Résolution de programmes quadratiques en nombres entiers

Amélie Lambert

▶ **To cite this version:**

Amélie Lambert. Résolution de programmes quadratiques en nombres entiers. Recherche opérationnelle [math.OC]. Conservatoire National des Arts et Métiers (CNAM), 2009. Français. NNT : . tel-02459253

HAL Id: tel-02459253
https://hal.science/tel-02459253

Submitted on 29 Jan 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Centre d'Étude et de Recherche en Informatique du CNAM – EA 1395

Ecole Doctorale d'Informatique Télécomunications et Electronique

Résolution de programmes quadratiques en nombres entiers

THÈSE

présentée et soutenue publiquement le 26 novembre 2009

pour l'obtention du

Doctorat du Conservatoire National des Arts et Métiers

(spécialité informatique)

par

Amélie Lambert

Composition du jury

Directeurs de thèse :	Alain Billionnet Professeur à l'Ecole Nationale Supérieure d'Informatique pour l'Industrie et l'Entreprise Sourour Elloumi Maître de conférences à l'Ecole Nationale Supérieure d'Informatique pour l'Industrie et l'Entreprise
Rapporteurs :	Walid Ben Ameur Professeur à Télécom & Management SudParis Franz Rendl Professeur à l'Université de Klagenfurt
Examinateurs :	Pierre Hansen Professeur à HEC Montréal Frédéric Roupin Maître de conférences au Conservatoire National des Arts et Métiers François Vanderbeck Professeur à l'Université Bordeaux 1

Conservatoire National des Arts et Métiers

Remerciements

Les deux premières personnes que je tiens à remercier sont ceux sans qui cette thèse n'aurait jamais vu le jour. On peut déjà leur accorder la conception des idées qui sont la base de ce travail, mais aussi, on peut leur reconnaître l'énorme investissement dans mon encadrement, aussi bien au niveau scientifique et méthodologique, que dans mon intégration au sein de l'équipe Optimisation Combinatoire. Alors, pour tout ça et pour le reste, merci à vous, Alain et Sourour, de m'avoir permis d'aller au bout de cette thèse.

Je remercie également Walid Ben Ameur et Franz Rendl de m'avoir fait l'honneur d'être rapporteurs de cette thèse. J'éprouve un profond respect pour la qualité de leur travail, et le regard encourageant qu'ils ont porté sur mes travaux me touche beaucoup.

Merci à Pierre Hansen, Frédéric Roupin et François Vanderbeck d'avoir accepté de faire parti de mon jury.

Je remercie aussi, les membres du laboratoire CEDRIC, et tout particulièrement les membres de mon équipe Christophe, Marie-Christine, Eric, Agnès, Alain, Mathieu et Benoît. Hélène mérite bien une phrase entière pour la remercier d'avoir partagé notre bureau, relu ma thèse et mes papiers, partagé les réjouissances des pots de thèse, et bien d'autre choses encore.

Je remercie également tous ceux qui ont étés présents pendant ces trois années : Antoine, Mélu, Manue, William, Manue, Ben, Loig, Mathilde, Greg, Rémi, Marc et tous ceux que j'oublie.

Je remercie évidemment ma famille sans qui cette thèse n'aurait pu être. Je pense à mes frères, Olivier et Benoît, ma soeur, Sophie, et bien sûr mes parents, que je remercie d'ailleurs particulièrement de m'avoir toujours soutenue et encouragée. Je remercie également tout le reste de la famille, dont je ne citerai ici que quelques noms : Bon-papa, Constance, Mathilde, Thierry et Nathalie. Un remerciement tout spécial à Manu pour avoir participer à cette thèse au cours de son stage.

Je crois qu'il y en a un qui mérite son paragraphe. Je te remercie pour un tas de choses : le soutien au quotidien, la participation à la relecture, réécriture de cette thèse, de mes papiers. Ainsi que pour toutes les idées que tu m'a soufflées. Merci pour tout Elie.

Table des matières

Notations **5**

Introduction **7**

1 Etat de l'art **13**

1.1 La programmation quadratique non convexe en nombres entiers 13

1.2 La programmation quadratique non convexe en variables mixtes 18

2 Certaines reformulations de la programmation quadratique en variables 0-1 utilisées dans cette thèse **21**

2.1 Reformulations linéaires de programmes quadratiques en variables 0-1 22

2.2 Reformulations quadratiques convexes de programmes quadratiques en variables 0-1 24

2.2.1 La méthode de la plus petite valeur propre, Hammer et Rubin, 1970 . . . 24

2.2.2 La méthode `UQCR (Unconstrained Quadratic Convex Reformulation)`, Billionnet et Elloumi, 2007 25

2.2.3 La méthode `QCR (Quadratic Convex Reformulation)`, Billionnet, Elloumi et Plateau, 2009 27

3 Reformulations linéaires de programmes quadratiques en nombres entiers **31**

3.1 Introduction et exemple de base 31

3.2 Une reformulation basée sur la linéarisation du produit de deux variables binaires : `BBL (Binary Binary Linearization)` 32

3.2.1 La méthode `BBL` 33

3.2.2 Son renforcement `BBLr` 34

3.2.3 Application à l'exemple de base 35

3.3 Une reformulation basée sur la linéarisation du produit d'une variable binaire par une variable entière : `BIL (Binary Integer Linearization)` 36

3.3.1 La méthode `BIL` 37

3.3.2 Son renforcement `BILr` 38

3.3.3 Application à l'exemple de base 40
3.4 Comparaison théorique de BBL et BIL 41
3.5 Conclusion 42

4 Reformulations quadratiques convexes de programmes quadratiques en nombres entiers 43

4.1 Introduction 43
4.2 Une reformulation convexe naïve : NC (Naive Convexification) 45
4.3 La méthode IQCR (Integer Quadratic Convex Reformulation) 47
4.3.1 Un schéma général de reformulations convexes pour les programmes quadratiques en nombres entiers 47
4.3.2 Une projection du polyèdre associé 50
4.3.3 Calcul de la meilleure convexification au sein de ce schéma : la méthode IQCR 52
4.3.4 Extension de IQCR aux problèmes mixtes entiers : la méthode IQCRs . . . 57
4.3.5 Utilisation des contraintes d'inégalité pour perturber la fonction objectif . 63
4.3.6 Application à l'exemple de base 64
4.4 Une restriction intéressante de IQCR : CQCR (Compact Quadratic Convex Reformulation) 68
4.5 Interprétation des méthodes NC, CQCR et IQCR pour la programmation quadratique en variables 0-1 73
4.5.1 La méthode NC correspond à la méthode de la plus petite valeur propre . 74
4.5.2 La méthode CQCR correspond à la méthode QCR 74
4.5.3 La méthode IQCR correspond à un renforcement de QCR 76
4.6 Conclusion 80

5 Un algorithme de Branch and Bound spécifique pour IQCR fondé sur les propriétés de projection de la Section 4.3.2 83

6 Résultats expérimentaux 89

6.1 Résultats expérimentaux pour les problèmes quadratiques en variables entières soumis à des contraintes linéaires 90
6.1.1 Présentation des classes de problèmes étudiées 90
6.1.2 Les résultats 92
6.2 Résultats expérimentaux pour les problèmes quadratiques en variables 0-1 99
6.2.1 Résultats pour le problème non contraint : les instances de Pardalos et Rodgers (1990) 99
6.2.2 Résultats pour des problèmes avec contraintes d'égalités : le problème d'affectation de tâches 101
6.3 Conclusion 102

Conclusion **103**

Bibliographie **109**

Annexes **113**

A Algorithmes et outils de calcul de la programmation semi-définie **115**

A.1 Rappel : les matrices semi-définies positives 115

A.2 La programmation semi-définie 116

A.3 Théorie de la dualité 118

A.4 Optimisation fondée sur les valeurs propres 121

A.5 Les algorithmes de résolution de la SDP 123

A.5.1 La méthode des points intérieurs, Borchers, 1999 123

A.5.2 La méthode des faisceaux (Spectral Bundle), Helmberg, 2000 125

B Details des résulats expérimentaux **127**

B.1 Résultats pour la classe de problèmes ($EIQP$) 128

B.2 Résultats pour la classe de problèmes ($IIQP$) 135

B.3 Résultats pour les instances de Pardalos and Rodgers 142

B.4 Résultats pour le problème d'affectation de tâche 145

C Le logiciel SIQP (Solution of Integer Quadratic Programs) **147**

Table des figures

2.1 Algorithme de résolution de (QP^{01}) basé sur le reformulation `UQCR` 27
2.2 Algorithme de résolution de (QP^{01}) basé sur le reformulation `QCR` 29

4.1 Algorithme de résolution de (QP) basé sur la reformulation `IQCR` 57
4.2 Algorithme de résolution de (QP) basé sur la reformulation `CQCR` 72
4.3 Algorithme de résolution de (QP^{01}) basé sur la reformulation `IQCR` 78

5.1 Les branchements possibles dans l'algorithme de Branch and Bound basé sur les solutions de ($P^e_{\alpha,\beta}$) 86

1 Tableau récapitulatif des méthodes proposées dans cette thèse 104

Résumé

Soit (QP) un programme quadratique en variables entières qui consiste à minimiser une fonction quadratique soumise à des contraintes linéaires. Un tel problème appartient à la classe des problèmes $\mathcal{NP}$-difficiles. Les solveurs standards qui utilisent des algorithmes de Branch and Bound peuvent résoudre efficacement (QP) dans le cas particulier où sa fonction objectif est convexe. Ainsi, pour résoudre (QP) nous avons choisi de le reformuler en un programme équivalent ayant une fonction objectif convexe. Deux cas sont alors possibles : soit nous reformulons (QP) en un programme linéaire, soit nous le reformulons en un programme quadratique et convexe.

Dans la première partie de cette thèse, nous présentons plusieurs reformulations linéaires de (QP), i.e. reformulations en un programme équivalent qui a une fonction objectif linéaire. Il existe de nombreuses méthodes de linéarisation pour la programmation quadratique binaire. Une approche naturelle pour résoudre (QP) est donc de le reformuler en un programme quadratique en variables binaires. Cela peut être fait en remplaçant chaque variable entière par sa décomposition binaire, puis en linéarisant chaque nouveau produit de variables binaires. Cependant, cette méthode que nous appelons `BBL (Binary Binary Linearization)`, fournit un programme linéaire avec un grand nombre de variables et de contraintes. Nous proposons donc une nouvelle approche, `BIL (Binary Integer Linearization)`, qui consiste à reformuler (QP) en un programme quadratique particulier où chaque terme quadratique est le produit d'une variable entière par une variable binaire. Comme dans la méthode `BBL`, les variables binaires viennent de la décomposition binaire des variables entières initiales. Ensuite, nous linéarisons le programme obtenu en remplaçant chaque terme quadratique par une variable réelle et un ensemble d'inégalités. Comme le nombre de termes quadratiques est plus petit que dans la méthode `BBL`, le nombre de variables additionnelles est réduit. Donc, le programme obtenu avec la méthode `BIL` est significativement plus petit. De plus, contrairement à ce que l'on pourrait attendre, l'approche `BIL` fournit une borne obtenue par relaxation continue de meilleure qualité que celle fournie par l'approche `BBL`. Chaque reformulation aboutit à un programme linéaire équivalent à (QP) que nous renforçons en lui ajoutant des inégalités valides.

Dans une deuxième partie, nous présentons plusieurs reformulations quadratiques convexes de (QP), i.e. nous reformulons (QP) en un programme équivalent ayant une fonction objectif quadratique et convexe. Nous introduisons d'abord une approche simple pour convexifier (QP) qui consiste à exprimer linéairement les carrés des variables entières en utilisant leurs décompo-

sitions unaires, puis à convexifier à l'aide de la plus petite valeur propre du Hessien de la fonction objectif de (QP). Nous appelons cette méthode `NC (Naive Convexification)`. Ensuite, nous introduisons un nouveau schéma de reformulations convexes qui perturbe la fonction objectif à l'aide de l'expression linéaire des produits des variables entières initiales, et des contraintes d'égalité de (QP). Puis nous montrons que nous pouvons calculer au sein de ce schéma une reformulation optimale du point de vue de la borne obtenue par relaxation continue : la reformulation `IQCR (Integer Quadratic Convex Reformulation)`. Cette reformulation est basée sur la solution optimale duale d'une relaxation semi-définie de (QP). De plus, nous montrons que la méthode `IQCR` peut s'adapter facilement à la programmation mixte entière. Cette adaptation, que nous appelons `IQCRs` permet également d'intégrer les contraintes d'inégalité dans la perturbation de la fonction objectif de notre schéma de reformulations convexes. Ensuite, nous présentons une restriction intéressante de la méthode `IQCR`, appelée `CQCR (Compact Quadratic Convex Reformulation)`. La différence entre cette approche et la méthode `IQCR` est que `CQCR` n'utilise que les expressions linéaires des carrés des variables pour perturber la fonction objectif, alors qu'`IQCR` utilise celles de tous les produits. L'intérêt est que de `CQCR` fournit un programme reformulé de plus petite taille en comparaison avec celui de l'approche `IQCR`, ce qui peut être avantageux expérimentalement. Finalement, nous appliquons les 3 méthodes `NC`, `CQCR` et `IQCR` à la programmation quadratique binaire. Nous montrons que `NC` et `CQCR` sont équivalentes à des méthodes déjà connues. Un résultat intéressant est que `IQCR` constitue une amélioration des convexifications existantes pour la programmation quadratique binaire.

Dans une troisième partie, nous présentons un algorithme de Branch and Bound spécifique basé sur une propriété de projection de la méthode `IQCR`.

Finalement, nous comparons expérimentalement les quatre reformulations linéaires et les trois reformulations quadratiques et convexes de (QP) que nous avons obtenues. Dans le cas où les variables sont entières, les expérimentations concernent deux classes d'instances de (QP), l'une ayant une contrainte d'égalité et l'autre une contrainte d'inégalité. Les résultats montrent que la plupart des instances ayant jusqu'à 40 variables peuvent être résolues en moins d'une heure avec un solveur standard par les reformulations `IQCR` et `CQCR`. Nous testons ensuite notre reformulation convexe `IQCR` sur la programmation quadratique binaire. Les résultats confirment que notre approche `IQCR` améliore les convexifications existantes.

Mots-clés: Programmation en nombres entiers, programmation en variables mixtes-entières, programmation quadratique, reformulation linéaire, reformulation quadratique convexe, programmation semi-définie, expérimentations

Abstract

Let (QP) be an integer quadratic program that consists in minimizing a quadratic function subject to linear constraints. A such problem belongs to the class of $\mathcal{NP}$-Hard problems. Standard solvers that use a Branch and Bound algorithm can efficiently solve (QP) in the specific case where its objective function is convex. Thus, to solve (QP), we choose to reformulate it into an equivalent problem with a convex objective function. Two reformulations are possible : either we reformulate (QP) into a linear program, or we reformulate it into a convex quadratic program.

In the first part of this dissertation, we present several linearizations of (QP), i.e. reformulations into an equivalent program with a linear objective function. Many linearization methods for the quadratic binary programs are known. A natural approach when considering (QP) is therefore to reformulate it into a quadratic binary program. This can be done by the binary decomposition of each integer variable and then the linearization of each new product of two binary variables. However, this method, that we denote by `BBL (Binary Binary Linearization)`, leads to a linear program with a large number of variables and constraints. We then present a new approach, `BIL (Binary Integer Linearization)`, that consists in reformulating (QP) into a particular quadratic integer program where each quadratic term is the product of an integer variable by a binary variable. As in the `BBL` approach, the binary variables come from the binary decomposition of the initial integer variables. Then, we linearize the obtained program by replacing each quadratic term by a new real variable and a set of inequalities. As the number of quadratic terms is lower than in the `BBL` approach, the number of additional variables is reduced. Hence, the obtained integer linear program is significantly smaller in the `BIL` approach. Moreover, contrary to what one might think, the `BIL` approach provides a better bound obtained by continuous relaxation than the `BBL` approach. Each reformulation leads to an integer linear program that is equivalent to (QP) and that we improve by adding valid inequalities.

In a second part, we present several quadratic convex reformulations of (QP), i.e. we reformulate (QP) into an equivalent program, with a quadratic convex objective function. We first introduce a simple approach to convexify (QP) that consists in expressing linearly the squares of integer variables using their unary decompositions, and then to convexify with the smallest eigenvalue of the Hessian matrix of (QP). We call this approach `NC (Naive Convexification)`. Then, we introduce a new convex reformulation scheme that perturbs the objective function of (QP) with the linear expression of the products of integer variables, and the equality constraints

of (QP). Then, we show that we can compute, within this scheme, an optimal reformulation of (QP) in terms of bound obtained by continuous relaxation : the `IQCR (Integer Quadratic Convex Reformulation)` approach. This reformulation is based on the optimal dual solution of a semi-definite relaxation of (QP). Moreover, we show that the method `IQCR` is easily adaptable to mixed-integer programming. This adaptation, that we call `IQCRs`, also allows us to integrate the inequality constraints of (QP) into the perturbation of the objective function of our convex reformulation scheme. Then, we present an interesting restriction of the method `IQCR`, called `CQCR (Compact Quadratic Convex Reformulation)`. The difference between this last approach and `IQCR` is that `CQCR` only uses the linear expression of the integer variable squares to perturb the objective function, while `IQCR` uses all the products. The interest is that `CQCR` produces a reformulated problem with a reduced size in comparison with the `IQCR` approach, what could be profitable experimentally. Finally, we apply these 3 methods `NC`, `CQCR` and `IQCR` to binary quadratic programming. We show that `NC` and `CQCR` are equivalent to existing binary convex reformulations. An interesting result is that `IQCR` is an improvement of existing convexifications for binary quadratic programming.

In a third part, we design a specific Branch and Bound algorithm based on a projection property of the `IQCR` approach.

Finally, we compare the four obtained linearizations and the three obtained quadratic convex reformulations from the computational point of view. For integer programming, computational experiences are carried out with two classes of instances of (QP), the first having one equality constraint, and the other having one inequality constraint. The results show that most of the considered instances with up to 40 variables can be solved in one hour of CPU time by the `IQCR` and `CQCR` approaches. We then test `IQCR` on binary quadratic programming. The results corroborate that our approach `IQCR` improves existing convexifications.

Keywords: Integer programming, mixed-integer programming, quadratic programming, linear reformulation, quadratic convex reformulation, semi-definite programming, experiments

Notations

- $\mathbf{S_n}$ espace des matrices symétriques réelles.
- $\mathbf{S_n^+}$ espace des matrices symétriques réelles semi-définies positives.
- $\mathbf{S_n^{++}}$ espace des matrices symétriques réelles définies positives.
- $X \succeq 0$, la matrice X est semi-définie positive.
- $X \succ 0$, la matrice X est définie positive.
- $tr(A)$ trace de la matrice A.
- $\lambda_{min}(A)$ plus petite valeur propre de A.
- $\lambda_{max}(A)$ plus grande valeur propre de A.
- I la matrice identité.
- $\mathbf{0}$ la matrice nulle.

Introduction

De nombreux problèmes de recherche opérationnelle peuvent se modéliser sous la forme d'un programme mathématique en variables entières dont la fonction objectif est quadratique, non convexe et est soumise à des contraintes linéaires. Par exemple, une telle formulation est utilisée dans certains problèmes de graphes multi-parties [14], de planification avec capacités [23] ou d'optimisation de découpes dans le plan [9]. Sans perte de généralité un problème de ce type peut s'écrire sous la forme :

$$(QP)\begin{cases} Min & f(x) = \sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij}x_i x_j + \sum_{i=1}^{n} c_i x_i & & \\ s.c. & \sum_{i=1}^{n} a_{ri}x_i = b_r & r \in R & (1) \\ & \sum_{i=1}^{n} d_{si}x_i \leq e_s & s \in S & (2) \\ & x_i \leq u_i & i \in I & (3) \\ & x_i \geq 0 & i \in I & (4) \\ & x_i \in \mathbb{N} & i \in I & (5) \end{cases}$$

où $Q \in \mathbf{S}_n$ (l'espace des matrices symétriques d'ordre n), $c \in \mathbb{R}^n$, $A \in \mathbf{M}_{m,n}$ (ensemble des matrices $m \times n$), $b \in \mathbb{N}^m$, $D \in \mathbf{M}_{p,n}$, $e \in \mathbb{N}^p$, $u \in \mathbb{N}^n$, $I = \{i : i = 1, \ldots, n\}$, $R = \{r : r = 1, \ldots, m\}$ et $S = \{s : s = 1, \ldots, p\}$.

Ce problème appartient à la classe des problèmes $\mathcal{NP}$-difficiles [16]. Dans la littérature, seuls quelques travaux considèrent la résolution de (QP) [27]. Cependant, il existe une littérature plus abondante concernant la résolution de programmes quadratiques non convexes en variables mixtes (MIQP : Mixed Integer Quadratic Programs) [30, 1, 39, 40, 12, 37]. Les $(MIQP)$ sont des problèmes où l'ensemble de définition des variables x_i est mixte réel-entier, (QP) en est donc un cas particulier où l'ensemble des variables réelles est l'ensemble vide.

Il existe des algorithmes pour résoudre les programmes quadratiques en variables mixtes, mais seulement dans le cas particulier où $f(x)$ est une fonction convexe. Ces algorithmes, basés sur le Branch and Bound, sont implantés efficacement dans les solveurs standards [24, 10, 7].

Leur principe est de parcourir un arbre des solutions entières réalisables de (QP) en se basant sur la valeur optimale de la relaxation continue de (QP). Concrètement, à chaque noeud de l'arbre on résout un problème polynômial, i.e. la relaxation continue de (QP).

Afin d'utiliser les performances des solveurs standards, nous avons choisi pour résoudre (QP) de le reformuler en un programme équivalent dont la fonction objectif est convexe : soit linéaire, soit quadratique et convexe. Nous proposons dans cette thèse plusieurs reformulations de (QP) soit en un programme équivalent dont la fonction objectif est linéaire, soit en un programme équivalent dont la fonction objectif est quadratique et convexe. Afin que ces reformulations soient efficaces, nous essayons pour chacune de réduire au maximum l'écart entre la valeur de la solution optimale en nombres réels et la valeur de la solution optimale en nombres entiers.

Dans la littérature, la majorité des reformulations de programmes quadratiques en variables entières sont conçues pour des problèmes en variables bivalentes. Dans ce type de problèmes, qui est en fait une restriction de (QP), la bivalence des variables induit certaines propriétés qui ne sont plus satisfaites lorsque les variables sont entières. Comme pour (QP), les solveurs standards ne savent les résoudre efficacement que si leur fonction objectif est convexe. Les reformulations linéaires, ainsi que les reformulations quadratiques convexes de programmes en variables binaires qui ont été étudiées dans ces 20 dernières années sont les bases des résultats de cette thèse.

La reformulation linéaire de Fortet [13] est probablement la linéarisation la plus utilisée en programmation quadratique binaire. Plusieurs améliorations de cette méthode ont été proposées dont la principale est la linéarisation RLT [38] qui améliore de façon significative la valeur optimale de la relaxation continue du problème reformulé. Une façon naturelle de résoudre (QP) est donc de le transformer en un programme en variables bivalentes, en remplaçant chaque variable entière par sa décomposition binaire, puis de lui appliquer la linéarisation de [13] et son amélioration [38]. Nous présentons en premier cette méthode, appelée `BBL (Binary Binary Linearization)`. Puis, nous améliorons la méthode `BBL` par les idées de [38] dans une méthode nommée `BBLr (Binary Binary Linearization reinforced)`. L'inconvénient majeur de cette méthode directe est la taille importante du problème reformulé qui pénalise fortement le temps de résolution.

Nous proposons donc une nouvelle méthode de reformulation linéaire, que nous appelons `BIL (Binary Integer Linearization)`. Elle consiste toujours à utiliser pour chaque variable entière sa décomposition binaire. Cependant, dans chaque produit de deux variables entières nous ne remplaçons qu'une seule variable entière par sa décomposition binaire. Ensuite, nous linéarisons chaque nouveau produit, composé d'une variable entière et d'une variable binaire, en ajoutant de nouvelles variables réelles et un ensemble d'inégalités, comme développé dans [31]. Cette méthode nous permet d'obtenir un programme reformulé de plus petite taille que dans la

méthode `BBL`, dont la valeur de la borne obtenue par relaxation continue est de meilleure qualité. De la même façon que pour la méthode `BBLr`, nous améliorons la méthode `BIL` en termes de valeur de relaxation continue dans une méthode appelée `BILr (Binary Integer Linearization reinforced)`.

Nous proposons ensuite plusieurs reformulations quadratiques convexes de (QP). Par reformulation convexe, nous entendons un problème quadratique équivalent à (QP) dont la fonction objectif est quadratique et convexe, i.e dont le Hessien est semi-défini positif (SDP). Dans la littérature, les reformulations convexes ont également été introduites pour des problèmes quadratiques en variables bivalentes. En effet, il est facile de convexifier un programme en variables bivalentes en utilisant la propriété $x_i^2 = x_i$. Hammer et Rubin [19] utilisent cette propriété pour ajouter à la fonction objectif l'expression $-\lambda_{min}(Q)\sum_{i=1}^{n}(x_i^2 - x_i)$. Cette addition revient à soustraire aux termes diagonaux de Q sa plus petite valeur propre, notée ici $\lambda_{min}(Q)$, ce qui est suffisant pour obtenir une matrice semi-définie positive. Cette approche peut facilement être étendue aux problèmes en variables entières sous forme d'une convexification naïve que nous appelons `NC (Naive Convexification)`. Cette méthode consiste à ajouter à la fonction objectif $-\lambda_{min}(Q)\sum_{i=1}^{n}(x_i^2 - v_i)$ où $v_i = \sum_{k=0}^{u_i} k^2 r_{ik}$ et r_{ik} sont des variables bivalentes supplémentaires qui satisfont les contraintes $\sum_{k=0}^{u_i} r_{ik} = 1$ et $x_i = \sum_{k=0}^{u_i} k r_{ik}$. Cela nous fournit une première reformulation convexe de (QP).

Dans le cas où les variables sont bivalentes, Billionnet et Elloumi [3] améliorent la méthode de la plus petite valeur propre. Toujours en utilisant la propriété $x_i^2 = x_i$, ils introduisent un schéma de reformulations convexes qui modifie chaque terme diagonal de Q par un coefficient particulier en utilisant un paramètre vectoriel δ. Pour cela, ils ajoutent à la fonction objectif l'expression $\sum_{i=1}^{n}\delta_i(x_i^2 - x_i)$ qui s'annule lorsque x_i est une variable binaire. Puis, au sein de ce schéma, ils déterminent une convexification optimale en termes de borne obtenue par relaxation continue. Ils montrent que le paramètre vectoriel optimal δ peut être déduit de la solution optimale duale d'une relaxation semi-définie de (QP). Billionnet, Elloumi et Plateau étendent cette dernière méthode aux problèmes soumis à des contraintes d'égalité dans [4]. Ils perturbent non seulement les termes diagonaux, mais aussi les autres termes de Q grâce à un paramètre matriciel η. Ainsi, en plus de $\sum_{i=0}^{n}\delta_i(x_i^2 - x_i)$, ils ajoutent à la fonction objectif des fonctions quadratiques qui s'annulent sur le domaine de définition de (QP). Ces nouvelles fonctions sont obtenues en multipliant les contraintes linéaires d'égalité par les variables x_i. Concrètement, à partir d'une contrainte $\sum_{i=1}^{n} a_{ri}x_i - b_r$, ils construisent les fonctions nulles $\sum_{r=1}^{m}\eta_{rj}(\sum_{i=1}^{n} a_{ri}x_i x_j - b_r x_j)$. Cette méthode s'appelle `QCR (Quadratic Convex Reformulation)`. En pratique, pour des problèmes

soumis à des contraintes d'égalité, QCR est plus performante que la méthode décrite dans [3] puisqu'elle intègre les contraintes d'égalité au processus de convexification.

Dans cette thèse, nous présentons un schéma de reformulations convexes qui étend les idées de [3] [4] aux problèmes quadratiques en nombres entiers. En partant de NC décrite précédemment, nous introduisons de nouvelles variables y_{ij} et de nouvelles contraintes linéaires pour assurer l'égalité $y_{ij} = x_i x_j$. Ces nouvelles variables nous permettent de perturber chaque terme de la matrice Q par un coefficient particulier grâce à un paramètre matriciel β. Pour forcer l'égalité $y_{ij} = x_i x_j$ avec un nombre réduit de variables, nous choisissons la décomposition binaire des variables x_i plutôt que la décomposition unaire qui est utilisée dans NC. Nous utilisons en fait les idées de la linéarisation BIL. Nous utilisons également les contraintes d'égalité pour perturber la fonction objectif en utilisant un paramètre scalaire α. Ainsi, nous construisons la fonction quadratique $\alpha \sum_{r=1}^{m} (\sum_{i=1}^{n} a_{ri} x_i - b_r)^2$ qui s'annule sur le domaine de définition de (QP). Ensuite, nous présentons une projection du polyèdre associé à la relaxation continue du problème reformulé au sein de notre schéma de reformulation. Puis, au sein de ce schéma, nous déterminons une convexification optimale en termes de borne obtenue par relaxation continue. Nous montrons, en utilisant la propriété de projection, que les paramètres optimaux α et β peuvent être déduits de la solution optimale duale d'une relaxation demi-définie de (QP). De plus, la valeur de cette solution optimale est égale à la valeur optimale de la relaxation continue du problème reformulé. Nous appelons cette méthode IQCR (Integer Quadratic Convex Reformulation).

Ensuite, nous adaptons cette méthode à la programmation quadratique en variables mixtes entières dans une méthode appelée IQCRs. Cette adaptation permet également d'intégrer les contraintes d'inégalité dans la perturbation de la fonction objectif de notre schéma de reformulations convexes.

