure calculée par relaxation linéaire de la formulation (SDHNDP) avec $\beta = \lceil \frac{N}{\bar{n}} \rceil$,
- b_1 est la borne inférieure calculée par relaxation linéaire de la formulation (SDHNDP) avec β égale à la borne inférieure sur le nombre de boucles de toute solution admissible de (SDHNDP), calculée par l'algorithme de la Section 3.3.3,
- b_2 est l'amélioration de b_1 obtenue par Cplex (MIP) au bout de deux heures de temps de calcul. On a donc $b_0 \leq b_1 \leq b_2$.

La Colonne 4 (resp. 6, 8, et 10) indique l'erreur relative de $\bar{s}$ (resp. b_0, b_1 et b_2) par rapport à la valeur de la meilleure solution connue s^*. La Colonne 11 donne l'erreur relative entre b_1 et $\bar{s}$

pb	s^*	$\bar{s}$	$\frac{\bar{s}-s^*}{s^*}$	b_0	$\frac{s^*-b_0}{s^*}$	b_1	$\frac{s^*-b_1}{s^*}$	b_2	$\frac{s^*-b_2}{s^*}$	$\frac{\bar{s}-b_1}{\bar{s}}$
pb1	217723	217723	0%	213884	2%	213884	2%	215475	1%	2%
pb2	331858	331858	0%	320671	3%	320671	3%	321502	3%	3%
pb3	340947	441898	30%	323646	5%	323646	5%	323646	5%	27%
pb4	457892	544827	19%	430293	6%	430293	6%	430293	6%	21%
pb5	548303	548303	0%	534667	2%	534667	2%	534667	2%	2%
pb6	934635	934703	0%	737818	21%	896013	4%	908821	3%	4%
pb7	997237	1108545	11%	840098	16%	932949	6%	944339	5%	16%
pb8	1344134	1447388	8%	1078821	20%	1155491	14%	1157487	14%	20%
pb9	1491262	1505517	1%	1185502	21%	1208283	19%	1209294	19%	20%
pb10	1661551	1679399	1%	1224937	26%	1228846	26%	1230096	26%	27%
moy.			7%		12%		9%		8%	14%

TAB. 3.2 – *Résultats du calcul de borne inférieure et de l'heuristique pour SDHNDP*

c'est à dire l'approximation à laquelle l'on aboutit en se limitant à un effort de calcul réduit. En effet, le calcul de $\bar{s}$ par l'heuristique se fait en moins de 10 minutes de temps CPU pour chacune des instances et le calcul de b_1, par la résolution d'un programme linéaire se fait en entre 1 seconde et 22 minutes pour les instances de forte densité et entre 2 secondes et 2 minutes pour les instances de densité moyenne.

La comparaison des Colonnes 6 et 8 nous permet d'évaluer la contribution de la méthode d'amélioration de notre modèle par le calcul d'une plus grande borne inférieure sur le nombre de boucles (cf. Section 3.3.3):

- Pour les instances correspondant à des réseaux denses (instances pb1-pb5), cette contribution est nulle et s'explique par le fait que la borne inférieure calculée sur le nombre de boucles n'est pas mieux que $\lceil \frac{N}{\bar{n}} \rceil$ pour chacune de ces instances. Cette borne est d'ailleurs atteinte pour toutes les instances puisqu'elle est égale au nombre de boucles constaté dans les meilleures solutions connues.
- Par contre, pour les instances correspondant à des réseaux peu denses (instances pb6-pb10), l'amélioration est considérable puisqu'on passe d'une erreur moyenne $\frac{s^*-b_0}{s^*}$ égale à 21% (sur ces seules instances) à une erreur moyenne $\frac{s^*-b_1}{s^*}$ égale à 12%. Ici, la borne inférieure calculée sur le nombre de boucles est strictement supérieure à $\lceil \frac{N}{\bar{n}} \rceil$ pour chacune des instances. Par contre, elle atteint le nombre de boucles constaté dans les meilleures solutions connues pour les instances pb6 et pb7 seulement.