Puis, nous introduisons une reformulation convexe plus compacte, appelée CQCR (Compact Quadratic Convex Reformulation), qui est une restriction de IQCR. En effet, CQCR suit les mêmes idées que IQCR, mais ne perturbe que la diagonale de Q, avec un paramètre vectoriel β. Plus précisément, en plus des contraintes d'égalités, CQCR n'utilise que l'expression linéaire des carrés des variables pour perturber l'objectif. Cette restriction induit une réduction du nombre de produits à linéariser, et donc une réduction de la taille du problème reformulé, ce qui peut être avantageux expérimentalement.

Ensuite, nous présentons une interprétation pour la programmation quadratique en variables bivalentes des reformulations convexes que nous avons proposées. Il est facile de montrer que, dans ce cadre, la reformulation NC est équivalente à la méthode de la plus petite valeur propre et que la méthode CQCR est équivalente à la méthode QCR. Finalement, nous prouvons que la méthode IQCR appliquée à la programmation quadratique binaire fournit une borne par relaxation

continue supérieure ou égale à celle qui est fournie par la méthode QCR.

Par ailleurs, nous essayons de concurrencer la résolution par un solveur standard du problème reformulé. Pour cela, nous proposons un algorithme spécifique de Branch and Bound qui tire profit de la propriété de projection citée ci-dessus. A chaque noeud de l'arbre de recherche, la propriété de projection permet de calculer une borne inférieure en résolvant un programme quadratique convexe en variables mixtes entières de taille réduite.

Enfin, nous présentons de nombreuses expérimentations sur des instances de (QP). Nous testons et comparons les méthodes BBL, BBLr, BIL, BIL, NC, IQCR et CQCR, ainsi que notre algorithme spécifique de Branch and Bound sur des instances de (QP) de deux classes de problèmes. La première, la classe $EIQP$(Equality Integer Quadratic Program), concerne les problème soumis à une contrainte d'égalité, et la deuxième, la classe $IIQP$(Inequality Integer Quadratic Program), concerne les problèmes soumis à une contrainte d'inégalité. Ensuite nous évaluons la qualité de IQCR sur deux problèmes classiques de la programmation quadratique en variables bivalentes, un problème non contraint et le problème d'allocation de tâches.

Notre étude se présente de la façon suivante. Le Chapitre 1 présente un état de l'art de la programmation quadratique non convexe en variables purement en nombres entiers et en variables mixtes entières. Le Chapitre 2 présente les reformulations de la programmation quadratique en variables bivalentes utilisées dans cette thèse. Le Chapitre 3 détaille les reformulations linéaires BBL, BBLr, BIL, et BILr. Le Chapitre 4 présente les reformulations quadratiques convexes NC, IQCR, IQCRs, et CQCR. Le Chapitre 5 présente l'algorithme spécifique de Branch and Bound. Le Chapitre 6 présente les résultats expérimentaux.

Chapitre 1

Etat de l'art

1.1 La programmation quadratique non convexe en nombres entiers

A notre connaissance, il n'existe que très peu de littérature sur la programmation quadratique non convexe dont les variables appartiennent à l'ensemble des entiers naturels. Nous en avons sélectionné un qui traitait ce sujet, il s'agit d'un article de F. Körner [27]. Dans cet article, Körner traite le problème du sac à dos quadratique en variables entières que nous appelons ici (QK) :

$$(QK)\begin{cases} max & x^TQx + c^Tx \\ s.c. & a^Tx \leq b \\ & 0 \leq x_i \leq u_i \quad i = 1, \ldots, n \\ & x_i \ entier \quad i = 1, \ldots, n \end{cases}$$

Avec $Q \in \mathbf{S}_n$, $c \in \mathbb{R}^n$, $b \in \mathbb{N}_+$ et $a \in \mathbb{N}_+^n$.

Ce problème est une restriction du problème (QP) que nous considérons dans cette thèse puisque (QK) ne possède qu'une contrainte d'inégalité dont les coefficients a_i sont positifs. L'approche de Körner pour résoudre (QK) est celle qui est majoritairement utilisée pour contourner la non convexité. Il s'agit de calculer une borne supérieure sur la valeur de la solution entière qu'il faut ensuite intégrer à un algorithme de Branch and Bound. Pour déterminer une nouvelle borne pour (QK), Körner résout des problèmes de sacs à dos linéaires en utilisant la programmation dynamique. Il obtient ainsi une fonction de majoration linéaire de (QK) . Ainsi, dans son article il détermine d'abord une borne supérieure pour (QK) à l'aide de problèmes de sacs à dos linéaires. Puis, après avoir détaillé l'algorithme classiquement utilisé pour résoudre ces sacs à dos linéaires, il propose des améliorations de cet algorithme qui rendent plus efficace le calcul de cette nouvelle borne. Enfin, il détermine les règles de branchement nécessaires lors de

l'utilisation de cette borne au sein d'un algorithme de Branch and Bound. Nous présentons dans la suite le détail de son approche.

Une fonction de majoration linéaire

Détaillons comment Körner détermine une nouvelle borne en utilisant une fonction de majoration linéaire de (QK). Pour cela, il s'est inspiré de Gallo et al. [15].

En notant B l'ensemble des solutions réalisables de (QK) :

$$B = \{x \in \mathbb{R}^n \, : \, a^T x \leq b, \, 0 \leq x_i \leq u_i, \, i = 1, \ldots, n\},$$

comme $x \geq 0$,

$$\max_{x \in B}\{x^T Q x + c^T x\} \leq \max_{x \in B}\{x^T (z + c)\} \quad (1)$$

avec

$$z_i = \max_{x \in B}\{x^T q^i\} \quad (2)$$

et $q^i = (q_{i1}, \ldots, q_{in})$.

Ainsi, en choisissant l'ensemble z comme défini dans (2), la valeur de $\max_{x \in B}\{x^T (z + c)\}$ est une borne supérieure de (QK). Cette borne est utilisée dans l'algorithme de Branch and Bound de la façon suivante : à chaque noeud, un nouveau problème de la forme de (QK) est généré, ce qui permet de fixer au moins une variable encore libre. Cela modifie l'ensemble B et permet de calculer de nouvelles valeurs pour les variables de l'ensemble z et ainsi une nouvelle borne.

En notant r le nombre de variables libres de (QK) à un certain noeud de l'arbre, il faut calculer r valeurs de z_i, et résoudre $r + 1$ problèmes de sacs à dos linéaires pour déterminer la nouvelle borne supérieure. Cette approche est coûteuse en termes de quantité de calculs. L'idée de Körner est donc de réduire le nombre d'opérations à effectuer pour calculer cette borne.

Résolution classique du sac à dos linéaire

Classiquement, le problème du sac à dos linéaire se résout par programmation dynamique. Cette technique algorithmique permet de résoudre une catégorie particulière de problèmes d'optimisation sous contraintes. Elle a été désignée par ce terme pour la première fois dans les années 1940 par Bellman. Elle s'applique à des problèmes d'optimisation dont la fonction objectif se décrit comme la somme de fonctions monotones strictement croissantes des ressources. Elle s'appuie sur une relation entre la solution optimale du problème et celles d'un nombre fini

de sous-problèmes. Concrètement, cela signifie que l'on va pouvoir déduire la solution optimale d'un problème à partir d'une solution optimale d'un sous problème. Le problème du sac à dos possède la propriété de sous-structure optimale, c'est-à-dire que l'on peut construire la solution optimale du problème à i variables à partir du problème à $i-1$ variables. Cette propriété permet d'utiliser une méthode de résolution par programmation dynamique.

Présentons l'algorithme classique de programmation dynamique de résolution de problèmes de sac à dos linéaires. Cet algorithme va permettre de résoudre (1) et (2) et a été initié par Gilmore et Gomory, une description précise peut être trouvée dans [28]. Appelons (LK) le problème général du sac à dos linéaire :

$$(LK)\begin{cases} max & c^T x \\ s.c. & a^T x \leq b \\ & 0 \leq x_i \leq u_i \quad i = 1, \ldots, n \\ & x_i \ entier \quad i = 1, \ldots, n \end{cases}$$

Avec $c \in \mathbb{R}^n$, $b \in \mathbb{N}_+$ et $a \in \mathbb{N}_+^n$.

En définissant $M_i = \{0, 1, \ldots, u_i\}$, $i = 1, \ldots, n$, les équations de récurrence de la programmation dynamique sont les suivantes :

(a) $d_j(y) = -\infty$ pour $y < 0$ et $j = 1, \ldots, n$

(b) $d_n(y) = \max_{x_n \in M_n}\{c_n x_n \ : \ a_n x_n \leq y\}$ pour $y = 0, 1, \ldots, b$

(c) $d_j(y) = \max_{x_j \in M_j}\{c_j x_j + d_{j+1}(y - a_j x_j) \ : \ a_j x_j \leq y\}$ pour $y = 0, 1 \ldots, b$ et $j = n-1, \ldots, 1$

Avec cette formulation, nous avons bien (LK) qui est équivalent à $d_1(b)$.

On en déduit par le critère d'optimalité de Bellman que la valeur de $d_j(y)$ est la valeur optimale du problème de sac à dos linéaire suivant :

$$(3)\begin{cases} max & c_j x_j + \ldots + c_n x_n \\ s.c. & a_j x_j + \ldots + a_n x_n \leq y \\ & x_r \in M_r \ r = j, \ldots, n \end{cases}$$

Calculer les valeurs de $d_j(y)$ pour $y = 0, 1 \ldots, b$ et $j = n-1, \ldots, 1$ nécessite un nombre d'opérations en $O(nsb)$, avec $s = \max_{i=1,\ldots,n}\{u_i\}$.

En imposant $x_i \leq 1$, on peut simplifier la relation de récurrence (c) de la manière suivante :

$$(c') \qquad d_j(y) = \max\left\{d_{j+1}(y)\,,\, c_j + d_j(y - a_j)\right\}$$

et le calcul des $d_j(y)$ pour $y = 0, 1 \ldots, b$ et $j = n-1, \ldots, 1$ ne nécessite alors plus que $O(nb)$ opérations.

Calculs des bornes

Nous cherchons maintenant à déterminer des bornes de (QK) en résolvant (2). Rappelons que nous avons n problèmes du type de (LK) à résoudre pour calculer notre borne supérieure. Plus formellement, nous devons résoudre pour $i = 1, \ldots, n$, (LK) où $c = q^i$. Pour chacun de ces problèmes nous aurons à calculer $d_j(y)$ pour $y = 0, 1 \ldots, b$ et $j = n-1, \ldots, 1$. Nous notons donc $d_j^i(y)$ la valeur de $d_j(y)$ correspondant au problème où le vecteur $c = q^i$.

Supposons que les valeurs de z_i, $i = 1, \ldots, k$ soient fixées, et notons b^0 la valeur de y, nous avons alors :

$$z_i = d_{k+1}^i(b^0), \quad i = k+1, \ldots, n$$

Cela permet de résoudre les problèmes de sacs à dos linéaires (2) uniquement une fois au début de l'algorithme de Branch and Bound. Puis, pour obtenir une meilleure borne pendant le processus de Branch and Bound, il suffira de résoudre un seul problème de sac à dos linéaire.

Après avoir fixé la valeur de x_k à t (nous considérons ici que les variables $x_1, \ldots, x_{k-1}$ sont déjà fixées), notons c^0 la valeur de c. Nous avons à résoudre, pour $t = 0, 1, \ldots, min\{u_k, \lfloor b^0/a_k \rfloor\}$, le problème (4) suivant :

$$(4) \begin{cases} max & t(tq_{kk} + c_k^0) + (z_{k+1} + c_{k+1}^0 + 2tq_{k(k+1)})x_{k+1} + \ldots + (z_n + c_n^0 + 2tq_{kn})x_n \\ s.c. & a_{k+1}x_{k+1} + \ldots + a_n x_n \leq b^0 - ta_k \\ & x_r \in M_r \ r = k+1, \ldots, n \end{cases}$$

Le nombre de ces problèmes peut être considérable, c'est pour cela que Körner propose de plutôt résoudre les deux problèmes (5) et (6) suivants :

$$(5) \begin{cases} max & (z_{k+1} + c_{k+1}^0)x_{k+1} + \ldots + (z_n + c_n^0)x_n \\ s.c. & a_{k+1}x_{k+1} + \ldots + a_n x_n \leq b^0 \\ & x_r \in M_r \ r = k+1, \ldots, n \end{cases}$$

et

$$(6) \begin{cases} max & 2q_{k(k+1)}x_{k+1} + \ldots + 2q_{kn}x_n \\ s.c. & a_{k+1}x_{k+1} + \ldots + a_n x_n \leq b^0 - ta_k \\ & x_r \in M_r \ r = k+1, \ldots, n \end{cases}$$

La somme des valeurs optimales de (5) et (6) ne sera jamais plus petite que la valeur optimale de (4) pour $t = 0$. En notant $d'_j(y)$ les valeurs de la programmation dynamique du programme

(5) et $d''_j(y)$ celles de (6), nous avons pour tout $t = t_0$ entier, la valeur :

$$d'_{k+1}(b^0 - t_0 a_k y) + d''_{k+1}(b^0 - t_0 a_k) + t_0 \bar{q}_k(t_0)$$

qui est une borne supérieure de (4).

Règles de branchement de l'algorithme des Branch and Bound

Nous avons maintenant une borne qui se calcule en un temps plus raisonnable. Il reste à déterminer les règles de branchements pour l'algorithme de Branch and Bound. L'ordre des variables avec lesquelles l'algorithme va brancher est déterminé au début de l'algorithme. L'idéal est de choisir d'abord une variable qui n'autorise qu'un petit nombre de branchements. Körner utilise l'heuristique qui consiste à calculer les valeurs :

$$t_k = \frac{\left| |q_{kk}| q_{kk} + 2 \sum_{\substack{i=1 \\ i \neq k}}^{n} q_{ik} + c_k \right|}{\min\{u_k, \lfloor b^0 / a_k \rfloor\}}$$

et les ordonner selon leurs tailles,

$$t_{k1} \geq t_{k2} \geq \ldots \geq t_{kn}$$

Les variables sont donc choisies selon l'ordre x_{k1} avant x_{k2}.

Cette approche de résolution de (QK) est en fait une amélioration de l'idée de Gallo et al. [15]. Elle permet de réduire considérablement la quantité de calculs pour résoudre (QK). L'article se termine sur une évaluation expérimentale de cette approche. Körner génère aléatoirement deux types d'instances du problème du sac à dos quadratique. La première considère des bornes supérieures sur les variables égales à 1, et la deuxième supprime les contraintes de bornes en considérant qu'elles sont intégrées dans la contrainte d'inégalité. Ces instances vont jusqu'à une taille de 30 variables binaires et 25 variables entières, et sont résolues pour les tailles 25 et 30 en un temps moyen de 6.3 secondes et 281.3 secondes respectivement. En suivant les indications de Körner qui décrivent son générateur d'instances, il nous a été impossible de retrouver des

valeurs de solutions comparables, nous n'avons donc pas pu nous comparer avec Körner pour le cas particulier où (QP) est du type (QK). Cependant, nous avons générés nos instances selon les mêmes idées.

Même si certaines idées de Körner, comme la majoration par des fonctions linéaires, peuvent se généraliser, cette méthode de résolution se restreint aux problèmes ayant une contrainte d'inégalité, elle ne couvre donc pas tout le panel d'instances générés par la formulation (QP). Nous proposons maintenant d'exposer les méthodes de résolutions de la programmation quadratique en variables mixtes, qui peuvent s'appliquer à (QP), puisqu'il s'agit de problèmes qui généralisent (QP).

1.2 La programmation quadratique non convexe en variables mixtes

Dans la littérature, il existe plusieurs méthodes de résolution de problèmes non convexes en variables mixtes (MINLP : Mixed Integer Non Linear Programming). Ce type de problèmes n'est pas exactement celui que nous traitons dans cette thèse, mais (QP) est une restriction des MINLP, puisque c'est un MINLP qui optimise une fonction quadratique soumise à des contraintes linéaires, où l'ensemble des variables réelles est l'ensemble vide. Plus formellement, un MINLP s'écrit de la façon suivante :

$$(MINLP) \begin{cases} min & f(x) \\ s.c. & g(x) \leq b \\ & x \in S \end{cases}$$

où $f : \mathbb{R}^n \to \mathbb{R}$, et $g : \mathbb{R}^n \to \mathbb{R}^m$, $b \in \mathbb{R}^m$, et $S := \mathbb{R}^{n-k} \times \mathbb{Z}^k$.

Les algorithmes de résolution des MINLP s'appliquent à des problèmes bien plus généraux que (QP) et ne sont donc pas très adaptés à la résolution de (QP). Les approches conventionnelles qui résolvent ce type de problèmes comprennent des méthodes heuristiques et des techniques d'optimisation globale. Les méthodes heuristiques consistent en général à réduire au maximum les effets de la non convexité, et donc le fait d'avoir plusieurs minimums locaux. Cependant, ce type d'approche ne fournit pas forcément la solution optimale exacte. L'autre approche consiste à appliquer à $(MINLP)$ un Branch and Bound. Le succès de ce type d'approche dépend des méthodes de branchement appliquées, qui doivent impérativement améliorer la borne pour chaque sous problème [30, 1, 40, 12].

Nous nous nous attardons ici sur une approche récente de Saxena et al. [37] qui consiste à résoudre $(MINLP)$ dans le cas particulier où la fonction objectif $f(x)$ est linéaire et où ses

contraintes $g(x)$ sont quadratiques et non convexes. Celle-ci consiste à générer des relaxations convexes serrées de $(MINLP)$ en utilisant la programmation semi-définie. Un rappel sur la programmation semi-définie est détaillé dans l'Annexe A. Dans [37], l'idée est de reformuler $(MINLP)$ en un problème équivalent $(MINLP')$ ayant une contrainte semi-définie :

$$(MINLP') \begin{cases} min & f(x,Y) \\ s.c. & g(x,Y) \leq b \\ & Y = xx^T \\ & x \in S \end{cases}$$

Chaque produit de variables est remplacé par un terme de la matrice variable Y. L'avantage de cette reformulation est qu'il ne reste que les contraintes d'intégrité et la contrainte $Y = xx^T$ qui ne sont pas convexe. Classiquement, on relâche les contraintes d'intégrité, puis on relâche la contrainte $Y = xx^T$ par la paire d'inégalités :

$$Y - xx^T \succeq 0$$

$$0 \succeq Y - xx^T$$

La première est une contrainte convexe, alors que la deuxième est concave. L'idée de Saxena et al. [37] est donc de remplacer cette deuxième contrainte par des coupes disjonctives calculées à partir des valeurs propres négatives de la matrice $Y - xx^T$, et de leurs vecteurs propres associés. De plus, ils utilisent la contrainte convexe $Y - xx^T \succeq 0$ pour dériver des coupes quadratiques et convexes et enfin, ils combinent ces deux approches dans un algorithme de plans coupants.

Les auteurs présentent ensuite des tests numériques de leur approche sur des instances de la librairie GLOBALLib [17]. Comme dans cette librairie il y a très peu d'instances dont le degré ne dépasse pas 2, ils ont linéarisé les termes polynômiaux pour appliquer leur approche. Les résultats montrent que leur approche fournit une borne obtenue par relaxation continue de bonne qualité, mais qu'elle est très lourde en terme de coupes ajoutées et donc en termes temps de calcul.

Le principal problème pour appliquer les méthodes de résolution de la programmation quadratique en variables mixtes à (QP) est que ces approches sont très générales et donc souvent moins pertinentes dans le cas particulier qu'est (QP). Nous aurions aimé nous comparer numériquement avec la méthode de Saxena et al., cependant la librairie d'instances qu'ils utilisent [17]

n'a quasiment aucune instance purement en nombres entiers.

Nous présentons dans la suite de cette thèse les principales reformulations de la programmation quadratique en variables bivalentes qui ont inspiré nos méthodes de résolutions pour la programmation quadratique en variables entières.

Chapitre 2

Certaines reformulations de la programmation quadratique en variables 0-1 utilisées dans cette thèse

Dans ce chapitre, nous présentons différentes reformulations d'un programme quadratique en variables binaires soumis à des contraintes linéaires. Un tel problème consiste à minimiser une fonction quadratique quelconque, sous les contraintes que les variables du problèmes doivent prendre des valeurs dans $\{0,1\}$ et satisfaire les contraintes linéaires qui lui sont associées (égalités et/ou inégalités). Plus formellement, ce problème est de la forme suivante :

$$(QP^{01}) \begin{cases} Min & f(x) = \sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij}x_ix_j + \sum_{i=1}^{n} c_ix_i & & \\ s.c. & \sum_{i=1}^{n} a_{ri}x_i = b_r & r \in R & (1) \\ & \sum_{i=1}^{n} d_{si}x_i \leq e_s & s \in S & (2) \\ & x_i \in \{0,1\} & i \in I & (3) \end{cases}$$

Sans perte de généralité, nous supposons que la matrice Q est symétrique. Si ce n'est pas le cas, Q peut se réécrire sous la forme $\frac{Q+Q^T}{2}$. (QP^{01}) appartient à la classe des problèmes $\mathcal{NP}$-difficiles. Les solveurs standards peuvent résoudre (QP^{01}) dans le cas particulier où la fonction $f(x)$ est convexe, et depuis sa dernière version, le solveur Cplex [24] peut résoudre les problèmes non convexes en variables binaires, mais de façon peu efficace. Une façon de résoudre (QP^{01}) est donc de le transformer en un programme équivalent dont la fonction objectif est convexe. On peut choisir de reformuler (QP^{01}) soit en un programme dont la fonction objectif est linéaire, soit en un programme dont la fonction objectif est quadratique et convexe. Dans

ce chapitre, nous présentons différentes méthodes de reformulation exacte de (QP^{01}) en des programmes linéaires ou quadratiques et convexes.

2.1 Reformulations linéaires de programmes quadratiques en variables 0-1

Une reformulation linéaire consiste à trouver un programme linéaire en variables mixtes qui soit équivalent au problème initial. L'idée est de remplacer chaque produit de variables $x_i x_j$ par une variable additionnelle y_{ij} et d'ajouter un ensemble d'inégalités linéaires qui forcent l'égalité $x_i x_j = y_{ij}$.

Une reformulation linéaire du produit de deux variables binaires, Fortet, 1959

La linéarisation du produit de deux variables binaires a été proposée par Fortet [13]. Elle consiste à remplacer chaque produit de variables binaires $x_i x_j$ par une nouvelle variable réelle y_{ij} et d'ajouter l'ensemble d'inégalités suivant :

$$L_{xy} \begin{cases} y_{ii} = x_i & i \in I & (4) \\ y_{ij} \leq x_i & (i,j) \in I^2 & (5) \\ y_{ij} \leq x_j & (i,j) \in I^2 & (6) \\ y_{ij} \geq x_i + x_j - 1 & (i,j) \in I^2 & (7) \\ y_{ij} \geq 0 & (i,j) \in I^2 & (8) \\ y_{ij} = y_{ji} & (i,j) \in I^2 & (9) \end{cases}$$

En considérant les valeurs possibles des variables x_i et x_j, il est facile de vérifier que les contraintes (4)-(9) forcent l'égalité entre les produits $x_i x_j$ et les variables y_{ij}. On obtient donc le programme en variables binaires (QP_L), équivalent à (QP^{01}), dont la fonction objectif est linéaire :

$$(QP_L) \begin{cases} Min & f_L(x,y) = \sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij} y_{ij} + \sum_{i=1}^{n} c_i x_i \\ s.c. & (1)(2)(3) \\ & x, y \in L_{xy} \end{cases}$$

La méthode RLT, Sherali, Adams, 1995

Dans [38], Sherali et Adams améliorent la valeur de la borne obtenue par relaxation continue du programme reformulé (QP_L). Pour cela, ils montrent que les inégalités (10)-(12) obtenues en

multipliant les contraintes d'égalités (1) par les variables x_j, et en multipliant les contraintes d'inégalités (2) par les variables x_j et leur complémentaire $(1 - x_j)$ sont valides. Il est ensuite simple de linéariser ces nouvelles contraintes en remplaçant par les variables y_{ij} chaque nouveau produit $x_i x_j$ introduit. Cette approche est en fait un renforcement par un ajout de coupes de la linéarisation de Fortet. La méthode RLT de niveau 1 fournit le programme (QP_{RLT}) équivalent à (QP^{01}), qui est le suivant :

$$(QP_{RLT}) \begin{cases} Min & f_L(x,y) = \sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij} y_{ij} + \sum_{i=1}^{n} c_i x_i & & \\ s.c. & (1)(2)(3) & & \\ & x, y \in L_{xy} & & \\ & \sum_{i=1}^{n} a_{ri} y_{ij} = b_r x_j & j \in I, r \in R & (10) \\ & \sum_{i=1}^{n} d_{si} y_{ij} \leq e_s x_j & j \in I, s \in S & (11) \\ & \sum_{i=1}^{n} d_{si}(x_i - y_{ij}) \leq e_s(1 - x_j) & j \in I, s \in S & (12) \end{cases}$$

Une reformulation linéaire du produit d'une variable binaire par une variable continue, McCormick, 1976

La linéarisation du produit d'une variable binaire par une variable réelle a été proposée par McCormick [31]. Elle consiste à remplacer chaque produit $t_i x_j$ où $t_i \in \{0,1\}$ et $x_j \in \mathbb{R}$, avec $0 \leq x_j \leq u_j$, par une nouvelle variable réelle z_{ij} et l'ensemble d'inégalités suivant :

$$\begin{cases} z_{ij} \leq t_i u_j & (i,j) \in I^2 & (13) \\ z_{ij} \leq x_j & (i,j) \in I^2 & (14) \\ z_{ij} \geq x_j - u_j(1 - t_i) & (i,j) \in I^2 & (15) \\ z_{ij} \geq 0 & (i,j) \in I^2 & (16) \end{cases}$$

En considérant les valeurs possibles de t_i, il est facile de vérifier que les contraintes (13)-(16) forcent l'égalité entre les produits $t_i x_j$ et les variables z_{ij}.

2.2 Reformulations quadratiques convexes de programmes quadratiques en variables 0-1

Nous nous intéressons maintenant aux méthodes existantes de convexifications. Une convexification consiste en la définition d'un problème équivalent au problème initial, dont la fonction objectif est quadratique et convexe, i.e., par exemple, son Hessien est semi-défini positif. Plusieurs méthodes de convexifications ont été proposées dans la littérature. Elles s'appliquent toutes à des programmes quadratiques en variables binaires. Dans la suite de cette section, nous présentons les reformulations convexes de la littérature utilisées dans cette thèse.

2.2.1 La méthode de la plus petite valeur propre, Hammer et Rubin, 1970

Cette méthode a été introduite par Hammer et Rubin dans les années 1970 [19]. Elle consiste à réécrire $f(x)$ en utilisant à la fois les contraintes d'intégrité de (QP^{01}) et le calcul des valeurs propres de Q. En effet, la propriété $x_i^2 = x_i$, permet de réécrire facilement la fonction $f(x)$, en perturbant la diagonale de Q avec un paramètre réel constant σ :

$$\begin{aligned} f_\sigma(x) &= \sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij}x_ix_j + \sum_{i=1}^{n} c_ix_i + \sigma\sum_{i=1}^{n}(x_i^2 - x_i) \\ &= x^TQ_\sigma x + c_\sigma^T x \end{aligned}$$

Avec $Q_\sigma = Q + diag(\sigma)$ et $c_\sigma = c - \sigma$.
Il est clair que $f_\sigma(x) = f(x)$, si $x \in \{0,1\}^n$. On peut donc construire une famille de fonctions $f_\sigma(x)$ toutes égales à $f(x)$. Il est intéressant de choisir le paramètre σ, telle que Q_σ soit semi-définie positive. Ils montrent d'abord que pour tout $\sigma \geq -\lambda_{min}(Q)$, la fonction $f_\sigma(x)$ est convexe. Ensuite, comme (QP^{01}) est un problème de minimisation, il est intéressant de choisir σ tel que la valeur de la relaxation continue du problème reformulé soit la plus grande possible. Hammer et Rubin montrent que comme

$$\begin{array}{ll} \min & \{f_\sigma(x)\} \\ {\scriptstyle (1)(2)} & \\ {\scriptstyle x_i \in \{0,1\}^n} & \end{array}$$

est une fonction décroissante en σ, si $\sigma = -\lambda_{min}(Q)$, alors la valeur optimale de la relaxation continue de (QP^{01}) est la plus grande possible.

Et on obtient un nouveau programme $(QP_{\texttt{EV}})$

$$(QP_{\texttt{EV}}) \begin{cases} Min & f_{\texttt{EV}}(x) = f(x) - \sum_{i=1}^{n}\lambda_{min}(Q)(x_i^2 - x_i) \\ s.c. & (1)(2)(3) \end{cases}$$

avec $(QP_{\texttt{EV}})$ qui est un programme quadratique et convexe.

Cette approche de résolution nous permet de donner directement le problème reformulé $(QP_{\texttt{EV}})$ à un solveur MIQP, qui est maintenant capable d'en résoudre la relaxation continue en un temps polynômial, puis de le résoudre par un algorithme de Branch and Bound. Il est connu que le comportement de tels algorithmes dépend de la borne fournie à la racine de l'arbre de résolution qui est la valeur de la relaxation continue de $(QP_{\texttt{EV}})$. En effet, plus la solution optimale en nombres réels d'un problème discret est proche de sa solution optimale en nombres entiers, plus l'arbre de résolution de l'algorithme de Branch and Bound aura des chances d'être petit, et donc plus le temps de résolution sera court. Ici la valeur de la relaxation continue de $(QP_{\texttt{EV}})$ est trop éloignée de sa solution optimale entière. C'est pourquoi plusieurs améliorations de ce principe de convexification ont été proposées, notamment pour les problèmes quadratiques en variables booléennes non contraints et pour les problèmes quadratiques en variables booléennes soumis à des contraintes d'égalité. Nous présentons ces algorithmes dans la suite de cette section.

2.2.2 La méthode `UQCR (Unconstrained Quadratic Convex Reformulation)`, Billionnet et Elloumi, 2007

Cette reformulation utilise le même principe que celle de Hammer et Rubin, c'est-à-dire qu'elle consiste toujours à trouver un problème équivalent à (QP^{01}) avec une fonction objectif convexe. L'idée est donc, toujours en utilisant la propriété que $x_i^2 = x_i$, de perturber le Hessien de la fonction objectif $f(x)$ en lui ajoutant des termes nuls sur le domaine de définition de (QP^{01}). Billionnet et Elloumi améliorent la méthode de la plus petite valeur propre, en perturbant chaque terme diagonal de la matrice Q par un paramètre de valeur différente en utilisant un paramètre vectoriel δ. Ils calculent ensuite les valeurs de ce paramètre vectoriel qui rendent convexe $f(x)$, tout en maximisant la valeur optimale de la relaxation continue du problème reformulé. Cette approche est appelée ici `UQCR`. Ainsi, ils transforment la fonction $f(x)$ de la façon suivante :

$$\begin{aligned}
\forall\, x \in \{0,1\}, \quad f(x) &= \sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij}x_ix_j + \sum_{i=1}^{n} c_ix_i \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij}x_ix_j + \sum_{i=1}^{n} c_ix_i + \sum_{i=1}^{n} \delta_i(x_i^2 - x_i) \\
&= x^T Q_{\texttt{UQCR}} x + c_{\texttt{UQCR}}^T x \\
&= f_{\texttt{UQCR}}(x)
\end{aligned}$$

Avec $\delta \in \mathbb{R}^n$, et $Q_{\texttt{UQCR}} = Q + diag(\delta)$:

$$Q_{\texttt{UQCR}} = \begin{pmatrix} q_{11}+\delta_1 & q_{12} & \dots & q_{1n} \\ q_{21} & q_{22}+\delta_2 & \dots & q_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ q_{n1}+ & q_{n2} & \dots & q_{nn}+\delta_n \end{pmatrix}$$

et $c_{\texttt{UQCR}} = c - \delta$, et donc :

$$c = (c_1 - \delta_1, c_2 - \delta_2, \dots, c_n - \delta_n)$$

Et ils obtiennent un nouveau programme $(QP_{\texttt{UQCR}})$

$$(QP_{\texttt{UQCR}}) \begin{cases} Min & f_{\texttt{UQCR}}(x) = f(x) + \sum_{i=1}^{n} \delta_i (x_i^2 - x_i) \\ s.c. & (1)(2)(3) \end{cases}$$

Il existe des valeurs de δ telles que $(QP_{\texttt{UQCR}})$ soit un programme quadratique et convexe. Prenons par exemple ce qui revient à la méthode de la plus petite valeur propre citée ci-dessus : $\delta_i = -\lambda_{min}\ \forall i \in I$.

Billionnet et Elloumi cherchent à résoudre le programme suivant :

$$(CP_{\texttt{UQCR}}) : \{ \max_{\delta \in \mathbb{R}^n, Q_{\texttt{UQCR}} \succeq 0} \{ v(\overline{QP}_{\texttt{UQCR}}) \}$$

Avec $v(\overline{QP}_{\texttt{UQCR}})$ la valeur optimale $(\overline{QP}_{\texttt{UQCR}})$. Billionnet et Elloumi montrent qu'une solution optimale de $(CP_{\texttt{UQCR}})$ est donnée par la solution optimale duale d'une relaxation semi-définie de $(QP_{\texttt{UQCR}})$. Plus formellement, ils prouvent le théorème suivant :

Théorème 2.2.1 (Billionnet et Elloumi, 2007) *Soit* $(SDP_{\texttt{UQCR}})$ *le programme semi-défini suivant :*

$$(SDP_{\texttt{UQCR}}) \begin{cases} Min & f(X,x) = \sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij} X_{ij} + \sum_{i=1}^{n} c_i xi & \\ s.c & (1)(2) & \\ & X_{ii} = x_i \qquad i \in I & (17) \\ & \begin{pmatrix} 1 & x \\ x^T & X \end{pmatrix} \succeq 0 & (18) \\ & x \in \mathbb{R}^n \quad X \in \mathbf{S}^n & (19) \end{cases}$$

Une solution optimale δ^* de $(CP_{\texttt{UQCR}})$ peut être déduite de la valeur optimale des variables duales de $(SDP_{\texttt{UQCR}})$. Les valeurs optimales de δ^* sont les valeurs optimales des variables duales associées aux contraintes (17).

Du théorème 2.2.1, Billionnet et Elloumi déduisent l'algorithme de résolution de programmes quadratiques en variables binaires qui est présenté dans la Figure 2.1.

`UQCR (Unconstrained Quadratic Convex Reformulation)`
1. Résoudre le programme semi-défini $(SDP_{\texttt{UQCR}})$
2. Déduire δ^* (théorème 2.2.1)
3. Résoudre le programme quadratique et convexe $(QP_{\texttt{UQCR}})$ pour $\delta = \delta^*$

FIG. 2.1 – Algorithme de résolution de (QP^{01}) basé sur le reformulation `UQCR`

Cette reformulation améliore la qualité de la borne obtenue par relaxation continue. Billionnet, Elloumi et Plateau ont étendu cette approche aux problèmes soumis à des contraintes d'égalité dans la méthode `QCR (Quadratic Convex Reformulation)`

2.2.3 La méthode `QCR (Quadratic Convex Reformulation)`, Billionnet, Elloumi et Plateau, 2009

Cette reformulation est une extension de `UQCR`. Elle utilise également les contraintes d'intégrité pour convexifier la fonction objectif à l'aide d'un paramètre vectoriel δ, mais outre la diagonale, elle perturbe les autres termes du Hessien en utilisant les contraintes d'égalité et un paramètre matriciel η. Plus précisément, cette méthode consiste à ajouter à l'objectif les fonctions suivantes, qui s'annulent sur le domaine de définition de (QP^{01}) :

(i) $\sum_{i=1}^{n} \delta_i (x_i^2 - x_i)$, $\delta \in \mathbb{R}^n$.

(ii) $\sum_{r=1}^{m} (\sum_{i=1}^{n} \eta_{ri} x_i)(\sum_{j=1}^{n} a_{rj} x_j - b_r)$, $\eta \in \mathbf{M}_{m,n}$

En considérant les contraintes (1) et (3), il est clair que les fonctions (i) et (ii) sont nulles sur le domaine de définition de (QP^{01}). Ainsi, Billionnet, Elloumi et Plateau transforment la

fonction $f(x)$ de la façon suivante :

$$\begin{aligned} f(x) &= \sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij}x_ix_j + \sum_{i=1}^{n} c_ix_i \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij}x_ix_j + \sum_{i=1}^{n} c_ix_i + \sum_{i=1}^{n}\delta_i(x_i^2 - x_i) + \sum_{r=1}^{m}(\sum_{i=1}^{n}\eta_{ri}x_i)(\sum_{j=1}^{n}a_{rj}x_j - b_r) \\ &= x^T Q_{\mathtt{QCR}} x + c_{\mathtt{QCR}}^T x \\ &= f_{\mathtt{QCR}}(x) \end{aligned}$$

Avec $\eta \in \mathbf{M}_{m,n}$ et $\delta \in \mathbb{R}^n$, et $Q_{\mathtt{QCR}} = Q + 1/2(\eta^T A + A^T\eta) + diag(\delta)$:

$$Q_{\mathtt{QCR}} = \begin{pmatrix} q_{11} + \sum_{r=1}^{m}\eta_{r1}a_{r1} + \delta_1 & \dots & \dots & q_{1j} + \sum_{r=1}^{m}\eta_{r1}a_{rj} & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \dots & q_{ii} + \sum_{r=1}^{m}\eta_{ri}a_{ri} + \delta_i & \dots & q_{ij} + \sum_{r=1}^{m}\eta_{ri}a_{rj} & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \dots & \dots & \dots & \dots & q_{nn} + \sum_{r=1}^{m}\eta_{rn}a_{rn} + \delta_n \end{pmatrix}$$

et $c_{\mathtt{QCR}} = c - \delta - \eta^T b$.

Et ils obtiennent un nouveau programme $(QP_{\mathtt{QCR}})$

$$(QP_{\mathtt{QCR}}) \begin{cases} Min & f_{\mathtt{QCR}}(x) = f(x) + \sum_{i=1}^{n}\delta_i(x_i^2 - x_i) + \sum_{r=1}^{m}(\sum_{i=1}^{n}\eta_{ri}x_i)(\sum_{j=1}^{n}a_{rj}x_j - b_r) \\ s.c. & (1)(2)(3) \end{cases}$$

Il est possible de choisir les valeurs du paramètre matriciel η et du paramètre vectoriel δ tels que $(QP_{\mathtt{QCR}})$ soit un programme quadratique et convexe. Prenons par exemple $\delta_i = -\lambda_{min}\ \forall i \in I$ et $\eta_{ri} = 0\ \forall i \in I,\ \forall r \in R$. Comme dans la convexification précédente ils s'intéressent aux valeurs des paramètres η_{ri} et δ_i qui à la fois rendent convexe $(QP_{\mathtt{QCR}})$ mais aussi qui maximisent sa relaxation continue. Soit $(\overline{QP}_{\mathtt{QCR}})$ la relaxation continue de $(QP_{\mathtt{QCR}})$, ils cherchent donc à résoudre le programme suivant :

$$(CP_{\mathtt{QCR}}) : \{ \max_{\eta\in\mathbb{R}^{mn},\delta\in\mathbb{R}^n, Q_{\mathtt{QCR}}\succeq 0} \{v(\overline{QP}_{\mathtt{QCR}})\}$$

Billionnet, Elloumi et Plateau montrent qu'une solution optimale de $(CP_{\mathtt{QCR}})$ est donnée par la solution optimale duale d'une relaxation semi-définie de $(QP_{\mathtt{QCR}})$. Plus formellement, ils

prouvent le théorème suivant :

Théorème 2.2.2 (Billionnet, Elloumi et Plateau, 2008) *Soit* $(SDP_{\texttt{QCR}})$ *le programme semi-défini suivant :*

$$(SDP_{\texttt{QCR}}) \begin{cases} Min & f(X,x) = \sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij}X_{ij} + \sum_{i=1}^{n} c_i x_i \\ s.c. & (1)(2) \\ & (17)(18)(19) \\ & -b_r x_i + \sum_{j=1}^{n} a_{rj}X_{ij} = 0 \quad i \in I, r \in R \quad (20) \end{cases}$$

Une solution optimale (η^*, δ^*) *de* $(CP_{\texttt{QCR}})$ *peut être déduite de la valeur optimale des variables duales de* $(SDP_{\texttt{QCR}})$*. Les valeurs optimales de* η^* *sont les valeurs optimales des variables duales associées aux contraintes (20) et les valeurs optimales de* δ^* *sont les valeurs optimales des variables duales associées aux contraintes (17).*

Du théorème 2.2.2, ils déduisent l'algorithme de résolution de programmes quadratiques en variables binaires qui est présenté dans la Figure 2.2.

`QCR (Quadratic Convex Reformulation)`
1. Résoudre le programme semi-défini $(SDP_{\texttt{QCR}})$
2. Déduire (η^*, δ^*) (théorème 2.2.2)
3. Résoudre le programme quadratique et convexe $(QP_{\texttt{QCR}})$ pour $\eta = \eta^*$ et $\delta = \delta^*$

FIG. 2.2 – Algorithme de résolution de (QP^{01}) basé sur le reformulation `QCR`

Nous allons dans la suite présenter plusieurs nouvelles approches de reformulation convexe de programmes quadratiques en variables entières. Ainsi, nous présentons dans le Chapitre 3 des reformulations linéaires, puis dans le Chapitre 4 des reformulations quadratiques convexes.

Chapitre 3

Reformulations linéaires de programmes quadratiques en nombres entiers

3.1 Introduction et exemple de base

Dans ce chapitre, nous présentons plusieurs reformulations linéaires de programmes quadratiques en variables entières de la forme de (QP) :

$$(QP)\begin{cases} Min & f(x) = \sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij}x_ix_j + \sum_{i=1}^{n} c_ix_i & & \\ s.c. & \sum_{i=1}^{n} a_{ri}x_i = b_r & r \in R & (1) \\ & \sum_{i=1}^{n} d_{si}x_i \leq e_s & s \in S & (2) \\ & x_i \leq u_i & i \in I & (3) \\ & x_i \geq 0 & i \in I & (4) \\ & x_i \in \mathbb{N} & i \in I & (5) \end{cases}$$

où $I = \{i : i = 1, \ldots, n\}$, $R = \{r : r = 1, \ldots, m\}$, et $S = \{s : s = 1, \ldots, p\}$.

Une reformulation linéaire consiste à construire un nouveau programme en nombres entiers équivalent à (QP) dont la fonction objectif est linéaire. La plupart des linéarisations de la littérature ont été conçues pour des programmes quadratiques en variables binaires. L'approche la plus naturelle pour résoudre (QP) est donc dans une première phase de le reformuler en un programme équivalent en variables binaires, puis, dans une deuxième phase, de le linéariser par des méthodes connues [13] [38] décrites dans le Chapitre 2. Cependant, cette approche que

nous appelons `BBL (Binary Binary Linearization)`, fournit un programme reformulé avec un grand nombre de variables et de contraintes.

Nous présentons donc une nouvelle approche, `BIL (Binary Integer Linearization)`, qui se décompose également en deux phases. La première consiste à reformuler (QP) en un programme quadratique particulier, où chaque terme quadratique est le produit d'une variable binaire par une variable entière. Et la deuxième consiste à linéariser ces termes quadratiques. Comme le nombre de termes quadratiques est plus petit que dans l'approche `BBL`, le nombre de variables et contraintes ajoutées lors de la linéarisation est réduit. Le programme obtenu par la méthode `BIL` est donc significativement plus petit que celui obtenu dans l'approche `BBL`.

Chacune de ces deux reformulations fournit un programme linéaire qui est équivalent à (QP) que nous renforçons en ajoutant des inégalités valides.

Dans ce chapitre, nous présentons d'abord les approches `BBL` et `BIL`. Ensuite, nous renforçons `BBL` et `BIL` par des inégalités valides. Nous notons ces reformulations renforcées `BBLr` et `BILr`. Enfin, nous présentons une comparaison théorique de `BBL` et `BIL` du point de vue de la borne obtenue par relaxation continue.

Nous illustrons chacune des quatre reformulations linéaires, sur le petit exemple suivant comportant 4 variables, 1 contrainte d'égalité et 1 contrainte d'inégalité :

Exemple 3.1.1

$$(QP_e)\begin{cases} Min & f(x) = x^T \begin{pmatrix} 5 & -7 & -6 & -1 \\ -7 & 3 & -8 & -18 \\ -6 & -8 & -17 & 10 \\ -1 & -18 & 10 & 3 \end{pmatrix} x + \begin{pmatrix} -5 \\ -11 \\ 4 \\ 1 \end{pmatrix} x \\ s.c. & 3x_1 + 19x_2 + 18x_3 + 11x_4 = 255 \\ & 11x_1 + 13x_2 + 8x_3 + x_4 \leq 165 \\ & 0 \leq x_i \leq 10 & i \in \{1, \ldots, 4\} \\ & x_i \in \mathbb{N} & i \in \{1, \ldots, 4\} \end{cases}$$

La valeur de la solution optimale entière de (QP_e) *est* -2552.

3.2 Une reformulation basée sur la linéarisation du produit de deux variables binaires : `BBL (Binary Binary Linearization)`

Cette linéarisation consiste à remplacer chaque variable entière x_i par sa décomposition en variables binaires. Ensuite, chaque produit de variables entières devient une somme pondérée de produits de variables binaires qu'il est possible de linéariser par la méthode de Fortet [19]

présentée dans le Chapitre 2. Dans cette section, nous présentons d'abord la méthode `BBL`, puis `BBLr` la version renforcée de `BBL`.

3.2.1 La méthode `BBL`

Dans cette méthode, nous remplaçons les variables x_i par l'ensemble des variables binaires t_{ik}. Comme x_i est une variable entière comprise entre 0 et u_i, nous introduisons $\lfloor log(u_i) \rfloor + 1$, variables binaires qui satisfont $x_i = \sum_{k=0}^{\lfloor log(u_i) \rfloor} 2^k t_{ik}$, l'unique décomposition en puissances de 2 de la variable x_i. Ainsi, chaque produit $x_i x_j$ devient une expression de produits $t_{ik} t_{jl}$, où $x_j = \sum_{l=0}^{\lfloor log(u_j) \rfloor} 2^l t_{jl}$, que nous linéarisons en ajoutant de nouvelles variables binaires w_{ikjl}. Nous obtenons le programme linéaire suivant :

$$(LP_{\texttt{BBL}}) \begin{cases} Min & f_{\texttt{BBL}}(x,y) = \sum_{i=1}^{n} \sum_{\substack{j=1 \\ q_{ij} \neq 0}}^{n} q_{ij} y_{ij} + \sum_{i=1}^{n} c_i x_i \\ s.c. & (1)(2) \\ & x, y, w, t \in P_{xywt}^{BBL} \end{cases}$$

Avec

$$P_{xywt}^{BBL} \begin{cases} & (3)(4) & & \\ & x_i = \sum_{k=0}^{\lfloor log(u_i) \rfloor} 2^k t_{ik} & i \in I & (6) \\ & y_{ij} = \sum_{k=0}^{\lfloor log(u_i) \rfloor} \sum_{l=0}^{\lfloor log(u_j) \rfloor} 2^{k+l} w_{ikjl} & (i,j) \in I^2 & (7) \\ x,y,w,t: & w_{ikjl} \leq t_{ik} & ((i,k),(j,l)) \in E^2,\ q_{ij} < 0 & (8) \\ & w_{ikjl} \leq t_{jl} & ((i,k),(j,l)) \in E^2,\ q_{ij} < 0 & (9) \\ & w_{ikjl} \geq t_{ik} + t_{jl} - 1 & ((i,k),(j,l)) \in E^2,\ q_{ij} > 0 & (10) \\ & w_{ikjl} \geq 0 & ((i,k),(j,l)) \in E^2,\ q_{ij} > 0 & (11) \\ & w_{ikik} = t_{ik} & (i,k) \in E,\ q_{ii} \neq 0 & (12) \\ & w_{ikjl} = w_{jlik} & ((i,k),(j,l)) \in E^2,\ i < j\ q_{ij} \neq 0 & (13) \\ & w_{ikil} = w_{ilik} & ((i,k),(i,l)) \in E^2,\ k < l,\ q_{ii} \neq 0 & (14) \\ & t_{ik} \in \{0,1\} & (i,k) \in E & (15) \end{cases}$$

où $E = \{(i,k) : i = 1, \ldots, n,\ k = 0, \ldots \lfloor log(u_i) \rfloor\}$.

Proposition 3.2.1 *La valeur optimale de* $(LP_{\texttt{BBL}})$ *est égale à la valeur optimale de* (QP).

Preuve. Comme les variables w_{ikjl} ne sont présentes que dans la fonction objectif, à travers les

variables y_{ij}, et dans les contraintes (8)-(14), nous remarquons que dans une solution optimale de $(LP_{\texttt{BBL}})$, les propriétés suivantes sont satisfaites :

- Si $q_{ij} < 0$ alors $w_{ikjl} = \min(t_{ik}, t_{jl})$
- Si $q_{ij} > 0$ alors $w_{ikjl} = \max(0, t_{ik} + t_{jl} - 1)$

Il en découle que les variables w_{ikjl} sont égales aux produits $t_{ik}t_{jl}$ si $t_{ik} \in \{0,1\}$. □

Remarquons que les variables $y_{ij} = \sum_{k=0}^{\lfloor log(u_i) \rfloor} \sum_{l=0}^{\lfloor log(u_j) \rfloor} 2^{k+l} t_{ik}t_{jl} = x_i x_j$ sont introduites uniquement pour alléger l'expression de la fonction économique. Nous pouvons également remarquer que les contraintes (13) et (14) viennent de l'égalité $t_{ik}t_{jl} = t_{jl}t_{ik}$, et que les contraintes (12) viennent de la propriété $t_{ik}^2 = t_{ik}$ lorsque $t_{ik} \in \{0,1\}$.

Notons que la transformation de notre problème en variables binaires, combinée à la reformulation linéaire standard augmente considérablement le nombre de variables et contraintes. En effet, on passe de n variables dans le problème de départ à $N + N^2$ variables, avec $N = \sum_{i=1}^{n}(\lfloor log(u_i) \rfloor + 1)$, dans le problème $(LP_{\texttt{BBL}})$. De même, on passe de $m + p$ contraintes à $3N^2 + 2N + m + p$ contraintes. La taille de $(LP_{\texttt{BBL}})$ est donc de l'ordre de $O(N^2)$ variables et $O(N^2)$ contraintes. Comme les variables w_{ikjl} et les contraintes associées ne sont pas définies lorsque $q_{ij} = 0$, cette taille dépend également de la densité de la matrice Q.

3.2.2 Son renforcement BBLr

Dans cette partie, nous améliorons la méthode BBL en ajoutant à $(LP_{\texttt{BBL}})$ des inégalités valides qui découlent des mêmes idées que dans [38]. Nous générons des inégalités valides en multipliant les contraintes initiales (1) et (2) par les variables binaires t_{ik}. Ensuite, nous linéarisons les contraintes quadratiques ainsi obtenues à l'aide des variables w_{ikjl}. Nous obtenons le programme renforcé suivant :

$$(LP_{\texttt{BBLr}}) \begin{cases} Min & f_{\texttt{BBLr}}(x,y) = \sum_{i=1}^{n} q_{ij} y_{ij} + \sum_{i=1}^{n} c_i l x_i \\ s.c. & (1)(2) \\ & x, y, w, t \in P_{xywt}^{BBLr} \end{cases}$$

Avec

$$P_{xywt}^{BBLr}\begin{cases} x,y,z,t: & (3)(4)(6)(7)(15) & & \\ & w_{ikjl} \leq t_{ik} & ((i,k),(j,l)) \in E^2 & (16) \\ & w_{ikjl} \leq t_{jl} & ((i,k),(j,l)) \in E^2 & (17) \\ & w_{ikjl} \geq t_{ik} + t_{jl} - 1 & ((i,k),(j,l)) \in E^2 & (18) \\ & w_{ikjl} \geq 0 & ((i,k),(j,l)) \in E^2 & (19) \\ & w_{ikik} = t_{ik} & (i,k) \in E & (20) \\ & w_{ikjl} = w_{jlik} & ((i,k),(j,l)) \in E^2,\ i<j & (21) \\ & w_{ikil} = w_{ilik} & ((i,k),(i,l)) \in E^2,\ k<l & (22) \\ & \sum_{i=1}^{n}\sum_{k=0}^{\lfloor log(u_i)\rfloor} 2^k a_{ri} w_{ikjl} = b_r t_{jl} & (j,l) \in E,\ r \in R & (23) \\ & \sum_{i=1}^{n}\sum_{k=0}^{\lfloor log(u_i)\rfloor} 2^k d_{si} w_{ikjl} \leq e_s t_{jl} & (j,l) \in E,\ s \in S & (24) \\ & \sum_{i=1}^{n} d_{si}x_i - \sum_{i=1}^{n}\sum_{k=0}^{\lfloor log(u_i)\rfloor} 2^k d_{si} w_{ikjl} \leq e_s(1-t_{jl}) & (j,l) \in E,\ s \in S & (25) \end{cases}$$

Après avoir remplacé chaque variable entière x_i par sa décomposition binaire dans les contraintes d'égalité (1) et d'inégalité (2), nous multiplions ces nouvelles contraintes d'égalité par les variables t_{jl} pour obtenir les contraintes (23). De la même manière, nous multiplions ces nouvelles contraintes d'inégalité par les variables t_{jl} (resp. $(1-t_{jl})$) pour obtenir les contraintes (24) (resp. (25)). Ajouter ces inégalités valides introduit des variables w_{ikjl} dans les nouvelles contraintes (23)-(25). Il est donc maintenant nécessaire de définir les contraintes (16)-(22) indépendamment du signe de q_{ij}. De plus, les variables w_{ikjl} deviennent indispensables même lorsque $q_{ij} = 0$. Ainsi, la taille de $(LP_{\texttt{BBLr}})$ ne dépend plus de la densité de la matrice Q.

Notons que ce renforcement ajoute encore $Nm + 2Np$ contraintes. Il n'est donc pas évident que le renforcement de la méthode `BBL` soit rentable en pratique. Nous l'étudierons expérimentalement dans le Chapitre 6.

3.2.3 Application à l'exemple de base

Rappelons que l'Exemple 3.1.1 est un problème ayant 4 variables, 1 contrainte d'égalité et 1 contrainte d'inégalité. Nous résumons les performances des méthodes `BBL` et `BBLr` dans le tableau suivant :

solution opt. entière	**-2552**	
	`BBL`	`BBLr`
nombres de variables	156	156
nombres de contraintes	228	430
temps de résolution (s)	0.26	0.15
valeur de la relax. cont.	-7018.33	-3670.54
saut d'intégrité	175.01 %	43.82%

Dans ce tableau, le saut d'intégrité est le rapport suivant :

$$\text{saut d'integrité} = \frac{\text{val. opt. entière} - \text{val. relax. continue}}{\text{val. opt. entière}} * 100$$

Nous constatons sur cet exemple une augmentation très large de la taille des problèmes reformulés par les méthodes `BBL` et `BBLr` par rapport à la taille initiale de (QP_e). De plus, le renforcement effectué sur `BBLr` est très bénéfique puisqu'on passe d'un saut d'intégrité de 175.01% dans `BBL` à 43.82% dans `BBLr` ; il est donc divisé par un facteur 4.

Le principal inconvénient de cette méthode est la taille du problème reformulé. En effet, les méthodes `BBL` et `BBLr`, qui consistent à utiliser la linéarisation classique des programmes quadratiques en variables binaires, ne sont pas efficaces car la décomposition en variables binaires combinée à la linéarisation des produits de variables binaires fournit un programme avec un nombre important de variables et de contraintes. C'est pour cela que nous introduisons maintenant une nouvelle linéarisation qui fournit un problème reformulé de plus petite taille.

3.3 Une reformulation basée sur la linéarisation du produit d'une variable binaire par une variable entière : `BIL (Binary Integer Linearization)`

Dans cette section nous présentons notre nouvelle linéarisation, la méthode `BIL`. Cette méthode utilise la linéarisation du produit d'une variable binaire par une variable réelle présentée dans le Chapitre 2.

3.3.1 La méthode BIL

Dans cette méthode nous utilisons toujours l'unique décomposition en variables binaires $x_i = \sum_{k=0}^{\lfloor log(u_i) \rfloor} 2^k t_{ik}$. Pour linéariser les termes quadratiques généraux $x_i x_j$ avec $i \neq j$ nous utilisons l'égalité $x_i x_j = \sum_{k=0}^{\lfloor log(u_i) \rfloor} 2^k t_{ik} x_j$ que nous linéarisons en introduisant de nouvelles variables z_{ijk} pour remplacer chaque terme quadratique $t_{ik}x_j$. Puis nous ajoutons un ensemble d'inégalités qui assurent l'égalité entre z_{ijk} et $t_{ik}x_j$. Nous obtenons le programme linéaire suivant :

$$(LP_{\texttt{BIL}}) \begin{cases} Min & f_{\texttt{BIL}}(x,y) = \sum_{i=1}^{n} \sum_{\substack{j=1 \\ q_{ij} \neq 0}}^{n} q_{ij} y_{ij} + \sum_{i=1}^{n} c_i x_i \\ s.c. & (1)(2) \\ & x, y, w, z, t \in P^{BIL}_{xywzt} \end{cases}$$

Avec

$$P^{BIL}_{xywzt} \begin{cases} x, y, w, z, t : & (3)(4)(6)(12)(14)(15) & & \\ & y_{ii} = \sum_{k=0}^{\lfloor log(u_i) \rfloor} \sum_{l=0}^{\lfloor log(u_i) \rfloor} 2^{2k} w_{ikil} & i \in I & (26) \\ & w_{ikil} \leq t_{ik} & ((i,k),(i,l)) \in E^2,\ q_{ii} < 0 & (27) \\ & w_{ikil} \leq t_{il} & ((i,k),(i,l)) \in E^2,\ q_{ii} < 0 & (28) \\ & w_{ikil} \geq t_{ik} + t_{il} - 1 & ((i,k),(i,l)) \in E^2,\ q_{ii} > 0 & (29) \\ & w_{ikil} \geq 0 & ((i,k),(i,l)) \in E^2,\ q_{ii} > 0 & (30) \\ & y_{ij} = \sum_{k=0}^{\lfloor log(u_i) \rfloor} 2^k z_{ijk} & (i,j) \in I^2,\ i \neq j,\ q_{ij} \neq 0 & (31) \\ & z_{ijk} \leq u_j t_{ik} & (i,k) \in E,\ j \in I,\ q_{ij} < 0,\ i \neq j & (32) \\ & z_{ijk} \leq x_j & (i,k) \in E,\ j \in I,\ q_{ij} < 0,\ i \neq j & (33) \\ & z_{ijk} \geq x_j - u_j(1 - t_{ik}) & (i,k) \in E,\ j \in I,\ q_{ij} > 0,\ i \neq j & (34) \\ & z_{ijk} \geq 0 & (i,k) \in E,\ j \in I,\ q_{ij} > 0,\ i \neq j & (35) \\ & y_{ij} = y_{ji} & (i,j) \in I^2 & (36) \end{cases}$$

Afin de ne pas perdre l'amélioration de la valeur de la relaxation continue induite par la propriété $t_{ik}^2 = t_{ik}$, nous linéarisons les carrés différemment des autres produits. Comme dans la méthode BBL nous utilisons les variables w_{ikil} qui représentent les produits $t_{ik}t_{il}$ et les inégalités (27)-(30) pour linéariser les carrés.