3.3.6 Prolongements possibles

Dans l'état actuel de notre travail sur ce problème, nous pensons à deux prolongements immédiats possibles:

- La section précédente a mis en évidence l'intérêt du calcul d'une bonne borne inférieure sur le nombre de boucles d'une solution admissible du problème (SDHNDP), tout au moins pour les instances correspondant à des réseaux peu denses. Notre algorithme de calcul d'une telle borne est assez efficace mais nous pensons qu'il peut être amélioré. Une façon de faire consiste à modifier le problème (P_k) de la Section 3.3.3 en ajoutant la contrainte que les arcs sélectionnés à l'intérieur d'un sous-ensemble doivent constituer un cycle élémentaire qui relie toutes les BTS d'un sous-ensemble et le site BSC de ce sous-ensemble. Ceci peut par exemple être réalisé par l'utilisation des coupes classiques suivantes, dont le problème de séparation associé est polynomial:

$$\sum_{k\in\{1,\ldots,K\}} \sum_{(i_1,i_2)\in\mathcal{B}(i_1<i_2)} y_{i_1i_2k} \leq |\mathcal{B}| - 1 \qquad \mathcal{B}\subset\mathcal{N},\, |\mathcal{B}|>1 \tag{3.33}$$

C'est d'ailleurs grâce à l'intégration de certaines de ces coupes que nous avons réussi, pour l'instance pb7, à mieux résoudre les problèmes (P_k) puis à en déduire une solution de valeur 997237 et à prouver son optimalité. Cette méthode de coupe mérite à notre avis d'être implémentée et testée.

- L'heuristique que nous avons implémentée est assez efficace. Les bons résultats qu'elle donne semblent indiquer que notre découpage en 3 phases ne s'écarte pas beaucoup de la nature globale du problème. Nous pensons qu'on peut cependant l'améliorer, par exemple en différant certaines décisions importantes, comme celle de l'ouverture des sites de BSC, à la fin de la Phase 1. On peut aussi, dans une démarche plus générale, essayer de l'améliorer par une procédure de recherche locale qui permettrait de revenir sur les décisions de chacune des phases.

Par ailleurs, nous nous sommes limités, dans notre approche de calcul de borne inférieure, à une modélisation par un programme linéaire mixte, contenant un nombre polynomial de variables et de contraintes. Ce choix a été fait dans le but d'utiliser directement des logiciels généralistes de résolution de programmes linéaires. Il nous semble toutefois indispensable également d'explorer des approches plus spécifiques à chaque problème comme des méthodes de coupes, surtout qu'on peut directement utiliser des coupes connues pour des problèmes très proches, comme par exemple les coupes d'élimination de sous-tours qui s'écrivent, avec les données et variables de notre formulation de (SDHNDP):

$$\sum_{i_1,i_2\in S} (v_{i_1i_2} + v_{i_2i_1}) <= |S| - 1 \quad S\subset\mathcal{N},\, |S|>1$$

et qui pourraient probablement être renforcées par:

$$\sum_{i_1,i_2\in S} (v_{i_1i_2} + v_{i_2i_1}) <= |S| - \lceil\frac{|S|}{\bar{n}}\rceil \quad S\subset\mathcal{N},\, |S|>1$$

3.4 Conclusion

Dans ce chapitre, nous avons décrit un problème "industriel" d'optimisation, qui se pose dans la phase de conception des réseaux de téléphonie mobile, dans la technologie SDH. Ce problème complexe a comme sous-problèmes, des problèmes biens connus en Optimisation Combinatoire : tournées de véhicules, localisation discrète et dimensionnement. Notre démarche a été d'abord de le modéliser par un programme linéaire en variables mixtes, le plus concis possible. Ce modèle, qui fait tout de même intervenir un grand nombre de variables et de contraintes, ne permet pas de résoudre les instances réelles. Toutefois, il permet de calculer une borne inférieure de la valeur de toute solution et nous avons amélioré cette borne inférieure en affinant certaines contraintes du modèle. Par ailleurs, nous avons conçu une méthode heuristique, de nature gourmande, pour calculer une solution admissible. Nos résultats montrent que l'écart moyen entre la borne inférieure et la valeur de la solution admissible ainsi calculées, est de 15% sur l'ensemble des instances que nous avons testées.