Proposition 3.3.1 *La valeur optimale de (QP) est égale à la valeur optimale de $(LP_{\texttt{BIL}})$.*

Preuve. Nous remarquons que dans une solution optimale de $(LP_{\texttt{BIL}})$, nous avons :

- Si $q_{ij} < 0$ alors $z_{ijk} = \min(u_j t_{ik}, x_j)$
- Si $q_{ij} > 0$ alors $z_{ijk} = \max(0, u_j t_{ik} + x_j - u_j)$

Il en découle que si $t_{ik} = 0$ alors $z_{ijk} = 0$ et si $t_{ik} = 1$ alors $z_{ijk} = x_j$. Ceci implique que dans toute solution optimale entière nous avons $z_{ijk} = t_{ik} x_j$. Pour les mêmes raisons que dans la reformulation $(LP_{\texttt{BBL}})$, nous avons aussi $w_{ikil} = t_{ik} t_{il}$. Le programme $(LP_{\texttt{BIL}})$ est donc un programme linéaire en variables mixtes équivalent à (QP). □

Remarquons que les contraintes (36) viennent du fait que dans tout produit $x_i x_j$ soit la variable x_i soit la variable x_j peuvent être remplacées par leurs décompositions binaires. En effet, soit $y_{ij} = \sum_{k=0}^{\lfloor log(u_i) \rfloor} 2^k z_{ijk}$, soit $y_{ji} = \sum_{l=0}^{\lfloor log(u_j) \rfloor} 2^l z_{jil}$, avec $\sum_{k=0}^{\lfloor log(u_i) \rfloor} 2^k z_{ijk}$ qui n'est pas nécessairement égal à $\sum_{l=0}^{\lfloor log(u_j) \rfloor} 2^l z_{jil}$.

La méthode `BIL` produit un programme $(LP_{\texttt{BIL}})$ avec $O(nN)$ variables et contraintes. Comme dans la méthode `BBL`, il n'est pas nécessaire de définir les variables z_{ijk} lorsque $q_{ij} = 0$. Sa taille dépend donc également de la densité de la matrice Q.

3.3.2 Son renforcement `BILr`

Comme pour la méthode `BBL`, nous pouvons renforcer la méthode `BIL` du point de vue de la borne obtenue par relaxation continue. Pour cela, nous ajoutons les variables z_{iik} qui représentent les produits $t_{ik} x_i$, et les contraintes (47)-(48) qui sont des inégalités valides. Puis, toujours en adaptant les idées de [38], nous ajoutons les contraintes (49)-(52). Il est également nécessaire de transformer les contraintes (32)-(35) en les contraintes (42)-(45). Nous obtenons donc le programme linéaire en variables mixtes $(LP_{\texttt{BILr}})$ suivant .

$$(LP_{\texttt{BILr}}) \begin{cases} Min & f_{\texttt{BBLr}}(x,y) = \sum_{i=1}^{n} q_{ij} y_{ij} + \sum_{i=1}^{n} c_i x_i \\ s.c. & (1)(2) \\ & x, y, w, z, t \in P_{xywzt}^{BILr} \end{cases}$$

Avec

$$P^{BILr}_{xywzt}\begin{cases} x,y,z,w,t: & (3)(4)(6)(15)(20)(22)(26)(36) & & \\ & w_{ikil} \leq t_{ik} & ((i,k),(i,l)) \in E^2 & (37) \\ & w_{ikil} \leq t_{il} & ((i,k),(i,l)) \in E^2 & (38) \\ & w_{ikil} \geq t_{ik} + t_{il} - 1 & ((i,k),(i,l)) \in E^2 & (39) \\ & w_{ikil} \geq 0 & ((i,k),(i,l)) \in E^2 & (40) \\ & y_{ij} = \sum_{k=0}^{\lfloor log(u_i) \rfloor} 2^k z_{ijk} & (i,j) \in I^2 & (41) \\ & z_{ijk} \leq u_j t_{ik} & (i,k) \in E,\ j \in I & (42) \\ & z_{ijk} \leq x_j & (i,k) \in E,\ j \in I & (43) \\ & z_{ijk} \geq x_j - u_j(1 - t_{ik}) & (i,k) \in E,\ j \in I & (44) \\ & z_{ijk} \geq 0 & (i,k) \in E,\ j \in I & (45) \\ & z_{iik} = \sum_{l=0}^{\lfloor log(u_i) \rfloor} 2^l y_{ikil} & (i,k) \in E & (46) \\ & y_{ij} \geq x_i u_j + x_j u_i - u_i u_j & (i,j) \in I^2 & (47) \\ & y_{ii} \geq x_i & i \in I & (48) \\ & \sum_{i=1}^{n} a_{ri} z_{jil} = b_r t_{jl} & (j,l) \in E,\ r \in R & (49) \\ & \sum_{i=1}^{n} d_{si} z_{jil} \leq e_s t_{jl} & (j,l) \in E,\ s \in S & (50) \\ & \sum_{i=1}^{n} (d_{si} x_i - d_{si} z_{jil}) \leq e_s(1 - t_{jl}) & (j,l) \in E,\ s \in S & (51) \\ & \sum_{i=1}^{n} (d_{si} x_i u_j - d_{si} \sum_{k=0}^{\lfloor log(u_i) \rfloor} 2^k z_{ijk}) \leq e_s(u_j - x_j) & j \in I,\ s \in S & (52) \end{cases}$$

Nous obtenons les inégalités valides (46)-(52) de la façon suivante :

- Les contraintes (46) définissent les variables z_{iik} qui représentent les produits $t_{ik}x_i$ dans une solution entière.
- Les contraintes (47) viennent de l'inégalité $(x_i - u_i)(x_j - u_j) \geq 0$.
- Les contraintes (48) viennent de l'inégalité $x_i^2 \geq x_i$ qui est satisfaite par n'importe quel entier x_i.
- Les contraintes (49) sont obtenues en multipliant les contraintes d'égalités (1) par les variables t_{jl}.
- Les contraintes (50) sont obtenues en multipliant les contraintes d'inégalités (2) par les variables t_{jl}.
- Les contraintes (51) sont obtenues en multipliant les contraintes d'inégalités (2) par $(1-t_{jl})$.

– Les contraintes (52) sont obtenues en multipliant les contraintes d'inégalités (2) par $(u_j - x_j)$.

Comme dans la version renforcée de `BBL`, la multiplication des contraintes (1) et (2) par les variables x_j et t_{jl} introduit les variables z_{ijk} dans les nouvelles contraintes (49)-(52). C'est pourquoi, il est nécessaire de définir les contraintes (37)-(40) et (42)-(45) indépendamment du signe de q_{ij}. De plus, les variables z_{ijk} sont également nécessaires lorsque $q_{ij} = 0$.

3.3.3 Application à l'exemple de base

Rappelons que l'Exemple 3.1.1 est un problème ayant 4 variables, 1 contrainte d'égalité et 1 contrainte d'inégalité. Nous résumons les performances des méthodes `BIL` et `BILr` dans le tableau suivant :

solution opt. entière	**-2552**	
	`BIL`	`BILr`
nombres de variables	132	132
nombres de contraintes	174	370
temps de résolution (s)	0.05	0.09
valeur de la relax. cont.	-5082.5	-3256.71
saut d'intégrité	99.15 %	27.6%

Dans ce tableau, le saut d'intégrité est encore le rapport suivant :

$$\text{saut d'integrité} = \frac{\text{val. opt. entière} - \text{val. relax. continue}}{\text{val. opt. entière}} * 100$$

Nous constatons sur cet exemple, une augmentation conséquente de la taille des problèmes reformulés par les méthodes `BIL` et `BILr`, par rapport à la taille initiale de (QP_e). De plus, le renforcement effectué sur `BILr` améliore significativement la valeur de la borne obtenue par relaxation continue, puisque le saut d'intégrité passe de 99.15% dans `BIL` à 27.6% dans `BILr` ; il est donc divisé par un facteur 5.

Nous pouvons maintenant comparer les quatre méthodes présentées dans ce chapitre sur l'Exemple 3.1.1. Un résumé est présenté dans le tableau suivant :

solution opt. entière	**-2552**			
	BBL	BBLr	BIL	BILr
nombres de variables	156	156	132	132
nombres de contraintes	228	430	174	370
temps de résolution (s)	0.26	0.15	0.05	0.09
valeur de la relax. cont.	-7018.33	-3670.54	-5082.5	-3256.71
saut d'intégrité	175.01 %	43.82%	99.15 %	27.6%

3.4 Comparaison théorique de BBL et BIL

Dans cette section, nous comparons théoriquement la qualité de la valeur de la relaxation continue des reformulations BBL et BIL.

Théorème 3.4.1 *En notant $v(P)$ la valeur de la solution optimale d'un problème (P), nous avons $v(\overline{LP}_{\texttt{BIL}}) \geq v(\overline{LP}_{\texttt{BBL}})$. Cette inégalité peut être stricte.*

Preuve. Montrons que pour toute solution réalisable $(\bar{x}, \bar{y}, \bar{w}, \bar{z}, \bar{t})$ de $(\overline{LP}_{\texttt{BIL}})$, nous pouvons construire une solution réalisable $(\bar{x}, \bar{y}, w, t)$ de $(\overline{LP}_{\texttt{BBL}})$.
Soit $(\bar{x}, \bar{y}, \bar{w}, \bar{z}, \bar{t})$ une solution réalisable de $(\overline{LP_{\texttt{BIL}}})$. Nous prenons $\forall (i,k) \in E$, $t_{ik} = \bar{x}_i/\bar{u}_i$, $\forall ((i,k),(j,l)) \in E^2$, $i < j$, $k \neq l$, $w_{ikjl} = \bar{y}_{ij}/\bar{u}_i\bar{u}_j$, $\forall (i,k),(i,l) \in E$, $k \leq l$, $w_{ikil} = \bar{w}_{ikil}$. De manière évidente, les contraintes (1), (2), (3), (4), (6), (7), (11), (12), (13), (14) et (15') sont satisfaites.
De plus les contraintes (8)-(10) sont directement satisfaites pour tout $(i,k) \in E$. Montrons maintenant qu'une telle solution satisfait les contraintes (8)-(10), pour tout $((i,k),(j,l)) \in E^2$ avec $i \leq j$ et $k \neq l$:

- Les contraintes (8), $w_{ikjl} \leq \bar{t}_{ik}$:
  Par les contraintes (32), nous avons $w_{ikjl} = \bar{y}_{ij}/\bar{u}_i\bar{u}_j = \sum_{k=0}^{\lfloor log(u_i) \rfloor} 2^k \bar{z}_{ijk}/\bar{u}_i\bar{u}_j \leq \sum_{k=0}^{\lfloor log(u_i) \rfloor} 2^k u_j \bar{t}_{ik}/\bar{u}_i\bar{u}_j = \bar{x}_i u_j/\bar{u}_i\bar{u}_j = \bar{t}_{ik} u_j/\bar{u}_j \leq \bar{t}_{ik}$
- Les contraintes (9), $w_{ikjl} \leq \bar{t}_{jl}$:
  Par les contraintes (33), nous avons $w_{ikjl} = \bar{y}_{ij}/\bar{u}_i\bar{u}_j = \sum_{k=0}^{\lfloor log(u_i) \rfloor} 2^k \bar{z}_{ijk}/\bar{u}_i\bar{u}_j \leq \sum_{k=0}^{\lfloor log(u_i) \rfloor} 2^k \bar{x}_j/\bar{u}_i\bar{u}_j = \bar{u}_i\bar{x}_j/\bar{u}_i\bar{u}_j = \bar{x}_j/\bar{u}_j = \bar{t}_{jl}$
- Les contraintes (10), $w_{ikjl} \geq t_{ik} + t_{jl} - 1$:
  Par les contraintes (34), nous avons $w_{ikjl} = \bar{y}_{ij}/\bar{u}_i\bar{u}_j = \sum_{k=0}^{\lfloor log(u_i) \rfloor} 2^k \bar{z}_{ijk}/\bar{u}_i\bar{u}_j \geq \sum_{k=0}^{\lfloor log(u_i) \rfloor} 2^k (\bar{x}_j -$

$u_j + u_j\bar{t}_{ik})/\bar{u}_i\bar{u}_j = \bar{u}_i\bar{x}_j/\bar{u}_i\bar{u}_j - \bar{u}_i u_j/\bar{u}_i\bar{u}_j + \bar{x}_i u_j/\bar{u}_i\bar{u}_j = \bar{x}_j/\bar{u}_j - u_j/\bar{u}_j(1 - \bar{x}_i/\bar{u}_i) = \bar{t}_{jl} - u_j/\bar{u}_j(1 - \bar{t}_{ik}) \geq \bar{t}_{jl} + \bar{t}_{ik} - 1$

Pour toute solution réalisable $(\bar{x}, \bar{y}, \bar{w}, \bar{z}, \bar{t})$ de $(\overline{LP}_{\texttt{BIL}})$ nous pouvons construire une solution réalisable $(\bar{x}, \bar{y}, w, t)$ de $(\overline{LP}_{\texttt{BBL}})$. De plus, comme les fonctions objectifs de $(\overline{LP}_{\texttt{BIL}})$ et $(\overline{LP}_{\texttt{BBL}})$ sont identiques et ne concernent que les variables x et y, ces solutions ont même valeur. Et comme $(\overline{LP}_{\texttt{BIL}})$ et $(\overline{LP}_{\texttt{BBL}})$ sont des problèmes de minimisation, nous avons donc :

$$v(\overline{LP}_{\texttt{BIL}}) \geq v(\overline{LP}_{\texttt{BBL}})$$

Cette inégalité peut être stricte (cf. Exemple 3.1.1). □

3.5 Conclusion

Dans ce chapitre, nous avons présenté quatre reformulations linéaires de (QP). Tout d'abord, nous avons proposé la reformulation linéaire `BBL` qui consiste à remplacer chaque variable entière par sa décomposition en variables binaires puis à linéariser les nouveaux produits de deux variables binaires par la linéarisation de Fortet [13]. Nous avons ensuite renforcé cette reformulation en lui ajoutant des inégalités valides dans la méthode appelée `BBLr`. L'étude de cette approche nous montre que les programmes obtenus par ces reformulations possèdent un très grand nombre de variables et de contraintes. De plus, nos expérimentations montrent que leurs bornes obtenues par relaxation continue sont trop faibles.

Ensuite, nous avons proposé la reformulation linéaire `BIL` qui consiste à remplacer, dans chaque produit de variables entières une des deux variables par sa décomposition binaire. Puis, nous linéarisons chaque nouveau produit d'une variable entière par un variable binaire par la linéarisation de McCormick [31]. Nous avons ensuite renforcé cette reformulation en lui ajoutant des inégalités valides dans la méthode appelée `BILr`. Cette approche réduit significativement le nombre de variables et de contraintes additionnelles en comparaison avec l'approche de la reformulation `BBL`. De plus, nous avons démontré que la borne obtenue par relaxation continue fournie par la reformulation `BIL` est toujours de meilleure qualité que celle fournie par la reformulation `BBL`. Ce résultat est intéressant car il allie une réduction de taille du problème reformulé et une amélioration de sa borne. Nos expérimentations corroborent nos résultats théoriques.

Une piste de recherche pour la reformulation linéaire `BILr` est la construction d'un algorithme de Branch and Cut approprié qui intègre les inégalités au fur et à mesure de la résolution entière.

Chapitre 4

Reformulations quadratiques convexes de programmes quadratiques en nombres entiers

4.1 Introduction

Dans ce chapitre, nous présentons différentes reformulations quadratiques convexes de (QP). Rappelons la définition de (QP) :

$$(QP)\begin{cases} Min & f(x) = \sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij}x_ix_j + \sum_{i=1}^{n} c_ix_i & & \\ s.c. & \sum_{i=1}^{n} a_{ri}x_i = b_r & r \in R & (1) \\ & \sum_{i=1}^{n} d_{si}x_i \leq e_s & s \in S & (2) \\ & x_i \leq u_i & i \in I & (3) \\ & x_i \geq 0 & i \in I & (4) \\ & x_i \in \mathbb{N} & i \in I & (5) \end{cases}$$

où $I = \{i : i = 1, \ldots, n\}$, $R = \{r : r = 1, \ldots, m\}$, et $S = \{s : s = 1, \ldots, p\}$.

Une reformulation convexe consiste à construire un nouveau programme équivalent à (QP) ayant une fonction objectif quadratique et convexe, i.e. dont le Hessien est semi-défini positif.

Notre première approche, appelée `NC (Naive Convexification)`, consiste à étendre la méthode de la plus petite valeur propre [19], décrite dans le Chapitre 2, aux problèmes quadratiques en nombres entiers en perturbant la diagonale de Q avec sa plus petite valeur propre.

Nous introduisons ensuite un nouveau schéma de reformulations convexes de (QP). Ce

schéma consiste à perturber, grâce à l'expression linéaire des produits des variables entières, tous les termes de la matrice Q avec un paramètre matriciel β, et également, à utiliser les contraintes d'égalité dans la perturbation de la fonction objectif à l'aide d'un paramètre scalaire α. Puis, au sein de ce schéma, nous calculons une reformulation convexe optimale du point de vue de la borne obtenue par relaxation continue. Nous prouvons que notre reformulation optimale est basée sur la solution optimale duale d'une relaxation semi-définie de (QP).

La présentation de cette approche, que nous appelons `IQCR (Integer Quadratic Convex Reformulation)`, se décompose en trois étapes. D'abord nous présentons le nouveau schéma de reformulations convexes pour les problèmes quadratiques en nombres entiers qui dépend du paramètre scalaire α et du paramètre matriciel β. Ce schéma implique l'ajout d'un nombre important de variables et de contraintes. Ainsi, dans une deuxième étape, nous projetons le polyèdre associé à la relaxation continue du problème reformulé sur un espace de plus petite taille. Finalement, nous montrons comment calculer les paramètres optimaux, α^* et β^*, au sein de notre schéma et selon notre critère d'optimalité.

Ensuite, nous montrons que `IQCR`, bien qu'initialement conçue pour les programmes purement en nombres entiers, peut aisément être adapté à la résolution de programmes quadratiques en nombres mixtes entiers. Nous appelons `IQCRs` cette adaptation d'`IQCR`. Cette adaptation est intéressante car elle permet de traiter une classe beaucoup plus importante de problèmes. De plus, en transformant, grâce à des variables d'écart réelles, les contraintes d'inégalité en contraintes d'égalité, `IQCRs` permet de perturber la fonction objectif du problème reformulé avec ces nouvelles contraintes.

`IQCR` fournit une borne obtenue par relaxation continue de très bonne qualité, mais aussi un programme reformulé de taille conséquente. C'est pour cela que nous présentons une restriction de `IQCR`, la méthode `CQCR (compact Quadratic Convex Reformulation)`. La différence entre cette approche et la méthode `IQCR` est que `CQCR` n'utilise que les expressions linéaires des carrés des variables pour perturber la fonction objectif, alors qu'`IQCR` utilise celles de tous les produits. Cette méthode fournit une reformulation quadratique convexe de (QP) au sein d'un schéma de plus petite taille que celui de la méthode `IQCR`. Ainsi, cette reformulation étant plus compacte, elle peut être avantageuse expérimentalement.

Dans une dernière partie, nous mettons en parallèle les convexifications pour la programmation quadratique en variables $0-1$ citées dans le Chapitre 2 avec nos convexifications appliquées à ces mêmes problèmes. Nous montrons qu'en prenant les bornes supérieures sur les variables x_i égales à 1, il y a une correspondance directe entre `NC` et la méthode de la plus petite valeur propre [19], ainsi qu'entre `CQCR` et `QCR` [4]. Nous appliquons ensuite `IQCR` à la programmation quadratique en variables binaires et nous prouvons qu'elle fournit une borne obtenue par relaxation

continue de meilleure qualité que celle fournie par la méthode `QCR`.

4.2 Une reformulation convexe naïve : `NC (Naive Convexification)`

Comme nous l'avons décrit dans le Chapitre 2, il existe dans la littérature plusieurs méthodes de reformulation convexe pour les problèmes quadratiques en variables binaires. Seulement, la propriété $x_i^2 = x_i$ qui est à la base de ces méthodes n'est plus valable pour les problèmes en variables entières. La façon la plus naïve d'étendre la méthode de la petite valeur propre [19] à la programmation en variables entières est probablement d'ajouter à la fonction objectif $-\lambda_{min}(Q)\sum_{i=1}^{n}(x_i^2 - v_i)$ avec $v_i = x_i^2 = \sum_{k=0}^{u_i} k^2 r_{ik}$, où r_{ik} sont des variables binaires additionnelles qui satisfont les contraintes $\sum_{k=0}^{u_i} r_{ik} = 1$ et $x_i = \sum_{k=0}^{u_i} k r_{ik}$. Cela nous fournit une méthode simple, que nous appelons `NC (Naive Convexification)`, pour résoudre (QP) par un solveur standard. Plus formellement, la reformulation par `NC` peut s'écrire de la façon suivante :

$$(QP_{\lambda_{min}}) \begin{cases} Min & f(x,v)_{\lambda_{min}} = x^T Q x + c^T x - \lambda_{min}(Q)\sum_{i=1}^{n}(x_i^2 - v_i) & & \\ s.c. & (1)(2) & & \\ & x_i = \sum_{k=0}^{u_i} k r_{ik} & i \in I & (6) \\ & v_i = \sum_{k=0}^{u_i} k^2 r_{ik} & i \in I & (7) \\ & \sum_{k=0}^{u_i} r_{ik} = 1 & i \in I & (8) \\ & r_{ik} \in \{0,1\} & (i,k) \in E & (9) \end{cases}$$

L'avantage de cette méthode est qu'elle est facile à mettre en oeuvre. Elle implique l'ajout de $\sum_{i=1}^{n}(u_i + 1)$ variables binaires et $3n$ contraintes, et utilise la décomposition unaire des variables entières pour exprimer linéairement les carrés des variables x_i. Cette reformulation convexe, qui est la plus naïve parmi celles que nous présentons, nous servira de référence pour nos expérimentations.

Application à l'exemple de base

Rappelons d'abord l'Exemple 3.1.1 qui est un problème ayant 4 variables, 1 contrainte d'égalité et 1 contrainte d'inégalité :

Exemple 3.1.1 (suite) :

$$(QP_e)\begin{cases} Min & f(x) = x^T \begin{pmatrix} 5 & -7 & -6 & -1 \\ -7 & 3 & -8 & -18 \\ -6 & -8 & -17 & 10 \\ -1 & -18 & 10 & 3 \end{pmatrix} x + \begin{pmatrix} -5 \\ -11 \\ 4 \\ 1 \end{pmatrix} x \\ s.c. & 3x_1 + 19x_2 + 18x_3 + 11x_4 = 255 \\ & 11x_1 + 13x_2 + 8x_3 + x_4 \leq 165 \\ & 0 \leq x_i \leq 10 & i \in \{1, \ldots, 4\} \\ & x_i \in \mathbb{N} & i \in \{1, \ldots, 4\} \end{cases}$$

La valeur de la solution optimale entière de (QP_e) *est* -2552.

Avec la méthode `NC`*, nous perturbons le Hessien de* $f(x)$ *de la manière suivante :*

$$\begin{pmatrix} 5 - \lambda_{min}(Q) & -7 & -6 & -1 \\ -7 & 3 - \lambda_{min}(Q) & -8 & -18 \\ -6 & -8 & -17 - \lambda_{min}(Q) & 10 \\ -1 & -18 & 10 & 3 - \lambda_{min}(Q) \end{pmatrix}$$

avec $\lambda_{min}(Q) = -22.64$.

Nous résumons les performances de la méthode `NC` dans le tableau suivant :

solution opt. entière	**-2552**
	`NC`
nombres de variables	52
nombres de contraintes	14
temps de résolution (s)	0.01
valeur de la relax. cont.	-3991.85
saut d'intégrité	56.42 %

Dans ce tableau, le saut d'intégrité est encore le rapport suivant :

$$\text{saut d'integrité} = \frac{\text{val. opt. entière} - \text{val. relax. continue}}{\text{val. opt. entière}} * 100$$

4.3 La méthode IQCR (Integer Quadratic Convex Reformulation)

4.3.1 Un schéma général de reformulations convexes pour les programmes quadratiques en nombres entiers

Dans cette section, nous présentons notre schéma général de reformulations convexes. Etant donné un paramètre scalaire α et un paramètre matriciel β, nous construisons le programme $(QP_{\alpha,\beta})$ équivalent à (QP). $(QP_{\alpha,\beta})$ peut être vu comme une transformation compacte de (QP) en un programme en variables mixtes réelles et binaires. Ensuite, nous prouvons une propriété de projection sur le polyèdre défini dans $(QP_{\alpha,\beta})$. Cette propriété de projection nous permet de prouver dans la Section 4.3.3 que les valeurs optimales des paramètres α et β peuvent être déduites d'une certaine relaxation semi-définie de (QP). Un des intérêts de cette propriété de projection est que cette relaxation semi-définie est construite seulement à partir des variables initiales x et de la relaxation classique de $X = xx^T$ en $X \succeq xx^T$.

Réécrivons (QP) en $(QP_{\alpha,\beta})$. L'idée est d'ajouter à la fonction objectif initiale $f(x)$ les fonctions suivantes qui s'annulent sur le domaine réalisable de (QP) et sous l'hypothèse $y_{ij} = x_i x_j$:

(i) $\alpha \sum_{r=1}^{m} (\sum_{i=1}^{n} a_{ri} x_i - b_r)^2$ où $\alpha \in \mathbb{R}$

(ii) $\sum_{i=1}^{n} \sum_{j=1}^{n} \beta_{ij} (x_i x_j - y_{ij})$ où $\beta_{ij} \in \mathbb{R}$ et $\beta_{ij} = \beta_{ji}$.

Soit $(QP_{\alpha,\beta})$ le programme suivant :

$$(QP_{\alpha,\beta}) \begin{cases} Min & f_{\alpha,\beta}(x,y) = f(x) + \alpha \sum_{r=1}^{m} (\sum_{i=1}^{n} a_{ri} x_i - b_r)^2 + \sum_{i=1}^{n} \sum_{j=1}^{n} \beta_{ij} (x_i x_j - y_{ij}) \\ s.c. & (1)(2) \\ & x, y, z, t \in P_{xyzt} \end{cases}$$

avec P_{xyzt} l'ensemble suivant :

$$
P_{xyzt}\begin{cases}
x,y,z,t: & (3)(4) & & \\
& x_i = \sum_{k=0}^{\lfloor log(u_i)\rfloor} 2^k t_{ik} & i \in I & (10)\\
& y_{ij} = \sum_{k=0}^{\lfloor log(u_i)\rfloor} 2^k z_{ijk} & (i,j) \in I^2 & (11)\\
& z_{ijk} \leq u_j t_{ik} & (i,k) \in E,\ j \in I & (12)\\
& z_{ijk} \leq x_j & (i,k) \in E,\ j \in I & (13)\\
& z_{ijk} \geq x_j - u_j(1-t_{ik}) & (i,k) \in E,\ j \in I & (14)\\
& z_{ijk} \geq 0 & (i,k) \in E,\ j \in I & (15)\\
& y_{ij} = y_{ji} & (i,j) \in I^2 & (16)\\
& y_{ij} \geq x_i u_j + x_j u_i - u_i u_j & (i,j) \in I^2 & (17)\\
& y_{ii} \geq x_i & i \in I & (18)\\
& t_{ik} \in \{0,1\} & (i,k) \in E & (19)
\end{cases}
$$

où $E = \{(i,k) : i = 1,\ldots,n,\ k = 0,\ldots \lfloor log(u_i)\rfloor\}$.

Montrons que $(QP_{\alpha,\beta})$ est équivalent à (QP). D'abord, il est évident que si x satisfait les contraintes (1) et que $x_i x_j = y_{ij}$, $f_{\alpha,\beta}(x,y) = f(x)$. Donc, il nous reste à montrer que les contraintes linéaires de P_{xyzt} assurent l'égalité $x_i x_j = y_{ij}$.

Pour construire P_{xyzt}, nous avons utilisé les mêmes idées que celles de la linéarisation `BIL` (cf. Chapitre 3). En effet, nous utilisons l'unique décomposition binaire $x_i = \sum_{k=0}^{\lfloor log(u_i)\rfloor} 2^k t_{ik}$. Ensuite, nous déduisons l'égalité $x_i x_j = \sum_{k=0}^{\lfloor log(u_i)\rfloor} 2^k t_{ik} x_j = y_{ij}$, que nous linéarisons en introduisant les nouvelles variables réelles z_{ijk} pour remplacer chaque terme quadratique $t_{ik}x_j$. Enfin, nous ajoutons l'ensemble d'inégalités (12)-(15) et la contrainte (19) qui forcent z_{ijk} à être égal à $t_{ik}x_j$. Pour prouver que dans l'ensemble P_{xyzt}, $z_{ijk} = t_{ik}x_j$, il est suffisant de considérer les deux valeurs possibles de t_{ik}.

Il faut remarquer que les contraintes (16), (17) et (18) sont redondantes lorsque les variables sont entières, mais elles sont nécessaires dans la suite de ce chapitre. Rappelons que les contraintes (16) viennent de l'égalité $x_i x_j = x_j x_i$. Nous avons construit les contraintes (17) en multipliant les contraintes (3) deux à deux : $(x_i - u_i)(x_j - u_j) \geq 0$, puis en linéarisant la nouvelle inégalité ainsi obtenue. Cette idée est connue dans la littérature et a été introduit notamment par McCormick [31]. Les contraintes (18) viennent de l'inégalité $x_i^2 \geq x_i$ qui est satisfaite par n'importe quel entier x_i.

Ici, nous cherchons à reformuler (QP) en $(QP_{\alpha,\beta})$. La reformulation de (QP) en $(QP_{\alpha,\beta})$

n'est intéressante que si $f_{\alpha,\beta}(x,y)$ est convexe. Ceci est toujours possible. Par exemple, si on note $\lambda_{min}(Q)$ la plus petite valeur propre de Q, en prenant $\forall i \in I$, $\bar{\beta}_{ii} = -\lambda_{min}(Q)$, $\forall i,j \in I$, $i \neq j$, $\bar{\beta}_{ij} = 0$, et $\bar{\alpha}$ n'importe quel réel positif, il est clair que la fonction $f_{\bar{\alpha},\bar{\beta}}(x,y)$ est convexe.

Nous avons choisi pour notre schéma de reformulations convexes, de perturber chaque terme de la matrice Q par un coefficient particulier β_{ij}. Il est connu qu'il est suffisant de perturber les termes diagonaux de la matrice Q pour la rendre semi-définie positive. Nous aurions donc pu choisir de ne linéariser que les termes x_i^2 auxquels nous aurions associé un paramètre vectoriel. Ce schéma est celui de la méthode `CQCR` présentée dans la Section 4.4, et comme nous le verrons dans nos expérimentations (cf. Chapitre 6), cette approche fournit une borne obtenue par relaxation continue de moins bonne qualité que celle fournie par l'approche `IQCR`.

De plus, il est intéressant de remarquer que nous avons choisi un multiplicateur commun α, pour les équations quadratiques $\sum_{r=1}^{m}(\sum_{i=1}^{n} a_{ri}x_i - b_r)^2$. Au premier abord, il semblerait qu'on puisse obtenir une meilleure borne en prenant un multiplicateur spécifique pour chaque équation $(\sum_{i=1}^{n} a_{ri}x_i - b_r)^2$. Cependant, ces deux approches sont équivalentes. En effet, si avec $(\bar{\alpha},\bar{\beta})$, la matrice $Q_{\bar{\alpha},\bar{\beta}} = Q + \bar{\alpha}A^TA + \bar{\beta}$ est semi-définie positive, elle l'est toujours avec $(\bar{\alpha}',\bar{\beta})$, tel que $\bar{\alpha}' \geq \bar{\alpha}$, et la valeur de la relaxation continue ne change pas puisque les contraintes (1) sont toujours satisfaites dans une solution réalisable. Il est assez simple de prouver qu'il est équivalent de prendre un multiplicateur commun ou un coefficient particulier pour chaque équation quadratique. En effet, si on prend $(\bar{\alpha},\bar{\beta})$ tels que $f_{\bar{\alpha},\bar{\beta}}(x,y)$ soit convexe. Soit α' n'importe quel réel positif, et $f_{\bar{\alpha},\bar{\beta},\alpha'}(x,y)$ la fonction suivante :

$$\begin{aligned} f_{\bar{\alpha},\bar{\beta},\alpha'}(x,y) &= f_{\bar{\alpha},\bar{\beta}}(x,y) + \alpha'\sum_{r=1}^{m}(\sum_{i=1}^{n} a_{ri}x_i - b_r)^2 \\ &= f(x) + \bar{\alpha}\sum_{r=1}^{m}(\sum_{i=1}^{n} a_{ri}x_i - b_r)^2 + \sum_{i=1}^{n}\sum_{j=1}^{n}\bar{\beta}_{ij}(x_ix_j - y_{ij}) + \alpha'\sum_{r=1}^{m}(\sum_{i=1}^{n} a_{ri}x_i - b_r)^2 \end{aligned}$$

Comme α' est positif, nous savons que : $\alpha'\sum_{r=1}^{m}(\sum_{i=1}^{n} a_{ri}x_i - b_r)^2$ est convexe, la fonction $f_{\bar{\alpha},\bar{\beta},\alpha'}(x,y)$ est donc convexe, puisque la somme de deux fonctions convexes est une fonction convexe. De plus, comme $\sum_{r=1}^{m}(\sum_{i=1}^{n} a_{ri}x_i - b_r)^2 = 0$, nous avons bien

$$\begin{aligned} f_{\bar{\alpha},\bar{\beta},\alpha'}(x,y) &= f(x) + (\bar{\alpha}+\alpha')\sum_{r=1}^{m}(\sum_{i=1}^{n} a_{ri}x_i - b_r)^2 + \sum_{i=1}^{n}\sum_{j=1}^{n}\bar{\beta}_{ij}(x_ix_j - y_{ij}) \\ &= f_{\bar{\alpha},\bar{\beta}}(x,y) \end{aligned}$$

Ceci est vrai pour toute solution admissible de $(QP_{\alpha,\beta})$.

Une méthode classique de résolution de programmes en nombres entiers consiste à appliquer un algorithme de Branch and Bound. Il est connu que le comportement d'un tel algorithme dépend de la qualité de la borne à la racine de l'arbre de résolution. C'est pourquoi dans la suite de cette section, nous nous intéresserons particulièrement à la relaxation continue de $(QP_{\alpha,\beta})$, que nous appelons $(\overline{QP}_{\alpha,\beta})$:

$$(\overline{QP}_{\alpha,\beta}) \begin{cases} Min & f_{\alpha,\beta}(x,y) = f(x) + \alpha\sum_{r=1}^{m}(\sum_{i=1}^{n} a_{ri}x_i - b_r)^2 + \sum_{i=1}^{n}\sum_{j=1}^{n}\beta_{ij}(x_ix_j - y_{ij}) \\ s.c. & (1)(2) \\ & x,y,z,t \in \overline{P}_{xyzt} \end{cases}$$

où $\overline{P}_{xyzt}$ est le polyèdre obtenu en relâchant les contraintes d'intégrité de P_{xyzt}, et donc en remplaçant les contraintes (19) par

$$0 \leq t_{ik} \leq 1 \quad (19')$$

Dans la suite, nous prouvons que $(\overline{QP}_{\alpha,\beta})$ a la même valeur optimale qu'un autre programme de plus petite taille.

4.3.2 Une projection du polyèdre associé

Etant donné que les variables z et t n'interviennent pas dans la fonction objectif $f_{\alpha,\beta}(x,y)$, il est intéressant de définir une projection de l'ensemble $\overline{P}_{xyzt}$ sur x et y.

Théorème 4.3.1 *La projection de $\overline{P}_{xyzt}$ sur x et y est le polyèdre suivant P_{xy} :*

$$P_{xy} \begin{cases} x,y : & (16)(17)(18) & & \\ & y_{ij} \geq 0 & (i,j) \in I^2 & (20) \\ & y_{ij} \leq u_j x_i & (i,j) \in I^2 & (21) \\ & y_{ij} \leq u_i x_j & (i,j) \in I^2 & (22) \end{cases}$$

Preuve.

1. Montrons que pour tout $(\bar{x},\bar{y})$ appartenant à P_{xy}, il existe z,t tels que $(\bar{x},\bar{y},z,t)$ appartiennent à $\overline{P}_{xyzt}$.

   Soit $(\bar{x},\bar{y}) \in P_{xy}$, et $\gamma_{ij} \geq 0$ tels que $\bar{y}_{ij} = \gamma_{ij}\bar{x}_i\bar{x}_j$. Prenons $\forall (i,k) \in E$, $t_{ik} = \bar{x}_i/\bar{u}_i$, avec $\bar{u}_i = \sum_{k=0}^{\lfloor log(u_i) \rfloor} 2^k$, et $\forall (i,k) \in E$, $j \in I$, $z_{ijk} = \gamma_{ij}t_{ik}\bar{x}_j$. De manière évidente les

contraintes (3), (4), (10), (11), (15), (16), (17), (18) et (19') sont satisfaites.

Montrons maintenant que dans une telle solution, les contraintes (12)-(14) sont satisfaites :

- Les contraintes (12), $z_{ijk} \leq u_j t_{ik}$:
  (i) Si $\bar{x}_i = 0$, $t_{ik} = 0$, alors $z_{ijk} = 0$.
  (ii) Si $\bar{x}_i > 0$, par les contraintes (21), nous avons $\gamma_{ij}\bar{x}_i\bar{x}_j \leq u_j\bar{x}_i$ et donc $\gamma_{ij}x_j \leq u_j$. Nous avons donc $\gamma_{ij}\bar{x}_j t_{ik} = z_{ijk} \leq u_j t_{ik}$.
- Les contraintes (13), $z_{ijk} \leq x_j$:
  Nous avons $z_{ijk} = \gamma_{ij}\bar{x}_j t_{ik} = \gamma_{ij}\bar{x}_j\bar{x}_i/\bar{u}_i = \bar{y}_{ij}/\bar{u}_i$, par les contraintes (22), $\bar{y}_{ij}/\bar{u}_i \leq \bar{x}_j u_i/\bar{u}_i \leq \bar{x}_j$, car $u_i/\bar{u}_i \leq 0$.
- Les contraintes (14), $z_{ijk} \geq x_j - u_j(1 - t_{ik})$:
  Nous avons $z_{ijk} = \gamma_{ij}\bar{x}_j t_{ik} = \bar{y}_{ij}/\bar{u}_i$, par les contraintes (17) $\bar{y}_{ij}/\bar{u}_i \geq (u_j\bar{x}_i + u_i\bar{x}_j - u_i u_j)/\bar{u}_i \geq t_{ik}u_j + u_i/\bar{u}_i(\bar{x}_j - u_j) \geq t_{ik}u_j + \bar{x}_j - u_j$, car $u_i/\bar{u}_i \leq 0$.

2. Montrons que pour tout $(\bar{x}, \bar{y}, \bar{z}, \bar{t})$ appartenant à $\overline{P}_{xyzt}$, $(\bar{x}, \bar{y})$ appartiennent à P_{xy}.
   Soit $(\bar{x}, \bar{y}, \bar{z}, \bar{t}) \in \overline{P}_{xyzt}$. La solution $(\bar{x}, \bar{y})$ satisfait les contraintes (20), (17), (18) et (16).
   Montrons maintenant que dans une telle solution, les contraintes (21) et (22) sont satisfaites :
   - Les contraintes (21), $y_{ij} \leq u_j x_i$:
     Par les contraintes (10), (11) et (12), nous avons $\bar{y}_{ij} = \bar{y}_{ji} = \sum_{k=0}^{\lfloor log(u_i) \rfloor} 2^k \bar{z}_{ijk} \leq \sum_{k=0}^{\lfloor log(u_i) \rfloor} 2^k u_j \bar{t}_{ik} = u_j\bar{x}_i$
   - Les contraintes (22), $y_{ij} \leq u_i x_j$:
     Par les contraintes (16) et (21), nous avons $\bar{y}_{ij} = \bar{y}_{ji} \leq u_i\bar{x}_j$.

□

Soit $(P_{\alpha,\beta})$ le programme suivant :

$$(P_{\alpha,\beta}) \begin{cases} Min & f_{\alpha,\beta}(x,y) = f(x) + \alpha\sum_{r=1}^{m}(\sum_{i=1}^{n} a_{ri}x_i - b_r)^2 + \sum_{i=1}^{n}\sum_{j=1}^{n}\beta_{ij}(x_i x_j - y_{ij}) \\ s.c. & (1)(2) \\ & x, y \in P_{xy} \end{cases}$$

Corollaire 4.3.1 *Minimiser $f_{\alpha,\beta}(x,y)$ sur le polyèdre $\overline{P}_{xyzt}$ ou sur le polyèdre P_{xy} est équivalent, i.e. résoudre $(\overline{QP}_{\alpha,\beta})$ ou $(P_{\alpha,\beta})$ est équivalent.*

Ainsi, le programme quadratique en variables continues $(\overline{QP}_{\alpha,\beta})$ a une propriété intéressante : toutes les variables z et t et les contraintes qui les contiennent peuvent être remplacées par les contrainte (20)-(22). $(\overline{QP}_{\alpha,\beta})$ est équivalent à $(P_{\alpha,\beta})$ qui est nettement plus petit.

Dans la section suivante, nous caractérisons les valeurs de α et β qui maximisent la valeur de $(\overline{QP}_{\alpha,\beta})$, et donc celle de $(P_{\alpha,\beta})$.

4.3.3 Calcul de la meilleure convexification au sein de ce schéma : la méthode IQCR

Dans cette section, nous montrons comment calculer par programmation semi-définie les valeurs de α^* et β^* qui rendent $f_{\alpha^*,\beta^*}(x,y)$ convexe tout en maximisant la valeur de la relaxation continue de (QP_{α^*,β^*}). Comme cela a été montré dans le Corollaire 4.3.1, cela revient à résoudre le problème suivant :

$$(CP): \max_{\substack{\alpha\in\mathbb{R},\quad \beta\in\mathbf{S}_n \\ Q_{\alpha,\beta}\succeq 0}} \{v(P_{\alpha,\beta})\}$$

où $v(P_{\alpha,\beta})$ est la valeur de la solution optimale de $(P_{\alpha,\beta})$ et $Q_{\alpha,\beta}$ le Hessien de $f_{\alpha,\beta}(x,y)$, i.e. $Q_{\alpha,\beta} = Q + \alpha A^T A + \beta$. Nous pouvons réécrire (CP) comme suit :

$$(CP): \max_{\substack{\alpha\in\mathbb{R},\quad \beta\in\mathbf{S}_n \\ Q_{\alpha,\beta}\succeq 0}} \min_{\substack{(1)(2) \\ (x,y)\in P_{x,y}}} \{f_{\alpha,\beta}(x,y)\}$$

Théorème 4.3.2 *Soit (SDP) le programme semi-défini suivant :*

$$(SDP)\begin{cases} Min & f(X,x) = \sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij}X_{ij} + \sum_{i=1}^{n} c_i x_i & & \\ s.c. & (1)(2) & & \\ & \sum_{r=1}^{m}(\sum_{i=1}^{n}(\sum_{j=1}^{n} a_{ri}a_{rj}X_{ij} - 2a_{ri}b_r x_i)) = -\sum_{r=1}^{m} b_r^2 & & (23) \\ & X_{ij} \le u_j x_i & (i,j)\in I^2 & (24) \\ & X_{ij} \le u_i x_j & (i,j)\in I^2 & (25) \\ & -X_{ij} \le -u_j x_i - u_i x_j + u_i u_j & (i,j)\in I^2 & (26) \\ & -X_{ij} \le 0 & (i,j)\in I^2 & (27) \\ & -X_{ii} \le -x_i & i\in I & (28) \\ & \begin{pmatrix} 1 & x \\ x^T & X \end{pmatrix} \succeq 0 & & (29) \\ & x\in\mathbb{R}^n \quad X\in\mathbf{S}^n & & (30) \end{cases}$$

Une solution optimale (α^*, β^*) de (CP) peut être déduite des valeurs optimales des variables duales de (SDP). Le paramètre optimal α^* est la valeur optimale de la variable duale associée à la contrainte (23). Les paramètres optimaux sont calculés par $\beta^*_{ij} = \beta^{1*}_{ij} + \beta^{2*}_{ij} - \beta^{3*}_{ij} - \beta^{4*}_{ij}$, $(i \neq j)$, et $\beta^*_{ii} = \beta^{1*}_{ii} + \beta^{2*}_{ii} - \beta^{3*}_{ii} - \beta^{4*}_{ii} - \beta^{5*}_{ii}$, où β^{1*}_{ij}, β^{2*}_{ij}, β^{3*}_{ij}, β^{4*}_{ij} $(i, j \in I)$ sont les valeurs optimales des variables duales associées aux contraintes (24), (25), (26) et (27) respectivement, et β^{5*}_{ii} $(i \in I)$ sont les valeurs optimales des variables duales associées aux contraintes (28).

Preuve.

1. Montrons d'abord que $v(CP) \leq v(SDP)$. Pour cela, comme (CP) est un problème de maximisation, nous allons montrer que pour toute solution réalisable $(\bar{\alpha}, \bar{\beta})$ de (CP), c'est à dire pour tout $(\bar{\alpha}, \bar{\beta})$ tels que $Q_{\bar{\alpha},\bar{\beta}} \succeq 0$, nous avons $v(P_{\bar{\alpha},\bar{\beta}}) \leq v(SDP)$.
   Soit $(\bar{\alpha}, \bar{\beta})$ tels que $Q_{\bar{\alpha},\bar{\beta}} \succeq 0$, nous avons bien $(\bar{\alpha}, \bar{\beta})$ qui est une solution réalisable de (CP), et soit $(\bar{X}, \bar{x})$ un solution réalisable de (SDP). A partir de cette solution nous allons construire une solution réalisable (x, y) de $(P_{\bar{\alpha},\bar{\beta}})$, telle que $f_{\bar{\alpha},\bar{\beta}}(x, y) \leq f(\bar{X}, \bar{x})$.
   Nous prenons $x = \bar{x}$ et $y = \bar{X}$. Les contraintes (1) et (2) sont satisfaites.
   Comme $0 \leq \bar{x}_i \leq u_i$, $y_{ij} = \bar{X}_{ij}$, et que $\bar{X}_{ij}$ satisfait les contraintes (24), (25), (26), (27) et (28), les contraintes (20), (21), (22), (17) et (18) sont satisfaites. De plus, comme $\bar{X} \in \mathbf{S}^n$, nous avons $\bar{X}_{ij} = \bar{X}_{ji}$, et les contraintes (16) sont satisfaites.

   Montrons maintenant que $E = f(\bar{X}, \bar{x}) - f_{\bar{\alpha},\bar{\beta}}(x, y) \geq 0$:

$$\begin{aligned}
E &= \sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij}\bar{X}_{ij} + \sum_{i=1}^{n} c_i\bar{x}_i - \sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij}\bar{x}_i\bar{x}_j - \sum_{i=1}^{n} c_i\bar{x}_i - \sum_{i=1}^{n}\sum_{j=1}^{n}\bar{\beta}_{ij}(\bar{x}_i\bar{x}_j - \bar{X}_{ij}) \\
&\quad -\bar{\alpha}\sum_{r=1}^{m}(\sum_{i=1}^{n} a_{ri}\bar{x}_i - b_r)^2 \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij}(\bar{X}_{ij} - \bar{x}_i\bar{x}_j) - \sum_{i=1}^{n}\sum_{j=1}^{n}\bar{\beta}_{ij}(\bar{x}_i\bar{x}_j - \bar{X}_{ij}) - \bar{\alpha}\sum_{r=1}^{m}(\sum_{i=1}^{n} a_{ri}\bar{x}_i - b_r)^2 \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n}(q_{ij} + \bar{\beta}_{ij})(\bar{X}_{ij} - \bar{x}_i\bar{x}_j) + \bar{\alpha}\sum_{r=1}^{m}(\sum_{i=1}^{n} 2a_{ri}b_r\bar{x}_i - b_r^2) - \bar{\alpha}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{r=1}^{m}\bar{x}_i\bar{x}_j
\end{aligned}$$

Par les contraintes (23), nous avons

$$\begin{aligned} E &= \sum_{i=1}^{n}\sum_{j=1}^{n}(q_{ij}+\bar{\beta}_{ij})(\bar{X}_{ij}-\bar{x}_i\bar{x}_j)+\bar{\alpha}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{r=1}^{m}a_{ri}a_{rj}\bar{X}_{ij}-\bar{\alpha}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{r=1}^{m}\bar{x}_i\bar{x}_j \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n}(q_{ij}+\bar{\beta}_{ij}+\bar{\alpha}\sum_{r=1}^{m}a_{ri}a_{rj})(\bar{X}_{ij}-\bar{x}_i\bar{x}_j) \\ &= \langle Q+\bar{\beta}+\bar{\alpha}A^TA, \bar{X}-\bar{x}\bar{x}^T\rangle \\ &= \langle Q_{\bar{\alpha},\bar{\beta}}, \bar{X}-\bar{x}\bar{x}^T\rangle \geq 0 \end{aligned}$$

puisque les deux matrices $Q_{\bar{\alpha},\bar{\beta}}$ et $\bar{X}-\bar{x}\bar{x}^T$ sont semi-définies positives.

Donc, nous avons $\forall(\bar{\alpha},\bar{\beta})$ tels que $Q_{\bar{\alpha},\bar{\beta}} \succeq 0$, $v(P_{\bar{\alpha},\bar{\beta}}) \leq v(SDP)$, ou de manière équivalente

$$v(CP) \leq v(SDP)$$

2. Ensuite, montrons que $v(CP) \geq v(SDP)$ ou de manière équivalente que $v(CP) \geq v(DSDP)$ avec $(DSDP)$ le dual de (SDP).
   Nous montrons que pour toute solution duale réalisable de (SDP), nous pouvons construire une solution réalisable de (CP) dont la valeur de la fonction objectif est plus grande.
   Le problème suivant $(DSDP)$ est le dual de (SDP) :

$$(DSDP)\begin{cases} Max & g(\alpha,\beta,\rho,\sigma)=\alpha\sum_{r=1}^{m}b_r^2-\sum_{i=1}^{n}\sum_{j=1}^{n}\beta_{ij}^3u_iu_j-\sum_{r=1}^{m}\rho_rb_r-\sum_{s=1}^{p}\sigma_se_s & \\ S.c. & Q+\alpha A^TA+\beta\succeq 0 & (31)\\ & c_i-2\alpha\sum_{r=1}^{m}a_{ri}b_r-\sum_{j=1}^{n}(\beta_{ij}^1+\beta_{ij}^2-2\beta_{ij}^3)u_j+\beta_i^5+\sum_{r=1}^{m}a_{ri}\rho_r+\sum_{s=1}^{p}d_{si}\sigma_s\geq 0\ i\in I & (32)\\ & \beta=\beta^1+\beta^2-\beta^3-\beta^4-diag(\beta^5) & (33)\\ & \alpha\in\mathbb{R},\ \rho\in\mathbb{R}^m,\ \sigma\in\mathbb{R}_+^p\ ,\beta\in\mathbf{S}_n,\ \beta^1\beta^2,\beta^3,\beta^4\in\mathbf{S}_n^+,\ \beta^5\in\mathbb{R}^n & (34)\end{cases}$$

où ρ et σ sont les variables duales associées aux contraintes (1) et (2) respectivement.

Soit $(\bar{\alpha},\bar{\beta},\bar{\rho},\bar{\sigma})$ une solution réalisable de $(DSDP)$, par (31), nous avons $Q_{\bar{\alpha},\bar{\beta}}\succeq 0$. Donc $(\bar{\alpha},\bar{\beta})$ est aussi une solution réalisable de (CP), dont la valeur de la fonction objectif est $v(P_{\bar{\alpha},\bar{\beta}})$.

Montrons que $v(P_{\bar{\alpha},\bar{\beta}}) \geq g(\bar{\alpha},\bar{\beta},\bar{\rho},\bar{\sigma})$. Pour cela, comme $(P_{\bar{\alpha},\bar{\beta}})$ est un problème de minimisation, nous prouvons que la valeur de $g(\bar{\alpha},\bar{\beta},\bar{\rho},\bar{\sigma})$ est inférieure à la valeur de n'importe

quelle solution de $(P_{\bar{\alpha},\bar{\beta}})$. Soit $(\bar{x},\bar{y})$ une solution réalisable de $(P_{\bar{\alpha},\bar{\beta}})$.

$$\begin{aligned} f_{\bar{\alpha},\bar{\beta}}(\bar{x},\bar{y}) &= \sum_{i=1}^{n}\sum_{j=1}^{n}(q_{ij}+\bar{\alpha}\sum_{r=1}^{m}a_{ri}a_{rj}+\bar{\beta}_{ij})\bar{x}_i\bar{x}_j+\sum_{i=1}^{n}(c_i-2\bar{\alpha}\sum_{r=1}^{m}a_{ri}b_r)\bar{x}_i+\bar{\alpha}\sum_{r=1}^{m}b_r^2 \\ &\quad -\sum_{i=1}^{n}\sum_{j=1}^{n}\bar{\beta}_{ij}\bar{y}_{ij} \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n}(q_{ij}+\bar{\alpha}\sum_{r=1}^{m}a_{ri}a_{rj}+\bar{\beta}_{ij})\bar{x}_i\bar{x}_j+\sum_{i=1}^{n}(c_i-2\bar{\alpha}\sum_{r=1}^{m}a_{ri}b_r)\bar{x}_i+\bar{\alpha}\sum_{r=1}^{m}b_r^2 \\ &\quad -\sum_{i=1}^{n}(\sum_{j=1}^{n}(\bar{\beta}_{ij}^1+\bar{\beta}_{ij}^2-\bar{\beta}_{ij}^3-\bar{\beta}_{ij}^4)\bar{y}_{ij}-\bar{\beta}_i^5\bar{y}_{ii}) \end{aligned}$$

Par les contraintes (21), (22) , (17) et (18), et comme $\bar{\beta}_{ij}^1$, $\bar{\beta}_{ij}^2$, $\bar{\beta}_{ij}^3$, $\bar{\beta}_{ij}^4$, $\bar{\beta}_i^5 \geq 0$, et $u_i \geq 0$, nous avons :

$$\begin{aligned} f_{\bar{\alpha},\bar{\beta}}(\bar{x},\bar{y}) &\geq \sum_{i=1}^{n}\sum_{j=1}^{n}(q_{ij}+\bar{\alpha}\sum_{r=1}^{m}a_{ri}a_{rj}+\bar{\beta}_{ij})\bar{x}_i\bar{x}_j+\sum_{i=1}^{n}(c_i-2\bar{\alpha}\sum_{r=1}^{m}a_{ri}b_r)\bar{x}_i+\bar{\alpha}\sum_{r=1}^{m}b_r^2 \\ &\quad -\sum_{i=1}^{n}\sum_{j=1}^{n}\bar{\beta}_{ij}^1u_j\bar{x}_i-\sum_{i=1}^{n}\sum_{j=1}^{n}\bar{\beta}_{ij}^2u_i\bar{x}_j+\sum_{i=1}^{n}\sum_{j=1}^{n}\bar{\beta}_{ij}^3(u_j\bar{x}_i+u_i\bar{x}_j-u_iu_j)+\sum_{i=1}^{n}\bar{\beta}_i^5\bar{x}_i \\ &\geq \sum_{i=1}^{n}\sum_{j=1}^{n}(q_{ij}+\bar{\alpha}\sum_{r=1}^{m}a_{ri}a_{rj}+\bar{\beta}_{ij})\bar{x}_i\bar{x}_j+\bar{\alpha}\sum_{r=1}^{m}b_r^2-\sum_{i=1}^{n}\sum_{j=1}^{n}\bar{\beta}_{ij}^3u_iu_j \\ &\quad +\sum_{i=1}^{n}(c_i-2\bar{\alpha}\sum_{r=1}^{m}a_{ri}b_r-\sum_{j=1}^{n}\bar{\beta}_{ij}^1u_j-\sum_{j=1}^{n}\bar{\beta}_{ij}^2u_j+2\sum_{j=1}^{n}\bar{\beta}_{ij}^3u_j+\bar{\beta}_i^5)\bar{x}_i \end{aligned}$$

Comme $(\bar{\alpha},\bar{\beta},\bar{\rho},\bar{\sigma})$ est une solution réalisable de $(DSDP)$, par les contraintes (32), nous avons $\forall i \in I$

$$c_i-2\bar{\alpha}\sum_{r=1}^{m}a_{ri}b_r-\sum_{j=1}^{n}\bar{\beta}_{ij}^1u_j-\sum_{j=1}^{n}\bar{\beta}_{ij}^2u_j+2\sum_{j=1}^{n}\bar{\beta}_{ij}^3u_j+\bar{\beta}_i^5 \geq -\sum_{r=1}^{m}a_{ri}\bar{\rho}_r-\sum_{s=1}^{p}d_{si}\bar{\sigma}_s$$

Et comme $x_i \geq 0$, nous avons :

$$\begin{aligned} f_{\bar{\alpha},\bar{\beta}}(\bar{x},\bar{y}) &\geq \sum_{i=1}^{n}\sum_{j=1}^{n}(q_{ij}+\bar{\alpha}\sum_{r=1}^{m}a_{ri}a_{rj}+\bar{\beta}_{ij})\bar{x}_i\bar{x}_j+\bar{\alpha}\sum_{r=1}^{m}b_r^2-\sum_{i=1}^{n}\sum_{j=1}^{n}\bar{\beta}_{ij}^3u_iu_j \\ &\quad +\sum_{i=1}^{n}(-\sum_{r=1}^{m}a_{ri}\bar{\rho}_r-\sum_{s=1}^{p}d_{si}\bar{\sigma}_s)\bar{x}_i \end{aligned}$$

De plus, par les contraintes (1) et (2)

$$\begin{aligned} f_{\bar{\alpha},\bar{\beta}}(\bar{x},\bar{y}) &\geq \sum_{i=1}^{n}\sum_{j=1}^{n}(q_{ij}+\bar{\alpha}\sum_{r=1}^{m}a_{ri}a_{rj}+\bar{\beta}_{ij})\bar{x}_i\bar{x}_j \\ &\quad +\bar{\alpha}\sum_{r=1}^{m}b_r^2-\sum_{i=1}^{n}\sum_{j=1}^{n}\bar{\beta}_{ij}^3u_iu_j-\sum_{r=1}^{m}\bar{\rho}_r b_r-\sum_{s=1}^{p}e_s\bar{\sigma}_s \\ &\geq \sum_{i=1}^{n}\sum_{j=1}^{n}(q_{ij}+\bar{\alpha}\sum_{r=1}^{m}a_{ri}a_{rj}+\bar{\beta}_{ij})\bar{x}_i\bar{x}_j+g(\bar{\alpha},\bar{\beta},\bar{\rho},\bar{\sigma}) \end{aligned}$$

Finalement, par les contraintes (31) de $(DSDP)$, nous avons :

$$\sum_{i=1}^{n}\sum_{j=1}^{n}(q_{ij}+\bar{\alpha}\sum_{r=1}^{m}a_{ri}a_{rj}+\bar{\beta}_{ij})\geq 0$$

et donc nous avons :

$$f_{\bar{\alpha},\bar{\beta}}(\bar{x},\bar{y})\geq g(\bar{\alpha},\bar{\beta},\bar{\rho},\bar{\sigma})$$

Ainsi, pour $(\bar{\alpha},\bar{\beta},\bar{\rho},\bar{\sigma})$ solution réalisable de $(DSDP)$, et donc telle que $Q_{\bar{\alpha},\bar{\beta}}\succeq 0$, nous avons $\forall(\bar{x},\bar{y})$ solution réalisable de $(P_{\bar{\alpha},\bar{\beta}})$, $f_{\bar{\alpha},\bar{\beta}}(\bar{x},\bar{y})\geq g(\bar{\alpha},\bar{\beta},\bar{\rho},\bar{\sigma})$ et donc,

$$v(CP)\geq v(DSDP)=v(SDP)$$

Pour résumer, nous avons montré dans une première partie que pour toute solution $(\bar{\alpha},\bar{\beta})$ de (CP), la valeur $v(P_{\bar{\alpha},\bar{\beta}})$ est inférieure ou égale à la valeur optimale de (SDP), $v(SDP)$. Donc, c'est aussi vrai pour (α^{*c},β^{*c}) une solution optimale de (CP), et donc nous avons

$$v(CP)=v(P_{\alpha^{*c},\beta^{*c}})\leq v(SDP)$$

Dans une deuxième partie, nous avons montré qu'étant donné une solution réalisable $(\bar{\alpha},\bar{\beta},\bar{\rho},\bar{\sigma})$ de $(DSDP)$, $(\bar{\alpha},\bar{\beta})$ est une solution réalisable de (CP) dont la valeur, $v(P_{\bar{\alpha},\bar{\beta}})$, est telle que $v(P_{\bar{\alpha},\bar{\beta}})\geq v(DSDP)$. Donc, c'est toujours vrai pour $(\alpha^{*d},\beta^{*d},\rho^{*d},\sigma^{*d})$ une solution optimale de $(DSDP)$, et donc nous avons $v(P_{\alpha^{*d},\beta^{*d}})\geq v(DSDP)$. De plus, comme (α^{*d},β^{*d}) est une solution réalisable de (CP), et comme (CP) est un problème de maximisation, nous avons :

$$v(CP)\geq v(P_{\alpha^{*d},\beta^{*d}})\geq v(DSDP)$$

Pour conclure, comme $v(SDP)=v(DSDP)$, nous déduisons que :

$$v(CP)\geq v(P_{\alpha^{*d},\beta^{*d}})\geq v(DSDP)=v(SDP)\geq v(P_{\alpha^{*c},\beta^{*c}})=v(CP)$$

Et donc la solution optimale de (CP) est $(\alpha^{*d}, \beta^{*d})$ donnée par la solution optimale de $(DSDP)$.
□

A partir du Théorème 4.3.2, nous construisons un algorithme général de résolution exacte de problèmes quadratiques non convexes et à variables entières basé sur le reformulation `IQCR` qui est présenté dans la Figure 4.1.