Ce travail constitue une première étape dans la tentative de résolution de ce nouveau problème et ouvre des perspectives intéressantes quant aux voies à suivre pour essayer de mieux le traiter, soit par une approche exacte soit par une approche heuristique.

Conclusion Générale et Perspectives

Ce rapport présente une synthèse d'un ensemble de travaux autour de la programmation mathématique en variables 0-1, très souvent utilisée en Optimisation Combinatoire. Nous présentons ici un certain nombre de conclusions et perspectives de ces travaux.

Le premier chapitre traite de programmation quadratique en 0-1. On s'intéresse précisément à des programmes mathématiques où les variables ne peuvent prendre que les valeurs 0 ou 1, la fonction objectif est quadratique et où des contraintes linéaires délimitent l'ensemble des solutions admissibles. Cette classe de problèmes inclut deux types de difficulté : d'une part le caractère non linéaire et en général non convexe de la fonction objectif et d'autre part le fait que les variables soient astreintes à prendre des valeurs entières. Un point crucial dans la résolution exacte des problèmes difficiles, par une approche de type *Procédure de Séparation et Evaluation*, est le calcul de bornes duales efficaces, c'est à dire à la fois proches des valeurs optimales et rapides à calculer. Le fil conducteur de l'ensemble des travaux présentés dans ce chapitre est l'étude comparative de différentes méthodes de calcul de bornes. Nous avons considéré deux types d'approches :

- Une première approche est la linéarisation, dont le but est de surmonter une des difficultés de ce problème en le transformant en un programme linéaire en nombres entiers, pour lequel on dispose de méthodes et outils de résolution performants. Nous avons abordé plusieurs linéarisations différentes :
  * La linéarisation classique, qui consiste à remplacer chaque produit de deux variables 0-1 par une nouvelle variable dans l'intervalle [0, 1].
  * La linéarisation produit, qui renforce la linéarisation classique par l'ajout de contraintes quadratiques, redondantes en 0-1; ces contraintes étant à leur tour linéarisées.
  * La linéarisation compacte, qui consiste à remplacer des sous-fonctions de la fonction objectif par de nouvelles variables réelles.
- Une deuxième approche est la méthode de décomposition, introduite au départ pour la programmation quadratique 0-1 sans contraintes, que nous avons généralisée au cas avec contraintes linéaires. Cette approche peut être vue comme un renforcement des méthodes de relaxation et décomposition lagrangiennes.

Notre contribution se situe d'une part à un niveau théorique et d'autre part à un niveau algorithmique et expérimental :

- Au niveau théorique,
  * nous avons prouvé que la méthode de décomposition fournit une borne au moins d'aussi bonne qualité que la linéarisation produit, dans le cas d'un programme quadratique général et

* nous avons prouvé que, pour le problème de la semi-affectation quadratique, la linéarisation produit fournit une réduction du problème. Une réduction est ici précisément une façon d'écrire un problème équivalent, dont la fonction objectif contient une constante c plus une somme de monômes de degré 1 ou 2, munis de coefficients positifs ou nuls. La constante c est alors une borne inférieure pour le problème de départ. Nous avons montré également que cette réduction est optimale dans le sens où c est la meilleure borne parmi celles fournies par d'autres réductions.

- Au niveau algorithmique et expérimental, nous avons discuté de l'implémentation des différentes méthodes et les avons comparées entre-elles au vu des critères de temps de calcul et de qualité de la borne calculée. Les tests que nous avons effectués ont considéré des problèmes "génériques" de la littérature : sac-à-dos quadratique, placement de tâches et semi-affectation quadratique.

Les conclusions principales et perspectives de ces travaux sont les suivantes :

- En présence de contraintes linéaires, la linéarisation produit est une méthode intéressante d'amélioration de la linéarisation classique. Comparée aux autres méthodes, elle fournit des bornes d'assez bonne qualité, voire une réduction optimale pour certains problèmes. Elle est générale et facile à mettre en œuvre car elle s'appuie sur la programmation linéaire. Cependant, elle nécessite la résolution de programmes linéaires dont la taille augmente très rapidement en fonction de la taille du problème, donc les temps de calcul deviennent vite très importants. Une voie de recherche future concernant cette méthode consiste à essayer de tirer profit de la structure de ces programmes linéaires pour améliorer leur résolution. Une autre voie de recherche est de prolonger les résultats théoriques obtenus pour certains problèmes. Par exemple, on montre aisément que la linéarisation produit permet également de réduire le problème de l'affectation quadratique et nous conjecturons que cette cette réduction est optimale.
- La linéarisation compacte pallie l'inconvénient principal de la linéarisation produit en utilisant des programmes linéaires dont la taille augmente beaucoup moins rapidement en fonction de la taille du problème. Cet avantage est considérable et fait que cette borne se calcule très vite. En terme de qualité, la borne obtenue par cette méthode est, pour les problèmes de placement de tâches, intermédiaire entre celle de la linéarisation classique et celle de la linéarisation produit. Nous pensons qu'une voie de recherche intéressante réside en l'amélioration de cette linéarisation, elle aussi très facile à mettre en œuvre. Cette amélioration peut se faire par l'introduction de coupes qui peuvent réduire le domaine des solutions fractionnaires admissibles.
- La méthode de décomposition calcule en théorie une borne au moins aussi bonne que la linéarisation produit et nos expériences de calcul montrent que dans certains cas, elle peut être aussi plus rapide. C'est donc une méthode très prometteuse mais elle présente l'inconvénient d'être plus délicate à mettre en œuvre : en plus du choix des paramètres de la méthode de sous-gradient, on a besoin, pour la méthode de décomposition, de faire un choix de partition de l'ensemble des variables et ce choix influence à la fois la qualité de la borne et son temps de calcul. Nous pensons que cette méthode mérite d'être davantage étudiée dans de futurs travaux de recherche.

Par ailleurs, nous avons l'intention d'explorer une voie différente de résolution ou d'approximation des programmes quadratiques en 0-1. Cette voie consiste à utiliser la programmation convexe continue. En effet, on peut "convexifier" la fonction objectif, c'est à dire la transformer pour obtenir une fonction convexe, équivalente quand les variables prennent les valeurs 0-1. On peut ensuite résoudre le programme quadratique convexe obtenu par relaxation continue pour obtenir une borne inférieure. Aujourd'hui, des solveurs commerciaux performants existent pour la programmation quadratique convexe. Reste que des techniques de convexification différentes aboutissent à des bornes inférieures différentes, il faut donc essayer de déterminer une méthode qui donne la meilleure borne possible. Nous pensons que cette approche constitue un axe de recherche intéressant pour les programmes quadratiques avec ou sans contraintes.

Dans le deuxième chapitre, nous nous intéressons essentiellement au problème du p-Centre qui est un problème de localisation discrète, avec un objectif min-max. Notre contribution tient essentiellement en deux points :

- Une nouvelle modélisation de ce problème. Une modélisation existait dans la littérature depuis une trentaine d'années. La nouvelle modélisation, comme la modélisation classique, utilise la programmation linéaire en nombres entiers mais nous montrons que la résolution de sa relaxation continue fournit une borne de bien meilleure qualité. En outre, si on utilise un solveur commercial de programmes linéaires en nombres entiers, la résolution est beaucoup plus rapide quand on considère la nouvelle formulation. Toutefois, cette nouvelle formulation ne permet pas directement de résoudre des instances de taille importante.
- Une nouvelle méthode de calcul de borne inférieure, que nous désignons par LB^*. Cette borne peut être obtenue par relaxation d'une partie des variables de la nouvelles formulation et résolution du programme linéaire en variables mixtes 0-1 obtenu. Mais nous proposons une méthode plus efficace, de complexité polynomiale, pour calculer cette même borne. De plus, nous prouvons que pour les instances euclidiennes la borne LB^* est supérieure ou égale à la moitié de la valeur optimale dans le cas particulier où les centres sont à placer sur les sites des clients. Dans le cas plus général que nous considérons, LB^* est supérieure ou égale au tiers de la valeur optimale. Notre étude expérimentale montre que cette borne est d'excellente qualité pour toutes les instances considérées par d'autres travaux de la littérature. Ces instances sont toutes euclidiennes. La qualité de cette borne nous permet de calculer une solution optimale pour des instances de taille jusqu'à deux fois plus importante que les travaux les plus récents. Nous pensons que l'efficacité de nos approches de formulation et de calcul de borne inférieure est en partie due au fait qu'elles exploitent la forte corrélation entre le problème du p-Centre et celui de recouvrement.