`IQCR (Integer Quadratic Convex Reformulation)`
1. Résoudre le programme semi-défini (SDP)
2. Déduire α^* et β^* (Théorème 4.3.2)
3. Résoudre le programme (QP_{α^*,β^*}) par un solveur MIQP.

FIG. 4.1 – Algorithme de résolution de (QP) basé sur la reformulation `IQCR`

Nous pouvons remarquer que la relaxation continue de (QP_{α^*,β^*}), $(\overline{QP}_{\alpha^*,\beta^*})$ est un programme convexe dont la valeur optimale est égale à la valeur optimale de (SDP).

Remarque 4.3.1 *A partir de la preuve du Théorème 4.3.2, il découle que pour toute solution réalisable de $(DSDP)$ dont la valeur de la fonction objectif vaut Δ, nous pouvons déduire une reformulation convexe dont la valeur de la borne obtenue par relaxation continue n'est pas plus petite que Δ.*

L'intérêt de la Remarque 4.3.1 est que le temps de résolution des solveurs de programmation semi-défini étant souvent relativement lent, si on arrête le solveur après un temps fixé, une solution réalisable $(\bar{\alpha}, \bar{\beta})$ de $(DSDP)$ fournit des valeurs pour les paramètres α et β qui rendent convexe la fonction $f_{\bar{\alpha},\bar{\beta}}(x, y)$.

4.3.4 Extension de `IQCR` aux problèmes mixtes entiers : la méthode `IQCRs`

Dans cette partie, nous adaptons la méthode `IQCR` aux problèmes mixtes entiers, en suivant les mêmes étapes de raisonnement que pour le cas des problèmes purement en nombres entiers. Soit (MQP) le problème suivant :

$$(MQP)\begin{cases} Min & h(x) & & \\ s.c. & Ax = b & & (35)\\ & Dx \leq e & & (36)\\ & 0 \leq x_i \leq u_i & i \in I & (37)\\ & 0 \leq x_i \leq u_i & i \in J & (38)\\ & x_i \in \mathbb{N} & i \in I & (39)\\ & x_i \in \mathbb{R} & i \in J & (40) \end{cases}$$

où $I = \{1, \ldots, n\}$ est le sous-ensemble des indices des variables entières, $J = \{n+1, \ldots, N\}$ est le sous-ensemble des indices des variables réelles, et

$$h(x) = x^T Q x + c^T x = \sum_{(i,j)\in I^2} q_{ij} x_i x_j + \sum_{(i,j)\in I\times J} 2q_{ij} x_i x_j + \sum_{(i,j)\in J^2} q_{ij} x_i x_j + \sum_{i\in I\cup J} c_i x_i$$

et $Q \in \mathbf{S}_N$, $c \in \mathbb{R}^N$, $A \in \mathbf{M}_{m,N}$, $b \in \mathbb{R}^m$, $D \in \mathbf{M}_{p,N}$, $e \in \mathbb{R}^p$, $u \in \mathbb{N}^N$.

Nous supposons ici que la sous-fonction constituée des produits de variables réelles de $h(x)$, $\sum_{(i,j)\in J^2} q_{ij} x_i x_j$, est convexe. Cette hypothèse nous permet d'être sûr de l'existence d'une reformulation convexe triviale de (MQP).

Une reformulation convexe de (MQP)

Nous définissons maintenant l'ensemble des couples d'indices suivant : $P = \{(i,j) \in I^2 \cup (I \times J) \cup (J \times I)\}$, et la fonction perturbée suivante :

$$h_{\alpha,\beta}(x,y) = h(x) + \sum_{(i,j)\in P} \beta_{ij}(x_i x_j - y_{ij}) + \alpha \sum_{r=1}^{m} (\sum_{i=1}^{N} a_{ri} x_i - b_r)^2$$

Pour simplifier la notation, nous considérons $\beta \in \mathbf{S}_N$, avec $\beta_{ij} = 0$, $\forall (i,j) \in J^2$.

En suivant les mêmes idées que dans la Section 4.3.1, nous considérons le problème suivant qui est équivalent à (MQP) :

$$(MQP_{\alpha,\beta}) \begin{cases} Min & h_{\alpha,\beta}(x,y) \\ s.c. & (35)(36) \\ & (x,y) \in MP_{xyzt} \end{cases}$$

où MP_{xyzt} est l'ensemble suivant :

$$MP_{xyzt}\left\{ x,y,z,t : \begin{array}{lll} (37)(38) & & \\ x_i = \sum_{k=0}^{\lfloor log(u_i)\rfloor} 2^k t_{ik} & i \in I & (41) \\ y_{ij} = \sum_{k=0}^{\lfloor log(u_i)\rfloor} 2^k z_{ijk} & (i,j) \in I \times I \cup J & (42) \\ z_{ijk} \leq u_j t_{ik} & (i,k) \in E,\ j \in I \cup J & (43) \\ z_{ijk} \leq x_j & (i,k) \in E,\ j \in I \cup J & (44) \\ z_{ijk} \geq x_j - u_j(1-t_{ik}) & (i,k) \in E,\ j \in I \cup J & (45) \\ z_{ijk} \geq 0 & (i,k) \in E,\ j \in I \cup J & (46) \\ y_{ij} = y_{ji} & (i,j) \in P & (47) \\ y_{ij} \geq x_i u_j + x_j u_i - u_i u_j & (i,j) \in P & (48) \\ y_{ii} \geq x_i & i \in I & (49) \\ y_{ij} \leq u_i x_j & (i,j) \in I \times J & (50) \\ t_{ik} \in \{0,1\} & (i,k) \in E & (51) \end{array} \right.$$

où $E = \{(i,k) : i = 1,\ldots,n,\ k = 0,\ldots \lfloor log(u_i)\rfloor\}$.

Remarquons que la différence avec le cas où toutes les variables sont entières est que nous ne perturbons pas $h(x)$ avec les expressions $\beta_{ij}(x_i x_j - y_{ij})$ lorsque $(i,j) \in J^2$, c'est à dire pour les produits de deux variables réelles.

Il est toujours possible de choisir α et β tels que la fonction $h_{\alpha,\beta}(x)$ soit convexe. Par exemple, nous pouvons prendre α égal à n'importe quel réel positif, $\beta_{ij} = -q_{ij},\ (i,j) \in (I \times J) \cup (J \times I)$, $\beta_{ij} = 0,\ (i,j) \in I^2,\ i \neq j$, et $\beta_{ii} = -\lambda_{min}(Q_I),\ i \in I$ où Q_I est la sous matrice de Q définie par $(q_{ij})_{(i,j)\in I^2}$. Remarquons que cette reformulation convexe triviale revient à convexifier les produits de variables entières et à linéariser les produits mixtes entiers.

Ici encore, nous sommes intéressés par la relaxation continue de $(MQP_{\alpha,\beta})$, qui est le problème suivant :

$$(MQP_{\alpha,\beta})\left\{ \begin{array}{ll} Min & h_{\alpha,\beta}(x,y) \\ s.c. & (35)(36) \\ & (x,y) \in \overline{MP}_{xyzt} \end{array} \right.$$

où $\overline{MP}_{xyzt}$ est le polyèdre obtenu par relaxation continue de MP_{xyzt}, c'est à dire en remplaçant les contraintes (51) par

$$0 \leq t_{ik} \leq 1 \quad (51')$$

Nous pouvons facilement adapter le Théorème 4.3.1 aux problèmes mixtes entiers.

Théorème 4.3.3 *La projection de $\overline{MP}_{xyzt}$ sur les variables x et y est le polyèdre MP_{xy} obtenu à partir de P_{xy} en remplaçant $(i,j) \in I^2$ par $(i,j) \in P$, c'est à dire :*

$$MP_{xy} \begin{cases} & (38)(47)(48)(49) & & \\ x,y: & y_{ij} \geq 0 & (i,j) \in P & (52) \\ & y_{ij} \leq u_j x_i & (i,j) \in P & (53) \\ & y_{ij} \leq u_i x_j & (i,j) \in P & (54) \end{cases}$$

Preuve.

1. Nous montrons d'abord que pour tout (x,y) dans MP_{xy}, il existe z,t tels que (x,y,z,t) appartient à $\overline{MP}_{xyzt}$.

   Soit $(x,y) \in P_{xy}$, et γ_{ij} un réel positif tel que $y_{ij} = \gamma_{ij} x_i x_j$. Soit $\bar{u}_i = \sum_{k=0}^{\lfloor log(u_i) \rfloor} 2^k$. Prenons $t_{ik} = x_i/\bar{u}_i$ et $z_{ijk} = \gamma_{ij} t_{ik} x_j$. De manière évidente les contraintes (37), (38), (41), (42), (46), (47), (48), (49), (50) et (51') sont satisfaites. Nous avons donc à montrer que les contraintes (43), (44) et (45) sont satisfaites :

   - Les contraintes (43) $z_{ijk} \leq u_j t_{ik}$, $(i,k) \in E, j \in I \cup J$:

     Si $x_i = 0$, alors $t_{ik} = 0$, et $z_{ijk} = 0$.

     Si $x_i > 0$, nous savons que $z_{ijk} = \gamma_{ij} x_j t_{ik} = (y_{ij}/x_i) t_{ik}$. Par (53), nous avons $z_{ijk} \leq (u_j x_i t_{ik})/x_i = u_j t_{ik}$.
   - Les contraintes (44) $z_{ijk} \leq x_j$, $(i,k) \in E, j \in I \cup J$:

     $z_{ijk} = \gamma_{ij} t_{ik} x_j = \gamma_{ij} x_j x_i/\bar{u}_i = y_{ij}/\bar{u}_i$. Par (54), nous avons $z_{ijk} \leq x_j u_i/\bar{u}_i \leq x_j$ puisque $u_i/\bar{u}_i \leq 1$.
   - Les contraintes (45) $z_{ijk} \geq x_j - u_j(1 - t_{ik})$, $(i,k) \in E, j \in I \cup J$:

     $z_{ijk} = \gamma_{ij} t_{ik} x_j = y_{ij}/\bar{u}_i$. Par (48), nous avons $z_{ijk} \geq (u_j x_i + u_i x_j - u_i u_j)/\bar{u}_i = t_{ik} u_j + u_i/\bar{u}_i (x_j - u_j) \geq t_{ik} u_j + x_j - u_j$ puisque $u_i/\bar{u}_i \leq 1$ et $x_j - u_j \leq 0$.

2. Montrons maintenant que pour tout (x,y,z,t) de $\overline{MP}_{xyzt}$, (x,y) appartient à MP_{xy}.

   De manière évidente les contraintes (38), (47), (48), (49) and (50) sont satisfaites. Nous avons maintenant à prouver que les contraintes (53) et (54) sont satisfaites.

   - Les contraintes (17) : $y_{ij} \leq u_j x_i$, $(i,j) \in P$

     (a) Nous prouvons que $y_{ij} \leq u_j x_i$, $(i,j) \in I \times I \cup J$:

     Par les contraintes (41), (42) and (43), nous avons $y_{ij} = \sum_{k=0}^{\lfloor log(u_i) \rfloor} 2^k z_{ijk} \leq \sum_{k=0}^{\lfloor log(u_i) \rfloor} 2^k u_j t_{ik} = u_j x_i$.

     (b) Nous prouvons que $y_{ij} \leq u_j x_i$, $(i,j) \in J \times I$:

Par les contraintes (50), nous avons :
$y_{ij} \leq u_i x_j$, $(i,j) \in I \times J$
Donc $y_{ji} \leq u_j x_i$, $(i,j) \in J \times I$
Et puisque $y_{ij} = y_{ji}$ par les contraintes (47), nous avons $y_{ij} = y_{ji} \leq u_j x_i$, $(i,j) \in J \times I$

– Les contraintes (54) : $y_{ij} \leq u_i x_j$, $(i,j) \in P$
Par les contraintes (53) nous savons que $y_{ij} \leq u_j x_i$ et donc que $y_{ji} \leq u_i x_j$. Par les contraintes (47), nous avons $y_{ij} = y_{ji} \leq u_i x_j$.

□

Soit $(MP_{\alpha,\beta})$ le problème suivant :

$$(MP_{\alpha,\beta}) \begin{cases} Min & h_{\alpha,\beta}(x,y) \\ s.c. & (35)(36) \\ & (x,y) \in MP_{xy} \end{cases}$$

Nous pouvons donc déduire le corollaire suivant :

Corollaire 4.3.2 *Minimiser $h_{\alpha,\beta}(x,y)$ sur le polyèdre MP_{xyzt} ou sur le polyèdre MP_{xy} est équivalent, i.e. résoudre $(\overline{MQP}_{\alpha,\beta})$ ou $(MP_{\alpha,\beta})$ est équivalent.*

Calcul de la meilleure convexification : la méthode `IQCRs`

Comme pour le cas où les variables sont purement en nombres entiers, nous voulons calculer les valeurs de α et β qui maximisent la valeur optimale de la relaxation continue de $(MQP_{\alpha,\beta})$, et donc la valeur de $(MP_{\alpha,\beta})$. Plus formellement, nous cherchons à résoudre le problème suivant :

$$(MCP) : \max_{\substack{\alpha \in \mathbb{R}, \quad \beta \in \mathbf{S}_N \\ Q_{\alpha,\beta} \succeq 0}} \min_{\substack{(35)(36) \\ (x,y) \in MP_{xy}}} \{h_{\alpha,\beta}(x,y)\}$$

où $Q_{\alpha,\beta}$ est le Hessien de $h_{\alpha,\beta}(x,y)$.Rappelons que $\beta_{ij} = 0$, $\forall (i,j) \in J^2$.

Théorème 4.3.4 *Une solution optimale (α^*, β^*) de (MCP) peut être déduite des valeurs optimales des variables duales d'une relaxation semi-définie de (MQP), qui est le programme sui-*

vant :

$$(MSDP)\begin{cases} Min & h(X,x)=\sum_{i=1}^{N}\sum_{j=1}^{N}q_{ij}X_{ij}+\sum_{i=1}^{N}c_ix_i & & \\ s.c. & (35)(36)(38) & & \\ & \sum_{r=1}^{m}(\sum_{i=1}^{N}(\sum_{j=1}^{N}a_{ri}a_{rj}X_{ij}-2a_{ri}b_rx_i))=-\sum_{r=1}^{m}b_r^2 & & (55)\\ & X_{ij}\le u_jx_i & (i,j)\in P & (56)\\ & X_{ij}\le u_ix_j & (i,j)\in P & (57)\\ & -X_{ij}\le -u_jx_i-u_ix_j+u_iu_j & (i,j)\in P & (58)\\ & -X_{ij}\le 0 & (i,j)\in P & (59)\\ & -X_{ii}\le -x_i & i\in I & (60)\\ & \begin{pmatrix}1 & x\\ x^T & X\end{pmatrix}\succeq 0 & & (61)\\ & x\in\mathbb{R}^N \quad X\in\mathbf{S}^N & & (62)\end{cases}$$

Les coefficients optimaux α^* et β^* sont déduits de la même façon que dans le Théorème 4.3.2. Plus précisément, une solution optimale (α^*,β^*) de (MCP) peut être déduite des valeurs optimales des variables duales de $(MSDP)$. Le paramètre optimal α^* est la valeur optimale de la variable duale associée à la contrainte (55). Les paramètres optimaux sont calculés par $\beta^*_{ij}=\beta^{1*}_{ij}+\beta^{2*}_{ij}-\beta^{3*}_{ij}-\beta^{4*}_{ij}$, $(i\neq j)$, et $\beta^*_{ii}=\beta^{1*}_{ii}+\beta^{2*}_{ii}-\beta^{3*}_{ii}-\beta^{4*}_{ii}-\beta^{5*}_{ii}$, où β^{1*}_{ij}, β^{2*}_{ij}, β^{3*}_{ij}, β^{4*}_{ij} $(i,j\in P)$ sont les valeurs optimales des variables duales associées aux contraintes (56), (57), (58) et (59) respectivement, et β^{5*}_{ii} $(i\in I)$ sont les valeurs optimales des variables duales associées aux contraintes (60).

Preuve. La preuve est directement déduite de la preuve du Théorème 4.3.2. □

Illustrons cette extension de `IQCR` par l'exemple suivant :

Exemple 4.3.1 *Soit (MQP_e) une instance de (MQP) qui a 2 variables entières et 2 variables continues :*

$$(MQP_e)\begin{cases} Min \quad f(x) = x^T \left(\begin{array}{cc|cc} -7 & 3 & -15 & -4 \\ 3 & -14 & -7 & -13 \\ \hline -15 & -7 & 8 & 7 \\ -4 & -13 & 7 & 12 \end{array}\right) x + \begin{pmatrix} 15 \\ 10 \\ -7 \\ -4 \end{pmatrix} x \\ s.c. \quad 5x_1 + x_2 + 8x_3 + 4x_4 \leq 95 \\ \qquad 0 \leq x_i \leq 10 \qquad\qquad i \in \{1, \ldots, 4\} \\ \qquad x_1, x_2 \in \mathbb{N} \\ \qquad x_3, x_4 \in \mathbb{R} \end{cases}$$

Remarquons tout d'abord que la sous matrice $\begin{pmatrix} 8 & 7 \\ 7 & 12 \end{pmatrix}$ *est demi-définie positive comme dans notre hypothèse.*

Une solution optimale de (MQP_e) *est* $x = (8, 10, 2.03, 7.19)$ *dont la valeur est* -3434.27.

Comme il n'y a pas de contraintes d'égalités, il en résulte qu'il n'y pas de paramètre α. *Nous perturbons donc le Hessien de la manière suivante :*

$$\begin{pmatrix} -7+\beta_{11} & 3+\beta_{12} & -15+\beta_{13} & -4+\beta_{14} \\ 3+\beta_{12} & -14+\beta_{22} & -7+\beta_{23} & -13+\beta_{24} \\ -15+\beta_{13} & -7+\beta_{23} & 8 & 7 \\ -4+\beta_{14} & -13+\beta_{24} & 7 & 12 \end{pmatrix}$$

avec $\beta_{11} = 12.25$, $\beta_{12} = 0.29$, $\beta_{13} = 8.74$, $\beta_{14} = 0$, $\beta_{22} = 21.17$, $\beta_{23} = 1.84$, *et* $\beta_{24} = 5.52$.

La valeur optimale de la relaxation continue du problème reformulé associé est de -4002.43. *Le gap est donc de* 16.5%

4.3.5 Utilisation des contraintes d'inégalité pour perturber la fonction objectif

En considérant un problème quadratique ayant p contraintes d'inégalité, il est facile de remplacer chaque contrainte d'inégalité par une contrainte d'égalité en ajoutant p variables d'écart. Pour effectuer cette reformulation, nous distinguons deux cas :

1. Considérons le problème (QP) définit au début du chapitre, où toutes les variables sont entières et supposons que tous les coefficients de ses contraintes d'inégalité soient entiers. Pour perturber la fonction objectif, non seulement avec les contraintes d'égalité, mais aussi avec les contraintes d'inégalité, nous reformulons (QP) en un programmes quadratique

dont toutes les variables sont entières de la façon suivante :

$$(QP')\begin{cases} Min & f(x) & & \\ s.c. & (1)(3)(4)(5) & & \\ & Dx+\gamma = e & & (63) \\ & 0 \leq \gamma_s \leq e_s & s \in S & (64) \\ & \gamma \in \mathbb{N}^p & & (65) \end{cases}$$

Un tel problème est une instance de (QP) de dimension $n+p$ et peut être résolu par l'algorithme `IQCR`.

2. Considérons le problème le problème (QP) définit au début du chapitre, où toutes les variables sont entières, ou bien le problème mixte entiers (MQP) défini juste avant et supposons que certains des coefficients de leurs contraintes d'inégalité soient réels. Pour perturber la fonction objectif, non seulement avec les contraintes d'égalité, mais aussi avec les contraintes d'inégalité, nous le reformulons en un programmes quadratique en variables mixtes entières de la façon suivante :

$$(MQP')\begin{cases} Min & f(x) & & \\ s.c. & (1)(3)(4)(5) & & \\ & Dx+\gamma = e & & (66) \\ & 0 \leq \gamma_s \leq e_s & s \in S & (67) \\ & \gamma \in \mathbb{R}^p & & (68) \end{cases}$$

Un tel problème est une instance de (MQP) de dimension $n+p$ et peut être résolu par l'algorithme `IQCRs`.

4.3.6 Application à l'exemple de base

Illustrons maintenant les méthodes `IQCR` et `IQCRs` sur l'Exemple 3.1.1 que nous rappelons et qui est un problème ayant 4 variables, 1 contrainte d'égalité et 1 contrainte d'inégalité :

Exemple 3.1.1 (suite) :

$$(QP_e)\begin{cases} Min & f(x) = x^T \begin{pmatrix} 5 & -7 & -6 & -1 \\ -7 & 3 & -8 & -18 \\ -6 & -8 & -17 & 10 \\ -1 & -18 & 10 & 3 \end{pmatrix} x + \begin{pmatrix} -5 \\ -11 \\ 4 \\ 1 \end{pmatrix} x \\ s.c. & 3x_1 + 19x_2 + 18x_3 + 11x_4 = 255 \\ & 11x_1 + 13x_2 + 8x_3 + x_4 \leq 165 \\ & 0 \leq x_i \leq 10 & i \in \{1, \ldots, 4\} \\ & x_i \in \mathbb{N} & i \in \{1, \ldots, 4\} \end{cases}$$

La valeur de la solution optimale entière de (QP_e) *est* -2552.

Avec la méthode `IQCR`*, nous perturbons le Hessien de* $f(x)$ *de la manière suivante :*

$$\begin{pmatrix} 5+\beta_{11}+9\alpha & -7+\beta_{12}+57\alpha & -6+\beta_{13}+54\alpha & -1+\beta_{14}+33\alpha \\ -7+\beta_{12}+57\alpha & 3+\beta_{22}+361\alpha & -8+\beta_{23}+342\alpha & -18+\beta_{24}+209\alpha \\ -6+\beta_{13}+54\alpha & -8+\beta_{23}+342\alpha & -17+\beta_{33}+324\alpha & 10+\beta_{34}+198\alpha \\ -1+\beta_{14}+33\alpha & -18+\beta_{24}+209\alpha & 10+\beta_{34}+198\alpha & 3+\beta_{44}+121\alpha \end{pmatrix}$$

avec $\alpha = 2090.76$*, et* $\beta_{11} = \beta_{12} = \beta_{13} = \beta_{14} = \beta_{22} = \beta_{23} = \beta_{24} = 0$*,* $\beta_{33} = 24.45$*,* $\beta_{34} = -6.80$*,* $\beta_{44} = 4.92$.

La valeur optimale de la relaxation continue du problème reformulé associé est de -2808.77*. Le gap est donc de* 10.06%.

Appliquons maintenant sur cet exemple l'intégration de la contrainte d'inégalité au processus de convexification comme décrit dans le point 2 de la Section 4.3.5, c'est à dire lorsque la variable d'écart est réelle. Nous appliquons ensuite la méthode `IQCRs` décrite dans la Section 4.3.4 :

Exemple 3.1.1 (suite) : *D'abord, nous reformulons* (QP_e) *en un problème soumis à* 2

contraintes d'égalité en ajoutant la variable d'écart x_5 :

$$(QP'_e)\begin{cases} Min & f(x) = x^T \begin{pmatrix} 5 & -7 & -6 & -1 \\ -7 & 3 & -8 & -18 \\ -6 & -8 & -17 & 10 \\ -1 & -18 & 10 & 3 \end{pmatrix} x + \begin{pmatrix} -5 \\ -11 \\ 4 \\ 1 \end{pmatrix} x \\ s.c. & 3x_1 + 19x_2 + 18x_3 + 11x_4 = 255 \\ & 11x_1 + 13x_2 + 8x_3 + x_4 + x_5 = 165 \\ & 0 \leq x_i \leq 10 & i \in \{1, \ldots, 4\} \\ & 0 \leq x_5 \leq 165 \\ & x_i \in \mathbb{N} & i \in \{1, \ldots, 4\} \\ & x_5 \in \mathbb{R} \end{cases}$$

Avec la méthode `IQCRs`*, nous perturbons le Hessien de $f(x)$ de la manière suivante :*

$$\begin{pmatrix} 5+\beta_{11}+9\alpha+121\alpha' & -7+\beta_{12}+57\alpha+143\alpha' & -6+\beta_{13}+54\alpha+88\alpha' & -1+\beta_{14}+33\alpha+11\alpha' & \beta_{15}+11\alpha' \\ -7+\beta_{12}+57\alpha+143\alpha' & 3+\beta_{22}+361\alpha+169\alpha' & -8+\beta_{23}+342\alpha+104\alpha' & -18+\beta_{24}+209\alpha+13\alpha' & \beta_{25}+13\alpha' \\ -6+\beta_{13}+54\alpha+88\alpha' & -8+\beta_{23}+342\alpha+104\alpha' & -17+\beta_{33}+324\alpha+64\alpha' & 10+\beta_{34}+198\alpha+8\alpha' & \beta_{35}+8\alpha' \\ -1+\beta_{14}+33\alpha+11\alpha' & -18+\beta_{24}+209\alpha+13\alpha' & 10+\beta_{34}+198\alpha+8\alpha' & 3+\beta_{44}+121\alpha+\alpha' & \beta_{45}+\alpha' \\ \beta_{15}+11\alpha' & \beta_{25}+13\alpha' & \beta_{35}+8\alpha' & \beta_{45}+\alpha' & \alpha' \end{pmatrix}$$

avec $\alpha = 116.91$*,* $\alpha' = 249.74$*, et* $\beta_{11} = \beta_{12} = 0$*,* $\beta_{13} = 0.04$*,* $\beta_{14} = -0.04$*,* $\beta_{14} = -0.38$*,* $\beta_{23} = -0.09$*,* $\beta_{24} = 0.09$*,* $\beta_{25} = 0.44$*,* $\beta_{33} = 27.23$*,* $\beta_{34} = -4.72$*,* $\beta_{35} = -2.18$*,* $\beta_{44} = 2.60$*,* $\beta_{45} = 1.63$*.*

Rappelons que le paramètre α est le paramètre associé à la contrainte d'égalité initiale élevée au carré $(3x_1 + 19x_2 + 18x_3 + 11x_4 - 255)^2$ et que le paramètre α' est le paramètre associé à la nouvelle contrainte d'égalité élevée au carré $(11x_1 + 13x_2 + 8x_3 + x_4 + x_5 - 165)^2$.