Par ailleurs, nous montrons comment ces approches de formulation et de calcul de borne inférieure s'étendent à des généralisations du problème du p-Centre. Nous appliquons également ces approches à un problème de codage qui se pose en Théorie de l'Information, et dont l'objectif est aussi de type min-max. Les axes de recherche future que nous prévoyons pour ce thème sont d'une part d'essayer de mieux résoudre les instances non-euclidiennes du problème du p-Centre et d'autre part de généraliser nos approches de formulation et de calcul de borne à d'autres problèmes ayant un objectif min-max.

Dans le troisième chapitre, nous nous intéressons à un problème de conception d'une partie du réseau de radio-téléphonie de nouvelle génération. Ce problème complexe a comme sous-problèmes des problèmes biens connus en Optimisation Combinatoire : tournées de véhicules, localisation discrète et dimensionnement. Notre démarche a été d'abord de le modéliser par un programme linéaire en variables mixtes, le plus concis possible. Ce modèle, qui fait tout de même intervenir un grand nombre de variables et de contraintes, ne permet pas de résoudre les instances réelles. Toutefois, il permet de calculer une borne inférieure de la valeur de toute solution et nous avons amélioré cette borne inférieure en affinant certaines contraintes du modèle. Par ailleurs, nous avons conçu une méthode heuristique, de nature gourmande, pour calculer une solution admissible. Ce travail constitue une première étape dans la tentative de résolution de ce nouveau problème et ouvre des perspectives intéressantes quant aux voies à suivre pour essayer de mieux le traiter, soit par une approche exacte soit par une approche heuristique.

Bibliographie

[1] W. P. Adams and H. D. Sherali. Linearization strategies for a class of zero-one mixed integer programming problems. *Operations Research*, 38(2):217–226, 1990.

[2] E. Balas and J.B. Mazzola. Nonlinear 0-1 programming i: Linearization techniques. *Mathematical Programming*, 30:1–21, 1984.

[3] E. Balas and J.B. Mazzola. Nonlinear 0-1 programming ii: Dominance relations and algorithms. *Mathematical Programming*, 30:22–45, 1984.

[4] J.E. Beasley. Or-library: Distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41(11):1069–1072, 1990.

[5] J.E. Beasley. Or-library: Distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41(11):1069–1072, 1990.

[6] A. Billionnet and F. Calmels. Linear programming for the 0-1 quadratic knapsack problem. *EJOR*, 92:310–325, 1996.

[7] A. Billionnet, M. C. Costa, and A. Sutter. An efficient algorithm for a task allocation problem. *Journal of the ACM*, 39(3):502–518, July 1992.

[8] A. Billionnet and S. Elloumi. Placement de tâches dans un système distribué et dualité lagrangienne. *Revue d'Automatique, d'Informatique et de Recherche Opérationnelle (R.A.I.R.O.), série verte*, 26(1):83–97, 1992.

[9] A. Billionnet and S. Elloumi. Placement des tâches d'un programme à structure arborescente sur un réseau de processeurs : synthèse de résultats récents. *Information Systems and Operational Research (INFOR)*, 32(2):65–86, 1994.

[10] A. Billionnet and S. Elloumi. An algorithm for finding the K-best allocations of a tree structured program. *Journal of Parallel and Distributed Computing*, 26(2):225–232, 1995.

[11] A. Billionnet and S. Elloumi. Best reduction of the quadratic semi-assignment problem. *DAMATH: Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science*, 109:197–213, 2001.

[12] A. Billionnet, S. Elloumi, and L. Grouz Djerbi. Designing radio-mobile access networks based on sdh rings. Technical Report 325, CEDRIC, 2002.

[13] A. Billionnet and A. Faye. A lower bound for a constrained quadratic 0-1 minimization problem. *DAMATH: Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science*, 74, 1997.

[14] A. Billionnet, A. Faye, and E. Soutif. A new upper bound for the 0-1 quadratic knapsack problem. *European Journal of Operational Research*, 112:664–672, 1999.

[15] E. Boros and P. L. Hammer. Pseudo-boolean optimization. Technical Report TR: 2001-33, DIMACS, 2001.

[16] E. Boros, P. L. Hammer, and X. Sun. Network flows and minimization of quadratic pseudo-boolean functions. Technical Report TR: 1991-17, RUTCOR, 1991.

[17] J. Cardinal, S. Elloumi, and M. Labbé. Local optimization of index assignments for multiple description coding, 2002. XIth European Signal Processing Conference (EUSIPCO-2002), Toulouse, September 2002.

[18] C.T. Chang and C.C. Chang. A linearization method for mixed 0-1 polynomial programs. *Computers and Operations Research*, 27:1005–1016, 2000.

[19] P. Chardaire and A. Sutter. A decomposition method for quadratic zero-one programming. *Management Science*, 41(4):704–712, 1995.

[20] G. Clarke and J.W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581, 1964.

[21] Cplex. *ILOG CPLEX Division.* ILOG CPLEX Division, 889 Alder Avenue, Suite 200, Incline Village, NV 89451.

[22] G.B. Dantzig. On the significance of solving linear programming problems with some integer variables, 1958. The Rand Corporation, document p. 1486.

[23] M. Daskin. Network and discrete location: Models, 1995.

[24] M. Daskin. A new approach to solving the vertex p-center problem to optimality: Algorithm and computational results. *Communications of the Operations Research Society of Japan*, 45(9):428–436, 2000.

[25] B. Decocq. *Placement de tâches dans les systèmes distribués.* Thèse de doctorat en informatique, Conservatoire National des Arts et Métiers, Paris, 1996.

[26] S. Elloumi, A. Faye, and E. Soutif. Decomposition and linearization for 0-1 quadratic programming. *Annals of Operations Research*, 99:79–93, 2000.

[27] S. Elloumi, M. Labbé, and Y. Pochet. A new formulation and exact solution method for the p-center problem, 2001. CORE Discussion Paper 2001/53. A paraître dans INFORMS Journal on Computing.

[28] S. Elloumi, M. Labbé, and Y. Pochet. Formulation et résolution d'un problème de p-centre tolérant aux pannes, 2002. Congrès National de la ROADEF.

[29] S. Elloumi and E. Soutif. Comparaison expérimentale de différentes bornes pour un problème de placement de tâches. Technical Report 323, CEDRIC, 2002.

[30] A. Faye. *Programmation quadratique en variables bivalentes sous contraintes linéaires. Application au placement de tâches dans les systèmes distribués et à la partition de graphes.* Thèse de doctorat en informatique, Conservatoire National des Arts et Métiers, Paris, 1994.

[31] R. Fortet. L'algebre de boole et ses applications en recherche operationelle. *Cahier du Centre d'Etudes de Recherche Opérationnelle*, 4:5–36, 1959.

[32] R. Fourer, D. M. Gay, and B. W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming.* The Scientific Press (now an imprint of Boyd & Fraser Publishing Co.), Danvers, MA, USA, 1993.

[33] A. M. Geoffrion. Lagrangean relaxation for integer programming. *Mathematical Programming Study*, 2:82–114, 1974.

[34] F. Glover. Improved linear integer programming formulations of nonlinear integer problems. *Management Science*, 22:455–460, 1975.

[35] F. Granot and P.L. Hammer. On the use of boolean functions in 0-1 programming. *Methods of Operations Research*, 12:154–184, 1971.

[36] L. Grouz-Djerbi. *Conception et dimensionnement de la partie fixe des réseaux d'accès radiomobiles à travers une architecture en boucles SDH.* Thèse de doctorat en informatique, Conservatoire National des Arts et Métiers, Paris, 2001.

[37] S. Gueye and P. Michelon. A linearization framework for unconstrained quadratic 0/1 problems. Technical Report 261, LIA, 2001.

[38] M. Guignard. Lagrangean relaxation: a short course. *Jorbel*, (3-4):5–21, 1995.