La valeur optimale de la relaxation continue du problème reformulé associé est de -2776.07. Le gap est donc de 8.9%.

Nous résumons les performances des méthodes `IQCR` et `IQCRs` dans le tableau suivant :

solution opt. entière	**-2552**	
	`IQCR`	`IQCRs`
nombres de variables	100	121
nombres de contraintes	246	302
temps de résolution (s)	0.004	0.04
valeur de la relax. cont.	-2808.77	-2776.07
saut d'intégrité	10.06 %	8.9%

Discussion sur le paramètre α :

Grâce au paramètre β, la méthode `IQCR` perturbe par un paramètre particulier chacun des termes de la matrice Q, il est donc légitime de se demander si le paramètre α a une réelle utilité. Pour expliquer son intérêt, présentons d'abord la résolution de l'exemple 3.1.1 :

1. La résolution de l'exemple 3.1.1 par l'approche `IQCR` fournit un programme dont la valeur de la solution optimale continue est -2808.77 où le paramètre α vaut 2090.76, et le paramètre β est la matrice suivante : $\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 24.45 & -6.80 \\ 0 & 0 & -6.80 & 4.92 \end{pmatrix}$
2. La résolution de l'exemple 3.1.1 par l'approche `IQCR` qui n'intègre pas les contraintes d'égalité dans le processus de convexification, et qui ne possède donc pas de paramètre α fournit un programme dont la valeur de la solution optimale continue est -3463.12 où le paramètre β est la matrice $\begin{pmatrix} 5.68 & 1.02 & 0.41 & 0 \\ 1.02 & 20.18 & 0 & 1.67 \\ 0.41 & 0 & 26.17 & 0 \\ 0 & 1.67 & 0 & 11.47 \end{pmatrix}$

Sur cet exemple simple nous voyons que le paramètre α a une incidence sur la qualité de la relaxation continue de la reformulation $(QP_{\alpha,\beta})$. En fait, lorsque nous perturbons la fonction initiale $f(x)$, on lui ajoute deux expressions de natures bien différentes :

(i) La première $\alpha\sum_{r=1}^{m}(\sum_{i=1}^{n}a_{ri}x_i - b_r)^2$ est une expression qui sera nulle lors de la résolution entière et lors de la résolution de la relaxation continue.

(ii) La deuxième $\sum_{i=1}^{n}\sum_{j=1}^{n}\beta_{ij}(x_ix_j - y_{ij})$ est une expression qui , presque toujours, sera nulle uniquement lors de la résolution entière. Lors de la résolution de la relaxation continue nous aurons dans la plupart des cas $x_ix_j \neq y_{ij}$. Cette expression pénalise donc la valeur de la relaxation continue, lorsque les variables x_i n'atteignent pas leurs bornes.

Il est donc plus intéressant de minimiser les valeurs du paramètre matriciel β et de "compenser" en additionnant sur chaque terme de la matrice Q la valeur du paramètre scalaire α, qui peut être de grande valeur sans perturber la valeur de $f_{\alpha,\beta}(x)$. C'est là que réside l'intérêt du paramètre α.

`IQCR` fournit une borne obtenue par relaxation continue de très bonne qualité, mais le problème reformulé est de taille conséquente. C'est pour cette raison que nous allons présenter maintenant la méthode `CQCR (Compact Quadratic Convex Reformulation)` qui consiste à perturber uniquement la diagonale de Q avec des paramètres particuliers. Cette méthode fournit une borne de moins bonne qualité, mais un programme reformulé de plus petite taille.

4.4 Une restriction intéressante de `IQCR` : `CQCR (Compact Quadratic Convex Reformulation)`

Ici, nous considérons la méthode `IQCR` en imposant la nullité des paramètres β_{ij} lorsque $i \neq j$. Ainsi, nous définissons un schéma de réécriture de (QP) en un programme quadratique $(QP^{\mathtt{C}}_{\alpha,\beta})$ qui a une fonction objectif convexe, et qui est de plus petite taille que celui présenté dans la section précédente. L'idée est, au lieu d'utiliser les expressions linéaires de tous les produits, de n'utiliser que l'expression linéaire des carrés pour perturber la fonction objectif. Pour cela, nous utilisons un paramètre vectoriel β. De plus, nous utilisons toujours les contraintes d'égalité pour perturber la fonction objectif à l'aide d'un paramètre scalaire α. Concrètement, nous perturbons $f(x)$ avec les fonctions suivantes qui s'annulent sur le domaine de définition de (QP) et sous l'hypothèse que $v_i = x_i^2$:

(i) $\alpha \sum_{r=1}^{m} (\sum_{i=1}^{n} a_{ri} x_i - b_r)^2$ avec $\alpha \in \mathbb{R}$

(ii) $\sum_{i=1}^{n} \beta_i (x_i^2 - v_i)$ avec $\beta_i \in \mathbb{R}$.

Soit $(QP^{\mathtt{C}}_{\alpha,\beta})$ le programme suivant :

$$(QP^{\mathtt{C}}_{\alpha,\beta}) \begin{cases} Min & f^{\mathtt{C}}_{\alpha,\beta}(x,v) = f(x) + \alpha \sum_{r=1}^{m} (\sum_{i=1}^{n} a_{ri} x_i - b_r)^2 + \sum_{i=1}^{n} \beta_i (x_i^2 - v_i) \\ s.c. & (1)(2) \\ & x, v, z, t \in P^{\mathtt{C}}_{xvzt} \end{cases}$$

avec $P^{\mathtt{C}}_{xvzt}$ l'ensemble suivant :

$$P^{\mathtt{C}}_{xvzt}\left\{\begin{array}{lll} & (3)(4)(19) & \\ & x_i = \sum_{k=0}^{\lfloor log(u_i)\rfloor} 2^k t_{ik} & i \in I \quad (69) \\ & z_{ik} \leq u_i t_{ik} & (i,k) \in E \quad (70) \\ & z_{ik} \leq x_i & (i,k) \in E \quad (71) \\ x,v,z,t: & z_{ik} \geq x_i - u_i(1-t_{ik}) & (i,k) \in E \quad (72) \\ & z_{ik} \geq 0 & (i,k) \in E \quad (73) \\ & v_i = \sum_{k=0}^{\lfloor log(u_i)\rfloor} 2^k z_{ik} & i \in I \quad (74) \\ & v_i \geq 2x_i u_i - u_i^2 & i \in I \quad (75) \\ & v_i \geq x_i & i \in I \quad (76) \end{array}\right.$$

Montrons que (QP) est équivalent à $(QP^{\mathtt{C}}_{\alpha,\beta})$. D'abord, il est évident que si la contrainte (1) est satisfaite et que $v_i = x_i^2$, nous avons $f(x) = f^{\mathtt{C}}_{\alpha,\beta}(x,v)$. Il reste donc à montrer que l'ensemble $(P^{\mathtt{C}}_{xvzt})$ force les variables v_i à être égales à x_i^2. Dans l'ensemble $P^{\mathtt{C}}_{xvzt}$, l'ensemble d'inégalités (70)-(73) et (19) forcent l'égalité $z_{ik} = x_i t_{ik}$, et par la contrainte (74), nous avons bien $v_i = x_i^2$.

Comme dans l'approche `IQCR`, nous pouvons ajouter les inégalités valides (75) et (76).

Nous cherchons à calculer la meilleure convexification à l'intérieur de ce schéma, et nous nous intéressons donc à la relaxation continue de $(QP^{\mathtt{C}}_{\alpha,\beta})$ qui est la suivante :

$$(\overline{QP^{\mathtt{C}}}_{\alpha,\beta})\left\{\begin{array}{ll} Min & f^{\mathtt{C}}_{\alpha,\beta}(x,v) \\ s.c. & (1)(2) \\ & x,v,z,t \in \overline{P}^{\mathtt{C}}_{xvzt} \end{array}\right.$$

avec $\overline{P}^{\mathtt{C}}_{xvzt}$ le polyèdre obtenu en relâchant les contraintes d'intégrité de $P^{\mathtt{C}}_{xvzt}$, et donc en remplaçant les contraintes (19) par

$$0 \leq t_{ik} \leq 1 \quad (19)'$$

Etant donné que les variables z et t n'interviennent pas dans la fonction objectif $f^{\mathtt{C}}_{\alpha,\beta}(x,v)$, comme dans l'approche `IQCR`, il est intéressant de définir une projection de l'ensemble $\overline{P}^{\mathtt{C}}_{xvzt}$ sur x et v. Nous pouvons déduire le théorème suivant :

Théorème 4.4.1 *La projection de $\overline{P}^{\mathtt{C}}_{xvzt}$ sur x et v est le polyèdre suivant $P^{\mathtt{C}}_{xv}$:*

$$P^{\mathtt{C}}_{xv}\left\{\begin{array}{lll} & (75)(76) & \\ x,v: & v_i \geq 0 & i \in I \quad (77) \\ & v_i \leq u_i x_i & i \in I \quad (78) \end{array}\right.$$

Preuve.

1. Montrons que pour tout $(\bar{x}, \bar{v})$ appartenant à $P^{\mathtt{C}}_{xv}$, il existe z, t tels que $(\bar{x}, \bar{v}, z, t)$ appartiennent à $\overline{P}^{\mathtt{C}}_{xvzt}$.

   Soit $(\bar{x}, \bar{v}) \in P^{\mathtt{C}}_{xv}$, et $\gamma_i \geq 0$ tels que $\bar{v}_i = \gamma_i \bar{x}_i^2$. Prenons, $\forall (i,k) \in E$, $t_{ik} = \bar{x}_i / \bar{u}_i$, et $\forall (i,k) \in E$, $z_{ik} = \gamma_i t_{ik} \bar{x}_i$. De manière évidente les contraintes (3), (4), (19'), (69), (73), (74), (75) et (76) sont satisfaites.

   Montrons maintenant que dans une telle solution, les contraintes (70)-(72) sont satisfaites :

   - Les contraintes (70), $z_{ik} \leq u_i t_{ik}$:

     (i) Si $\bar{x}_i = 0$, $t_{ik} = 0$, alors $z_{ik} = 0$.

     (ii) Si $\bar{x}_i > 0$, par les contraintes (78), on a $\gamma_i \bar{x}_i^2 \leq u_i \bar{x}_i$ et donc $\gamma_i x_i \leq u_i$. Nous avons donc $\gamma_i \bar{x}_i t_{ik} = z_{ik} \leq u_i t_{ik}$.

   - Les contraintes (71), $z_{ik} \leq x_i$:

     Nous avons $z_{ik} = \gamma_i \bar{x}_i t_{ik} = \gamma_i \bar{x}_i^2 / \bar{u}_i = \bar{v}_i / \bar{u}_i$, par les contraintes (78), $\bar{v}_i / \bar{u}_i \leq \bar{x}_i u_i / \bar{u}_i \leq \bar{x}_i$, car $u_i / \bar{u}_i \leq 0$.

   - Les contraintes (72), $z_{ik} \geq x_i - u_i(1 - t_{ik})$:

     On a $z_{ik} = \gamma_i \bar{x}_i t_{ik} = \bar{v}_i / \bar{u}_i$, par les contraintes (75) $\bar{v}_i / \bar{u}_i \geq (2u_i \bar{x}_i - u_i^2)/\bar{u}_i \geq 2t_{ik} u_i + u_i / \bar{u}_i (\bar{x}_i - u_i) \geq t_{ik} u_i + \bar{x}_i - u_i$, car $u_i / \bar{u}_i \leq 0$.

2. Montrons que pour tout $(\bar{x}, \bar{v}, \bar{z}, \bar{t})$ appartenant $\overline{P}^{\mathtt{C}}_{xvzt}$, $(\bar{x}, \bar{v})$ appartient à $P^{\mathtt{C}}_{xv}$.

   Soit $(\bar{x}, \bar{v}, \bar{z}, \bar{t}) \in \overline{P}^{\mathtt{C}}_{xvzt}$. De manière évidente $(\bar{x}, \bar{v})$ satisfait les contraintes (75), (76) et (77).

   Montrons maintenant que dans une telle solution, les contraintes (78) sont satisfaites :

   - Les contraintes (78), $v_i \leq u_i x_i$:

     Par les contraintes (74), (70), on a $\bar{v}_i = \sum_{k=0}^{\lfloor log(u_i) \rfloor} 2^k \bar{z}_{ik} \leq \sum_{k=0}^{\lfloor log(u_i) \rfloor} 2^k u_i \bar{t}_{ik} = u_i \bar{x}_i$

□

Par le Théorème 4.4.1 et parce que la fonction objectif de $(\overline{QP}^{\mathtt{C}}_{\alpha,\beta})$ ne contient que les variables x et v, on en déduit que le polyèdre $\overline{P}^{\mathtt{C}}_{xvzt}$ dans $(\overline{QP}^{\mathtt{C}}_{\alpha,\beta})$ peut être remplacé par la polyèdre $P^{\mathtt{C}}_{xv}$. Plus formellement, soit $(P^{\mathtt{C}}_{\alpha,\beta})$ le programme suivant :

$$(P^{\mathtt{C}}_{\alpha,\beta}) \begin{cases} Min & f^{\mathtt{C}}_{\alpha,\beta}(x,v) \\ s.c. & (1)(2) \\ & x, v \in P^{\mathtt{C}}_{xv} \end{cases}$$

Corollaire 4.4.1 *La valeur optimale de $(\overline{QP}^{\mathtt{C}}_{\alpha,\beta})$ est égale à la valeur optimale de $(P^{\mathtt{C}}_{\alpha,\beta})$.*

Par le Théorème 4.3.2, nous déduisons le calcul, par programmation semi-définie, des valeurs de α^* et β^* qui rendent $f^{\mathtt{C}}_{\alpha^*,\beta^*}(x,v)$ convexes, tout en maximisant la valeur de la relaxation continue de $(QP^{\mathtt{C}}_{\alpha^*,\beta^*})$. Comme cela a été montré dans le Corollaire 4.4.1, cela revient à résoudre le problème suivant :

$$(CP^{\mathtt{C}}): \max_{\substack{\alpha\in\mathbb{R},\quad \beta\in\mathbb{R}^n \\ Q_{\alpha,\beta}\succeq 0}} \{v(P^{\mathtt{C}}_{\alpha,\beta})\}$$

avec $v(P^{\mathtt{C}}_{\alpha,\beta})$ la valeur de la solution optimale de $(P^{\mathtt{C}}_{\alpha,\beta})$ et $Q^{\mathtt{C}}_{\alpha,\beta}$ le hessien de $f^{\mathtt{C}}_{\alpha,\beta}(x,v)$, i.e. $Q^{\mathtt{C}}_{\alpha,\beta} = Q + \alpha A^T A + diag(\beta)$, avec $diag(\beta)$ la matrice dont les termes diagonaux sont les éléments du vecteur β. Nous pouvons réécrire $(CP^{\mathtt{C}})$ comme suit :

$$(CP^{\mathtt{C}}): \max_{\substack{\alpha\in\mathbb{R},\quad \beta\in\mathbb{R}^n \\ Q^{\mathtt{C}}_{\alpha,\beta}\succeq 0}} \min_{\substack{(1)(2) \\ (x,v)\in P^{\mathtt{C}}_{xv}}} \{f^{\mathtt{C}}_{\alpha,\beta}(x,v)\}$$

Du Théorème 4.3.2, nous déduisons le Corollaire suivant :

Corollaire 4.4.2 *Soit $(SDP^{\mathtt{C}})$ le programme suivant :*

$$(SDP^{\mathtt{C}}) \begin{cases} Min & f(X,x) = \sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij}X_{ij} + \sum_{i=1}^{n} c_i x_i & & \\ s.c. & (1)(2)(23)(29)(30) & & \\ & X_{ii} \le u_i x_i & i \in I & (79) \\ & -X_{ij} \le -2u_i x_i + u_i^2 & i \in I & (80) \\ & -X_{ii} \le 0 & i \in I & (81) \\ & -X_{ii} \le -x_i & i \in I & (82) \end{cases}$$

Une solution optimale (α^*, β^*) de $(CP^{\mathtt{C}})$ peut être déduite des valeurs optimales des variables duales de $(SDP^{\mathtt{C}})$. Le paramètre optimal α^* est la valeur optimale de la variable duale associée à la contrainte (23). Les paramètres optimaux sont calculés par $\beta_i^* = \beta_i^{1*} - \beta_i^{2*} - \beta_i^{3*} - \beta_i^{4*}$, où β_i^{1*}, β_i^{2*}, β_i^{3*}, β_i^{4*} $(i \in I)$ sont les valeurs optimales des variables duales associées aux contraintes (79), (80), (81) et (82) respectivement.

A partir du Corollaire 4.4.2, nous construisons un algorithme compact de résolution exacte de problèmes quadratiques non convexes et à variables entières basé sur le reformulation `CQCR` :

CQCR (Compact Quadratic Convex Reformulation)
1. Résoudre le programme semi-défini ($SDP^{\mathtt{C}}$)
2. Déduire α^* et β^* (Corollaire 4.4.2)
3. Résoudre le programme ($QP^{\mathtt{C}}_{\alpha^*,\beta^*}$) par un solver MIQP.

FIG. 4.2 – Algorithme de résolution de (QP) basé sur la reformulation CQCR

La Remarque 4.3.1 s'applique également à la reformulation convexe compacte. On en déduit la remarque suivante :

Remarque 4.4.1 *Pour toute solution réalisable de ($DSDP^{\mathtt{C}}$) dont la valeur de la fonction objectif vaut Δ, nous pouvons déduire une reformulation convexe dont la valeur de la borne obtenue par relaxation continue n'est pas plus petite que Δ.*

Application à l'exemple de base

Rappelons d'abord l'Exemple 3.1.1 qui est un problème ayant 4 variables, 1 contrainte d'égalité et 1 contrainte d'inégalité :

Exemple 3.1.1 (suite) :

$$(QP_e)\begin{cases} Min & f(x) = x^T \begin{pmatrix} 5 & -7 & -6 & -1 \\ -7 & 3 & -8 & -18 \\ -6 & -8 & -17 & 10 \\ -1 & -18 & 10 & 3 \end{pmatrix} x + \begin{pmatrix} -5 \\ -11 \\ 4 \\ 1 \end{pmatrix} x \\ s.c. & 3x_1 + 19x_2 + 18x_3 + 11x_4 = 255 \\ & 11x_1 + 13x_2 + 8x_3 + x_4 \leq 165 \\ & 0 \leq x_i \leq 10 \qquad i \in \{1,\ldots,4\} \\ & x_i \in \mathbb{N} \qquad i \in \{1,\ldots,4\} \end{cases}$$

La valeur de la solution optimale entière de (QP_e) est -2552.

Avec la méthode CQCR, nous perturbons le Hessien de $f(x)$ de la manière suivante :

$$\begin{pmatrix} 5+\beta_1+9\alpha & -7+57\alpha & -6+54\alpha & -1+33\alpha \\ -7+57\alpha & 3+\beta_2+361\alpha & -8+342\alpha & -18+209\alpha \\ -6+54\alpha & -8+342\alpha & -17+\beta_3+324\alpha & 10+198\alpha \\ -1+33\alpha & -18+209\alpha & 10+198\alpha & 3+\beta_4+121\alpha \end{pmatrix}$$

avec $\alpha = 1332.14$, et $\beta_1 = \beta_2 = 0$, $\beta_3 = 30.18$, $\beta_4 = 12.82$.

La valeur optimale de la relaxation continue du problème reformulé associé est de -2819.59. *Le gap est donc de* 10.46.

Nous résumons les performances des méthodes `NC`, `IQCR`, `IQCRs` et `CQCR` dans le tableau suivant :

solution opt. entière	**-2552**			
	`NC`	`IQCR`	`IQCRs`	`CQCR`
nombres de variables	52	100	121	40
nombres de contraintes	14	246	302	66
temps de résolution (s)	0.01	0.004	0.04	0.012
valeur de la relax. cont.	-3991.82	-2808.77	-2776.07	-2819.59
saut d'intégrité	56.42%	10.06 %	8.9%	10.46 %

Nous constatons sur cet exemple que la méthode `NC` est en termes de temps de résolution et de qualité de borne la moins performante des 4 approches. Pour les autres méthodes, c'est la méthode `IQCRs` qui a le meilleur gap, puis la méthode `IQCR` et enfin la méthode `CQCR`. Par contre, en termes de tailles de problèmes, il est clair que la méthode `CQCR` produit un programme de taille significativement plus petite que celui fourni par les méthodes `IQCR` et `IQCRs`.

Nous allons maintenant mettre en parallèle les convexifications pour la programmation quadratique en variables $0-1$ citées dans le Chapitre 2, avec les convexifications présentées dans ce chapitre. Cette comparaison est intéressante puisque la programmation en variables binaires est un cas particulier de la programmation en variables entières. Il est simple d'appliquer `NC`, `CQCR` et `IQCR` à la programmation quadratique en variables binaires en imposant aux bornes supérieures sur les variables x_i d'être égales à 1. Ainsi, nous montrons d'abord la correspondance entre la méthode `NC` appliquées aux programmes en variables binaires et la méthode de la plus petite valeur propre [19], puis celle entre la méthode `CQCR` appliquée aux programmes en variables binaires et la méthode `QCR` [4]. Enfin, nous appliquons `IQCR` aux problèmes en variables binaires.

4.5 Interprétation des méthodes `NC`, `CQCR` et `IQCR` pour la programmation quadratique en variables 0-1

Dans cette section, nous interprétons chacune des méthodes de convexification présentées dans le début de ce chapitre, à savoir `NC`, `CQCR`, et `IQCR` pour les problèmes particuliers où les

variables entières sont bornées par 1. Nous nous intéressons donc aux problèmes définis dans le Chapitre 2. Nous rappelons la formulation d'un tel problème :

$$(QP^{01}) \begin{cases} Min & f(x) = \sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij}x_ix_j + \sum_{i=1}^{n} c_ix_i \\ s.c. & (1)(2) \\ & x_i \in \{0,1\} \quad i \in I \qquad (83) \end{cases}$$

Nous montrons que nous pouvons faire un parallèle intéressant entre les méthodes de convexifications pour la programmation quadratique en variables $0-1$, présentées dans le Chapitre 2, et les trois méthodes de convexification que nous venons de présenter.

4.5.1 La méthode NC correspond à la méthode de la plus petite valeur propre

Dans le Chapitre 2, nous avons présenté la méthode la méthode de la plus petite valeur propre [19]. Cette méthode consiste à perturber la diagonale de Q, le hessien de $f(x)$, à l'aide de la plus petite valeur propre de Q, $\lambda_{min}(Q)$, et de la propriété $x_i^2 = x_i$. La méthode NC, reprend simplement cette idée, en exprimant de façon simple le carré de chaque entier.

En remplaçant, dans le programme $(QP_{\lambda_{min}})$, les bornes supérieures u_i sur les variables x_i, par 1, pour tout $i \in I$, nous obtenons exactement le programme $(QP_{\texttt{EV}})$ (cf. Chapitre 2) de la reformulation de Hammer et Rubin [19].

4.5.2 La méthode CQCR correspond à la méthode QCR

Dans le Chapitre 2, nous avons présenté QCR. Nous rappelons que cette approche consiste à reformuler (QP^{01}) en un problème équivalent $(QP_{\texttt{QCR}})$. L'innovation et la qualité de cette méthode proviennent à la fois de la perturbation de chaque élément de la diagonale de Q par un paramètre particulier mais aussi de l'intégration des contraintes d'égalités (1) dans le processus de convexification. La combinaison de ces deux idées améliore considérablement la borne obtenue par relaxation continue du problème reformulé, en comparaison avec la méthode de la plus petite valeur propre, et fournit un problème reformulé de taille raisonnable.

Rappelons plus formellement l'algorithme QCR. Billionnet, Elloumi et Plateau définissent un schéma de reformulation dépendant de deux paramètres δ et η, qui est le suivant :

$$(QP_{\texttt{QCR}}) \begin{cases} Min & f_{\texttt{QCR}}(x) = f(x) + \sum_{i=1}^{n}\delta_i(x_i^2 - x_i) + \sum_{r=1}^{m}(\sum_{i=1}^{n}\eta_{ri}x_i)(\sum_{j=1}^{n}a_{ri}x_i - b_r) \\ s.c. & (1)(83) \end{cases}$$

Puis, au sein de ce schéma, ils calculent la convexification optimale du point de vue de la borne

obtenue par relaxation continue. Ils cherchent donc les valeurs optimales des paramètres δ et η solutions du problème suivant :

$$(CP_{\texttt{QCR}}) : \{ \quad \max_{\eta \in \mathbb{R}^{mn}, \delta \in \mathbb{R}^n, Q_{\texttt{QCR}} \succeq 0} \{ v(\overline{QP}_{\texttt{QCR}}) \}$$

avec $v(\overline{QP}_{\texttt{QCR}})$ la valeur optimale de la relaxation continue de $(QP_{\texttt{QCR}})$. Ils prouvent le Théorème 2.2.2 suivant :

Théorème 2.2.2 ***(Billionnet, Elloumi, Plateau, 2009)*** *Soit $(SDP_{\texttt{QCR}})$ le programme semi-défini suivant :*

$$(SDP_{\texttt{QCR}}) \begin{cases} Min \quad f(X,x) = \sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij} X_{ij} + \sum_{i=1}^{n} c_i x_i & \\ s.c. \quad (1) & \\ \qquad -b_r x_i + \sum_{j=1}^{n} a_{rj} X_{ij} \quad i \in I, r \in R & (84) \\ \qquad X_{ii} = x_i \qquad\qquad i \in I & (85) \\ \qquad \begin{pmatrix} 1 & x \\ x^T & X \end{pmatrix} \succeq 0 & (86) \\ \qquad x \in \mathbb{R}^n \quad X \in \mathbf{S}^n & (87) \end{cases}$$

Une solution optimale (η^*, δ^*) de $(CP_{\texttt{QCR}})$ peut être déduite de la valeur optimale des variables duales de $(SDP_{\texttt{QCR}})$. Les valeurs optimales de η^* sont les valeurs optimales des variables duales associées aux contraintes (84) et les valeurs optimales de δ^* sont les valeurs optimales des variables duales associées aux contraintes (94).

Remarquons d'abord que la méthode `QCR` a été conçue initialement pour les programmes quadratiques en variables $0-1$ soumis seulement à des contraintes d'égalité. Nous mettons donc en parallèle `QCR` et `CQCR` pour les programmes quadratiques en variables $0-1$ soumis à des contraintes d'égalité. En remplaçant, dans le programme $(QP^{\texttt{C}}_{\alpha,\beta})$, les bornes supérieures sur les variables u_i, par 1 nous obtenons le programme suivant :

$$(QP^{\mathtt{C}}_{\alpha,\beta})\begin{cases} Min & f^{\mathtt{C}}_{\alpha,\beta}(x,v) = f(x) + \alpha\sum_{r=1}^{m}(\sum_{i=1}^{n}a_{ri}x_i - b_r)^2 + \sum_{i=1}^{n}\beta_i(x_i^2 - v_i) \\ s.c. & (1)(83) \\ & v_i \geq 2x_i - 1 \quad i \in I \\ & v_i \geq x_i \quad\quad i \in I \\ & v_i \geq 0 \quad\quad i \in I \\ & v_i \leq x_i \quad\quad i \in I \end{cases}$$

Ou de manière équivalente :

$$(QP^{\mathtt{C}}_{\alpha,\beta})\begin{cases} Min & f^{\mathtt{C}}_{\alpha,\beta}(x) = f(x) + \alpha\sum_{r=1}^{m}(\sum_{i=1}^{n}a_{ri}x_i - b_r)^2 + \sum_{i=1}^{n}\beta_i(x_i^2 - x_i) \\ s.c. & (1)(83) \end{cases}$$

Il faut remarquer que $(QP^{\mathtt{C}}_{\alpha,\beta})$ et $(QP_{\mathtt{QCR}})$ ne sont pas à première vue les mêmes problèmes, puisque leurs fonctions objectifs $f^{\mathtt{C}}_{\alpha,\beta}(x)$ et $f_{\mathtt{QCR}}(x)$ ne sont égales que si $\alpha\sum_{r=1}^{m}(\sum_{i=1}^{n}a_{ri}x_i - b_r)^2 = \sum_{r=1}^{m}(\sum_{i=1}^{n}\eta_{ri}x_i)(\sum_{j=1}^{n}a_{ri}x_i - b_r)$. Il a été prouvé dans [36] que les problèmes $(SDP_{\mathtt{QCR}})$ et $(SDP^{\mathtt{C}})$ ont la même valeur optimale. Il est donc équivalent de perturber la fonction objectif par l'une ou l'autre de ces deux expressions. On peut le justifier informellement de la façon suivante : pour $(\bar{\delta},\bar{\eta})$ calculés par l'algorithme `QCR`, en prenant $\bar{\alpha} = \max_{\bar{\eta}_i}\{\sum_{r=1}^{m}\bar{\eta}_{ri}a^{01}_{ri}\}$ et $\bar{\beta} = \bar{\delta}$, on obtient forcément une fonction $f^{\mathtt{C}}_{\bar{\alpha},\bar{\beta}}(x)$ convexe, de même valeur que $f_{\mathtt{QCR}}(x)$. Nous pouvons donc établir une relation claire entre `CQCR` et `QCR`.