[39] M. Guignard and S. Kim. Langrangean decomposition for integer programming: Theory and applications. *RAIRO Recherche Opérationnelle*, 21(4):307–323, 1987.

[40] P. L. Hammer, P. Hansen, and B. Simeone. Roof duality, complementation and persistency in quadratic 0-1 optimization. *Mathematical Programming*, 28:121–155, 1984.

[41] P. Hansen, B. Jaumard, and V. Mathon. Constrained nonlinear 0–1 programming. *ORSA Journal on Computing*, 5:97–119, 1993.

[42] P.H. Hansen, B. Hegedahl, S. Hjortkjær, and B. Obel. A heuristic solution to the warehouse location-routing problem. *Eur. J. Operational Research*, 76(1):111–127, 1994.

[43] C. Helmberg. Semidefinite programming for combinatorial optimization. Technical Report ZR-00-34, Konrad-Zuse-Zentrum, Berlin, 2000.

[44] D. Hochbaum and D. Shmoys. A best possible heuristic for the k-center problem. *Math. Oprns. Res.*, 10:180–184, 1985.

[45] T. Ilhan, , and M. C. Pinar. An efficient exact algorithm for the vertex p-center problem. Technical report, Bilkent University, Department of Industrial Engineering, Ankara, Turkey, September 2001.

[46] S. E. Karisch, E. Çela, J. Clausen, and T. Espersen. A dual framework for lower bounds of the quadratic assignment problem based on linearization. *Computing*, 63:351–403, 1999.

[47] O. Kariv and S.L. Hakimi. An algorithmic approach to network location problems. part 1: The p-centers. *SIAM J. Appl. Math.*, 37:513–538, 1979.

[48] S. O. Krumke. On a generalization of the p-center problem. *Information Processing Letters*, 56(2):67–71, 1995.

[49] X. Lagrange, P. Godlewski, and S. Tabbane. *Réseaux GSM-DCS, des principes à la norme.* Hèrmes, 1999.

[50] G. Laporte. Location routing problems. In *Vehicle Routing: Methods and Studies*, volume 16, pages 163–197, 1988.

[51] P. Michelon and N. Maculan. Lagrangian decomposition for integer nonlinear programming with linear constraints. *Mathematical Programming*, 52:303–313, 1991.

[52] C. E. Miller, A. W. Tucker, and R. A. Zemlin. Integer programming formulations and traveling salesman problems. *J. Assoc. Comput. Mach.*, 7:326–329, 1960.

[53] H. Min, V. Jayaraman, and R. Srivastava. Combined location–routing problems: A synthesis and future research directions. *Eur. J. Operational Research*, 108(1):1–15, 1998.

[54] N. Mladenovic, M. Labbé, and P. Hansen. Tabu search and variable neighborhood search in solving the p-center problem. *ULB-ISRO preprint 2000/20, (available at http://smg.ulb.ac.be/)*, 2000.

[55] P. Pardalos, F. Rendl, and H. Wolkowicz. The quadratic assignment problem: a survey of recent developments. In P.Pardalos and H.Wolkowicz, editors, *Quadratic assignment and related problems*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, pages 1–42. AMS, 1994.

[56] J. Perl and M. Daskin. A warehouse location-routing problem. *Transportation Research*, 19B:381–396, 1985.

[57] R.C. Prim. Shortest connection networks and some generalization. *The Bell System Technical Journal*, 36:1389–1401, 1957.

[58] G. Reinelt. TSPLIB— A traveling salesman problem library. *ORSA Journal on Computing*, 3(4):376–384, 1991.

[59] C. ReVelle, C. Toregas, R. Swain, and L. Bergman. The location of emergency service facilities. *Operations Research*, 19:1363–73, 1971.

[60] E. Soutif. *Résolution du problème du sac-à-dos quadratique en variables bivalentes.* Thèse de doctorat en informatique, Conservatoire National des Arts et Métiers, Paris, 2000.

[61] D. Tuzun and L.I. Burke. A two-phase tabu search approach to the location routing proble. *Eur. J. Operational Research*, 116(1):87–99, 1999.