4.5.3 La méthode `IQCR` correspond à un renforcement de `QCR`

Dans cette section, nous appliquons `IQCR` à la programmation quadratique en variables binaires. Nous remplaçons donc dans $(P_{\alpha,\beta})$, les bornes supérieures sur les variables par 1, et donc $\forall i \in I$, $u_i = 1$, nous appelons ce nouveau programme $(P^{01}_{\alpha,\beta})$:

$$(P^{01}_{\alpha,\beta})\begin{cases} Min & f_{\alpha,\beta}(x,y) = f(x) + \alpha\sum_{r=1}^{m}(\sum_{i=1}^{n}a_{ri}x_i - b_r)^2 + \sum_{i=1}^{n}\sum_{j=1}^{n}\beta_{ij}(x_ix_j - y_{ij}) \\ s.c & (1)(2) \\ & x, y \in P^{01}_{xy} \end{cases}$$

Avec P^{01}_{xy} l'ensemble suivant :

$$P_{xy}^{01}\begin{cases} x,y: & (83) & & \\ & y_{ii} = x_i & i \in I & (88) \\ & y_{ij} \leq x_i & (i,j) \in I^2 & (89) \\ & y_{ij} \leq x_j & (i,j) \in I^2 & (90) \\ & y_{ij} \geq x_i + x_j - 1 & (i,j) \in I^2 & (91) \\ & y_{ij} \geq 0 & (i,j) \in I^2 & (92) \\ & y_{ij} = y_{ji} & (i,j) \in I^2 & (93) \end{cases}$$

Nous nous intéressons à la relaxation continue de $(P_{\alpha,\beta}^{01})$, appelée $(\overline{P}_{\alpha,\beta}^{01})$ qui est la suivante :

$$(\overline{P}_{\alpha,\beta}^{01})\begin{cases} Min & f_{\alpha,\beta}(x,y) = f(x) + \alpha\sum_{r=1}^{m}(\sum_{i=1}^{n}a_{ri}x_i - b_r)^2 + \sum_{i=1}^{n}\sum_{j=1}^{n}\beta_{ij}(x_ix_j - y_{ij}) \\ s.c. & (1)(2) \\ & x,y \in \overline{P}_{xy}^{01} \end{cases}$$

Avec $\overline{P}_{xy}^{01}$ le polyèdre obtenu en relâchant les contraintes d'intégrité de P_{xy}^{01}, et donc en remplaçant les contraintes (83) par

$$0 \leq x_i \leq 1$$

Nous pouvons caractériser les valeurs des paramètre α et β qui maximisent la valeur de $(\overline{P}_{\alpha,\beta}^{01})$, en résolvant le problème suivant :

$$(CP^{01}): \max_{\substack{\alpha\in\mathbb{R},\quad \beta\in\mathbf{S}_n \\ Q_{\alpha,\beta}\succeq 0}} \{v(\overline{P}_{\alpha,\beta}^{01})\}$$

où $v(\overline{P}_{\alpha,\beta}^{01})$ est la valeur de la solution optimale de $(\overline{P}_{\alpha,\beta}^{01})$ et $Q_{\alpha,\beta}$ le hessien de $f_{\alpha,\beta}(x,y)$, i.e. $Q_{\alpha,\beta} = Q + \alpha A^TA + \beta$. Nous pouvons réécrire (CP^{01}) comme suit :

$$(CP^{01}): \max_{\substack{\alpha\in\mathbb{R},\quad \beta\in\mathbf{S}_n \\ Q_{\alpha,\beta}\succeq 0}} \min_{\substack{(1)(2) \\ (x,y)\in\overline{P}_{x,y}^{01}}} \{f_{\alpha,\beta}(x,y)\}$$

Du Théorème 4.3.2, nous pouvons déduire le corollaire suivant :

Corollaire 4.5.1 *Soit (SDP^{01}) le programme semi-défini suivant :*

$$
(SDP^{01})\begin{cases}
Min \quad f(X,x) = \sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij}X_{ij} + \sum_{i=1}^{n} c_i x_i & & \\
s.c. \quad (1)(2) & & \\
\quad \sum_{r=1}^{m}(\sum_{i=1}^{n}(\sum_{j=1}^{n} a_{ri}a_{rj}X_{ij} - 2a_{ri}b_r x_i)) = -\sum_{r=1}^{m} b_r^2 & & (23)\\
\quad X_{ii} = x_i & i \in I & (94)\\
\quad X_{ij} \leq x_i & (i,j) \in I^2,\ i \neq j & (95)\\
\quad X_{ij} \leq x_j & (i,j) \in I^2,\ i \neq j & (96)\\
\quad -X_{ij} \leq -x_i - x_j + 1 & (i,j) \in I^2,\ i \neq j & (97)\\
\quad -X_{ij} \leq 0 & (i,j) \in I^2,\ i \neq j & (98)\\
\quad \begin{pmatrix} 1 & x \\ x^T & X \end{pmatrix} \succeq 0 & & (99)\\
\quad x \in \mathbb{R}^n \quad X \in \mathbf{S}^n & & (100)
\end{cases}
$$

Une solution optimale (α^*, β^*) *de* (CP^{01}) *peut être déduite des valeurs optimales des variables duales de* (SDP^{01}). *Le paramètre optimal* α^* *est la valeur optimale de la variable duale associée à la contrainte (23). Les paramètres optimaux* β_{ii}^{1*}, $(i \in I)$ *sont les valeurs optimales des variables duales associées aux contraintes (94). Les paramètres optimaux* β_{ij}^* $((i,j) \in I^2, i \neq j)$ *sont calculés par* $\beta_{ij}^* = \beta_{ij}^{2*} - \beta_{ij}^{3*} - \beta_{ij}^{4*} - \beta_{ij}^{5*}$, *où* β_{ij}^{2*}, β_{ij}^{3*}, β_{ij}^{4*}, β_{ij}^{5*} *sont les valeurs optimales des variables duales associées aux contraintes (95), (96), (97) et (98) respectivement.*

A partir du Corollaire 4.5.1, nous construisons un algorithme de résolution exacte de problèmes quadratiques non convexes et à variables binaires basé sur la reformulation `IQCR` qui est présenté dans la figure 4.3.

`IQCR` pour la programmation quadratique binaire
1. Résoudre le programme semi-défini (SDP^{01})
2. Déduire α^* et β^* du Corollaire 4.5.1
3. Résoudre le programme $(QP^{01}_{\alpha^*,\beta^*})$ par un solveur MIQP.

FIG. 4.3 – Algorithme de résolution de (QP^{01}) basé sur la reformulation `IQCR`

La Remarque 4.3.1 s'applique également pour la programmation quadratique binaire. On en déduit la remarque suivante :

Remarque 4.5.1 *Pour toute solution réalisable de* $(DSDP^{01})$ *dont la valeur de la fonction objectif vaut* Δ*, nous pouvons déduire une reformulation convexe dont la valeur de la borne obtenue par relaxation continue n'est pas plus petite que* Δ*.*

Nous allons maintenant comparer théoriquement la valeur de la borne obtenue par relaxation continue de `QCR` avec celle de `IQCR`.

Théorème 4.5.1 *La solution optimale de* (SDP^{01}) *est toujours plus grande que la solution optimale de* $(SDP_{\texttt{QCR}})$*.*

Preuve. Montrons qu'à partir de toute solution réalisable $(\bar{X}, \bar{x})$ de (SDP^{01}) nous pouvons construire une solution réalisable de $(SDP_{\texttt{QCR}})$ de même valeur par l'objectif.
Prenons $(X = \bar{X}, x = \bar{x})$, de manière évidente $(\bar{X}, \bar{x})$ satisfait les contraintes (1), (2), (94), (99) et (100). Montrons maintenant que les contraintes (84) sont satisfaites. Nous savons par la contrainte (23) que :

$$\sum_{r=1}^{m}(\sum_{i=1}^{n}(\sum_{j=1}^{n}a_{ri}a_{rj}\bar{X}_{ij} - 2a_{ri}b_r\bar{x}_i)) = -\sum_{r=1}^{m}(b_r)^2$$
$$\sum_{r=1}^{m}(\sum_{i=1}^{n}a_{ri}(\sum_{j=1}^{n}a_{rj}\bar{X}_{ij} - b_r\bar{x}_i)) - 2\sum_{r=1}^{m}(\sum_{i=1}^{n}a_{ri}b_r\bar{x}_i) = -\sum_{r=1}^{m}(b_r)^2$$

et que

$$\sum_{i=1}^{n}a_{ri}\bar{x}_i = b_r \qquad \forall r \in R$$

ou de manière équivalente

$$\sum_{i=1}^{n}a_{ri}\bar{x}_i b_r = (b_r)^2 \qquad \forall r \in R$$
$$\sum_{r=1}^{m}(\sum_{i=1}^{n}a_{ri}\bar{x}_i b_r) = \sum_{r=1}^{m}(b_r)^2$$

Donc nous avons

$$\sum_{r=1}^{m}(\sum_{i=1}^{n}a_{ri}(\sum_{j=1}^{n}a_{rj}\bar{X}_{ij} - b_r\bar{x}_i)) - \sum_{r=1}^{m}(\sum_{i=1}^{n}a_{ri}b_r\bar{x}_i) = 0$$
$$\sum_{r=1}^{m}(\sum_{i=1}^{n}a_{ri}(\sum_{j=1}^{n}a_{rj}\bar{X}_{ij} - b_r\bar{x}_i)) = 0$$

et comme $a_{ri} \geq 0$, on a donc

$$\sum_{j=1}^{n}a_{rj}\bar{X}_{ij} - b_r\bar{x}_i = 0 \qquad \forall r \in R$$

et les contraintes (84) sont satisfaites. Donc, pour toute solution réalisable de (SDP^{01}), nous pouvons construire une solution réalisable de ($SDP_{\texttt{QCR}}$), de même valeur par l'objectif. On en déduit donc que :

$$v(SDP^{01}) \geq v(SDP_{\texttt{QCR}})$$

□

4.6 Conclusion

Dans ce chapitre, nous avons introduit trois méthodes de résolutions pour les problèmes quadratiques généraux en variables entières (QP). Nous avons d'abord proposé la reformulation quadratique convexe `NC` qui à consiste exprimer linéairement les carrés des variables, en utilisant leurs décompositions unaires, puis à convexifier en utilisant la plus petite valeur propre du Hessien de l'objectif de (QP). Cette approche, très facile à mettre en oeuvre, s'est avérée peu efficace expérimentalement car elle fournit une borne obtenue par relaxation continue trop faible.

Nous avons ensuite proposé une deuxième approche de reformulation quadratique convexe, que nous avons appelée `IQCR`. Elle consiste en la définition d'un schéma général de reformulation quadratique convexe. Dans ce schéma nous ajoutons à l'objectif des fonctions qui s'annulent sur le domaine de notre problème reformulé. Ces fonctions sont construites grâce à l'expression linéaire des produits de variables entières, ainsi qu'à l'aide des contraintes d'égalité. Ensuite, nous cherchons à l'intérieur de ce schéma la meilleure reformulation convexe du point de vue de la valeur de la borne obtenue par relaxation continue. Nous avons démontré que cette meilleure reformulation, `IQCR`, peut être déduite de la solution optimale duale d'une relaxation semi-définie de (QP). Nos expérimentations montrent que cette approche est très efficace en pratique.

Puis, nous avons montré comment adapter `IQCR` à une large classe de problèmes de la programmation mixte entière, dans la méthode appelée `IQCRs`. Cette adaptation permet également d'utiliser les contraintes d'inégalité, en plus des contraintes d'égalité, pour perturber la fonction objectif. En effet, il est toujours possible de transformer les contraintes d'inégalité en contraintes d'égalité à l'aide de variables d'écarts réelles. Cette transformation fournit un programme en variables mixtes entières qu'il est possible de résoudre avec `IQCRs`. Expérimentalement, cette approche s'est avérée très prometteuse. Ces résultats montrent l'impact et la pertinence d'intégrer toutes les contraintes dans la perturbation de la fonction objectif.

Ensuite, nous avons présenté la méthode `CQCR` qui est une simplification de la méthode `IQCR`. Cette méthode fournit une reformulation quadratique convexe de (QP) choisie dans un sous-ensemble des reformulations possibles dans le schéma associé à `IQCR`. En effet, au lieu de perturber la fonction objectif avec les expressions linéaires de tous les produits de variables, elle

ne la perturbe qu'avec les expressions linéaires des carrés des variables. Cette reformulation étant plus compacte, elle induit sur certaines instances un temps de résolution réduit. Cependant, par construction, elle fournit une borne par relaxation continue qui est inférieure ou égale à celle fournie par la méthode `IQCR`. De plus, elle ne peut ni être adaptable à la programmation en variables mixtes entières, ni intégrer les contraintes d'inégalité dans la perturbation de la fonction objectif. L'intérêt de cette approche est donc expérimental.

Finalement, nous avons appliqué les trois méthodes `NC`, `CQCR` et `IQCR` à la programmation quadratique binaire. Nous avons montré que les méthodes `NC` et `CQCR` sont équivalentes respectivement à la méthode de la plus petite valeur propre et à la méthode `QCR`. Un résultat intéressant est que `IQCR` constitue une amélioration des méthodes existantes pour la programmation quadratique binaire. Nos expérimentations confirment notre étude théorique.

De nombreuses pistes de recherches émergent pour la reformulation convexe `IQCR`. Tout d'abord, comme nous avons vu dans ce chapitre, il n'est pas forcément nécessaire de calculer le paramètre scalaire α dans la reformulation `IQCR`, mais il est possible de le fixer à une certaine valeur. Un nouveau travail de recherche consiste à déterminer à quelle valeur doit être fixé ce paramètre α, ainsi que son influence sur le temps de résolution. En effet, peut être qu'un choix approprié de la valeur de ce paramètre fournirait une forme de fonction plus adaptée aux algorithmes de résolution de la programmation quadratique convexe. Il est même possible qu'il soit intéressant d'utiliser cette propriété pour reformuler des problèmes déjà convexes.

Une autre piste de recherche est une extension plus fine et intelligente de `IQCR` à la programmation mixte entière, car la solution proposée pour le moment est assez directe.

Finalement, il s'avère que `IQCR` est directement adaptable aux programmes quadratiques soumis à des contraintes quadratiques. Effectivement, `IQCR` combinant des idées de linéarisation et de convexification, il est possible de linéariser ces contraintes quadratiques. De plus, comme nous l'avons fait pour les contraintes linéaires, une nouvelle piste de recherche est d'intégrer ces contraintes quadratiques au processus de convexification.

Une grande partie de la preuve de l'optimalité de `IQCR` est basée sur la projection du polyèdre associé au schéma général de reformulations convexes que nous avons introduit. Cette projection, que nous avons appelé P_{xy} est de plus petite taille que le polyèdre de départ P_{xyzt}. Une amélioration intéressante de `IQCR` est la conception d'un Branch and Bound spécifique basé sur le polyèdre P_{xy} pour résoudre `IQCR`. Nous développons cette idée dans le chapitre suivant.

Chapitre 5

Un algorithme de Branch and Bound spécifique pour IQCR fondé sur les propriétés de projection de la Section 4.3.2

L'idée de ce chapitre est d'utiliser la propriété de projection du Théorème 4.3.1 au sein d'un algorithme de Branch and Bound. Le principe d'un tel algorithme est la parcours d'un arbre des solutions possibles du problème à résoudre. Cet arbre est construit pendant l'exécution de l'algorithme, où à chaque noeud les solutions d'un nouveau problème résolu déterminent les règles de branchements à respecter.

Rappelons tout d'abord que nous cherchons à résoudre à l'optimalité le problème suivant :

$$(QP)\begin{cases} Min & f(x) = \sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij}x_ix_j + \sum_{i=1}^{n} c_ix_i & & \\ s.c. & \sum_{i=1}^{n} a_{ri}x_i = b_r & r \in R & (1) \\ & \sum_{i=1}^{n} d_{si}x_i \leq e_s & s \in S & (2) \\ & x_i \leq u_i & i \in I & (3) \\ & x_i \geq 0 & i \in I & (4) \\ & x_i \in \mathbb{N} & i \in I & (5) \end{cases}$$

où $I = \{i : i = 1, \ldots, n\}$, $R = \{r : r = 1, \ldots, m\}$, et $S = \{s : s = 1, \ldots, p\}$.

Nous avons défini dans le Chapitre 4 un schéma de reformulations convexes qui est le suivant :

$$(QP_{\alpha,\beta})\begin{cases} Min & f_{\alpha,\beta}(x,y) = f(x) + \alpha\sum_{r=1}^{m}(\sum_{i=1}^{n}a_{ri}x_i - b_r)^2 + \sum_{i=1}^{n}\sum_{j=1}^{n}\beta_{ij}(x_ix_j - y_{ij}) \\ s.c. & (1)(2) \\ & x,y,z,t \in P_{xyzt} \end{cases}$$

où P_{xyzt} est l'ensemble suivant :

$$P_{xyzt}\begin{cases} x,y,z,t: & (3)(4) & & \\ & x_i = \sum_{k=0}^{\lfloor log(u_i)\rfloor} 2^k t_{ik} & i \in I & (6) \\ & y_{ij} = \sum_{k=0}^{\lfloor log(u_i)\rfloor} 2^k z_{ijk} & i,j \in I & (7) \\ & z_{ijk} \leq u_j t_{ik} & (i,k) \in E,\ j \in I & (8) \\ & z_{ijk} \leq x_j & (i,k) \in E,\ j \in I & (9) \\ & z_{ijk} \geq x_j - u_j(1-t_{ik}) & (i,k) \in E,\ j \in I & (10) \\ & z_{ijk} \geq 0 & (i,k) \in E,\ j \in I & (11) \\ & y_{ij} = y_{ji} & i,j \in I & (12) \\ & y_{ij} \geq x_iu_j + x_ju_i - u_iu_j & (i,k) \in E,\ j \in I & (13) \\ & y_{ii} \geq x_i & i \in I & (14) \\ & t_{ik} \in \{0,1\} & (i,k) \in E & (15) \end{cases}$$

Nous nous sommes ensuite intéressés à la relaxation continue de $(QP_{\alpha,\beta})$ que nous appelons $(\overline{QP}_{\alpha,\beta})$:

$$(\overline{QP}_{\alpha,\beta})\begin{cases} Min & f_{\alpha,\beta}(x,y) = f(x) + \alpha\sum_{r=1}^{m}(\sum_{i=1}^{n}a_{ri}x_i - b_r)^2 + \sum_{i=1}^{n}\sum_{j=1}^{n}\beta_{ij}(x_ix_j - y_{ij}) \\ s.c. & (1)(2) \\ & x,y,z,t \in \overline{P}_{xyzt} \end{cases}$$

avec $\overline{P}_{xyzt}$ le polyèdre obtenu en relâchant les contraintes d'intégrités de P_{xyzt}, et donc en remplaçant les contraintes (15) par :

$$0 \leq t_{ik} \leq 1 \quad (15')$$

Puis nous avons montré dans le Théorème 4.3.1 que la projection du polyèdre $\overline{P}_{xyzt}$ sur les

variables x et y est le polyèdre de plus petite taille P_{xy} :

$$P_{xy}\begin{cases} x,y: & (12)(13)(14) & & \\ & y_{ij} \geq 0 & i,j \in I,\ q_{ij} \neq 0,\ i \neq j & (16) \\ & y_{ij} \leq u_j x_i & i,j \in I,\ q_{ij} \neq 0,\ i \neq j & (17) \\ & y_{ij} \leq u_i x_j & i,j \in I,\ q_{ij} \neq 0,\ i \neq j & (18) \end{cases}$$

Nous en avons déduit que $(\overline{QP}_{\alpha,\beta})$ est équivalent à $(P_{\alpha,\beta})$ qui est le programme suivant :

$$(P_{\alpha,\beta})\begin{cases} Min & f_{\alpha,\beta}(x,y) = f(x) + \alpha\sum_{r=1}^{m}(\sum_{i=1}^{n} a_{ri}x_i - b_r)^2 + \sum_{i=1}^{n}\sum_{j=1}^{n}\beta_{ij}(x_i x_j - y_{ij}) \\ s.c. & (1)(2) \\ & x,y \in P_{xy} \end{cases}$$

Le problème $(P_{\alpha,\beta})$ étant de plus petite taille que le problème $(\overline{QP}_{\alpha,\beta})$, nous cherchons dans ce chapitre à utiliser le problème $(P_{\alpha,\beta})$ pour résoudre (QP) à l'optimalité. Pour cela, nous proposons un algorithme de Branch and Bound basé sur la propriété suivante :

Proposition 5.0.1 *Soit $(P^e_{\alpha,\beta})$ le problème suivant :*

$$(P^e_{\alpha,\beta})\begin{cases} Min & f_{\alpha,\beta}(x,y) = f(x) + \alpha\sum_{r=1}^{m}(\sum_{i=1}^{n} a_{ri}x_i - b_r)^2 + \sum_{i=1}^{n}\sum_{j=1}^{n}\beta_{ij}(x_i x_j - y_{ij}) \\ s.c. & (1)(2) \\ & x,y \in P^e_{xy} \end{cases}$$

avec P^e_{xy} l'ensemble suivant :

$$P^e_{xy}\begin{cases} x,y: & (12)(13)(14)(16)(17)(18) & \\ & x_i \in \mathbb{N} & i \in I \quad (19) \end{cases}$$

Une solution optimale de (QP), i.e. de $(QP_{\alpha,\beta})$, peut être obtenue en résolvant le problème $(P^e_{\alpha,\beta})$ augmenté des contraintes $x_i x_j = y_{ij} \quad \forall i, \forall j$.

Ainsi, une manière de trouver une solution optimale de (QP) est de résoudre à chaque noeud de l'arbre de résolution le problème $(P^e_{\alpha,\beta})$, puis de brancher de la manière suivante :
Soit $(\bar{x},\bar{y})$ la solution du noeud courant, plusieurs cas de figure se présentent :

1. Si $(\bar{x},\bar{y})$ respecte la propriété $\forall (i,j) \in I^2$, $\bar{x}_i\bar{x}_j = \bar{y}_{ij}$, alors $\bar{x}$ est une solution réalisable de (QP) et $f(\bar{x}) = f_{\alpha,\beta}(\bar{x},\bar{y})$.
2. Si $\exists (i,j) \in I^2$, tels que $\bar{x}_i\bar{x}_j \neq \bar{y}_{ij}$, nous branchons de la façon suivante :

(a) La branche où $x_i \neq \bar{x}_i$. Plus précisément, cette branche donne naissance à deux sous arbres, dans le premier on ajoute à la formulation de $(P^e_{\alpha,\beta})$ la contrainte $x_i \leq \bar{x}_i - 1$ et dans le deuxième on ajoute la contrainte $x_i \geq \bar{x}_i + 1$

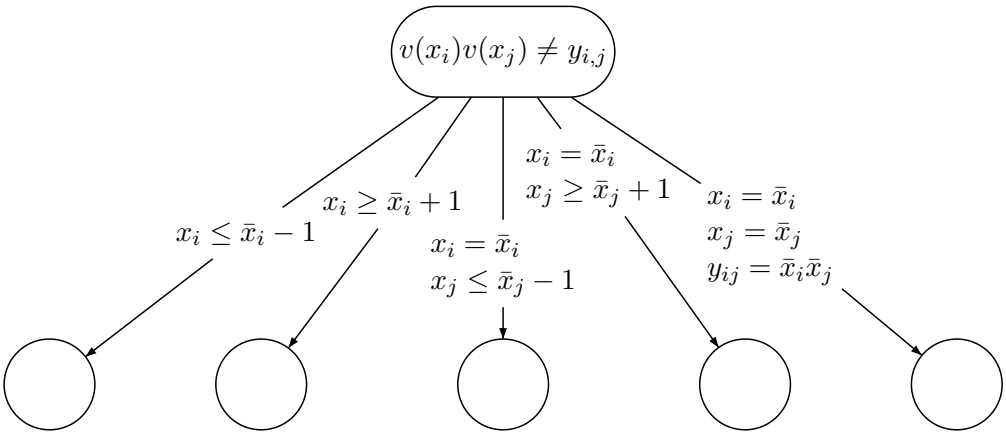
(b) La branche où $x_i = \bar{x}_i$ et $x_j \neq \bar{x}_j$. Plus précisément, cette branche donne naissance à deux sous arbres, dans le premier on ajoute à la formulation de $(P^e_{\alpha,\beta})$ les contraintes $x_i = \bar{x}_i$ et $x_j \leq \bar{x}_j - 1$ et dans le deuxième on ajoute le contraintes $x_i = \bar{x}_i$ et $x_j \geq \bar{x}_j + 1$

(c) La branche où $x_i = \bar{x}_i$, $x_j = \bar{x}_j$ et $y_{ij} = \bar{x}_i\bar{x}_j$. Dans le sous arbre, on ajoute à la formulation de $(P^e_{\alpha,\beta})$ les contraintes $x_i = \bar{x}_i$, $x_j = \bar{x}_j$ et $y_{ij} = \bar{x}_i\bar{x}_j$.

Remarquons que dans tous les cas $\bar{x}$ est une solution réalisable de (QP) et $f(\bar{x})$ est une borne supérieure de la valeur de la solution optimale de (QP).

La Figure 5 présente les branchements possibles à un noeud donné de l'arbre de résolution, et l'Algorithme 1 présente l'algorithme récursif de notre Branch and Bound spécifique. Pour trouver la solution optimale de (QP), il faut appeler le programme $Branch_\&_Bound(P^e_{\alpha,\beta})$ après avoir initialisé la borne supérieure de la façon suivante : $Borne_sup = +\infty$.

FIG. 5.1 – Les branchements possibles dans l'algorithme de Branch and Bound basé sur les solutions de $(P^e_{\alpha,\beta})$



Cet algorithme permet de résoudre à chaque noeud du Branch and Bound le problème $(P^e_{\alpha,\beta})$ qui est de plus petite taille que le problème $(QP_{\alpha,\beta})$.

Algorithme 1 $Branch_\&_Bound(P)$

Résoudre (P) et en déduire une solution $(\bar{x}, \bar{y})$ {la solution $(\bar{x}, \bar{y})$ est entière}
si $(f_{\alpha,\beta}(\bar{x}, \bar{y}) > Borne_sup)$ **alors**
  Arrêter
finsi
si $\forall (i,j) \in I^2$, $\bar{x}_i \bar{x}_j = \bar{y}_{ij}$ **alors**

  si $f_{\alpha,\beta}(\bar{x}, \bar{y}) < Borne_sup$ **alors**
    $Borne_sup \leftarrow f_{\alpha,\beta}(\bar{x}, \bar{y})$
  finsi
  Arrêter
sinon
  $\{\exists (i,j) \in I^2$, tels que $\bar{x}_i \bar{x}_j \neq \bar{y}_{ij}\}$

  si $f(\bar{x}) < Borne_sup$ **alors**
    $Borne_sup \leftarrow f(\bar{x})$
  finsi
  Ajouter à (P) la contrainte $x_i \leq \bar{x}_i - 1$
  $Branch_\&_Bound(P)$
  Ajouter à (P) la contrainte $x_i \geq \bar{x}_i + 1$
  $Branch_\&_Bound(P)$
  Ajouter à (P) les contraintes $x_i = \bar{x}_i$ et $x_j \leq \bar{x}_j - 1$
  $Branch_\&_Bound(P)$
  Ajouter à (P) les contraintes $x_i = \bar{x}_i$ et $x_j \geq \bar{x}_j + 1$
  $Branch_\&_Bound(P)$
  Ajouter à (P) les contraintes $x_i = \bar{x}_i$, $x_j = \bar{x}_j$ et $y_{ij} = \bar{x}_i \bar{x}_j$
  $Branch_\&_Bound(P)$
finsi

Chapitre 6

Résultats expérimentaux

Ce chapitre présente une synthèse des résultats des expérimentations que nous avons effectuées sur les méthodes présentées dans cette thèse. Le détails des résultats pour chaque instance se trouve en Annexe B. Ce chapitre se décompose en deux parties. D'abord, nous évaluons expérimentalement la qualité des reformulations sur les problèmes ayant des variables entières, puis sur ceux ayant des variables binaires.

Ainsi, dans une première partie nous évaluons la qualité de nos reformulations sur des problèmes quadratiques non convexes soumis à contraintes linéaires et à variables entières. Comme il existe très peu de littérature sur ce type de problèmes et que nous n'avons pas réussi à générer les mêmes instances que celles de Körner [27], nous avons choisi de générer aléatoirement deux classes de problèmes. La classe de problèmes $EIQP$(Equality Integer Quadratic Problem) qui consiste à minimiser une fonction quadratique non convexe soumise à 1 contrainte d'égalité, et la classe de problèmes $IIQP$(Inequality Integer Quadratic Problem) qui consiste à minimiser une fonction quadratique non convexe soumise à 1 contrainte d'inégalité. Nous testons donc nos reformulations linéaires `BBL`, `BBLr`, `BIL` et `BILr`, nos reformulations quadratiques convexes `NC`, `IQCR`, `IQCRs` et `CQCR` et notre algorithme spécifique de Branch and Bound sur ces deux classes de problèmes.

Ensuite, nous évaluons expérimentalement la qualité de la reformulation quadratique convexe `IQCR` sur des programmes quadratiques en variables binaires. Il est intéressant de comparer `IQCR` avec ses "parents" `UQCR` et `QCR`. Nous avons choisi pour cela de tester notre méthode sur deux classes de problèmes. La première concerne des problèmes non contraints qui sont les instances générées aléatoirement par Pardalos et Rodgers [33]. Ensuite, nous avons évalué `IQCR` sur une classe de problèmes soumis à des contraintes d'égalité afin de comparer `IQCR` avec `QCR`. Nous avons choisi pour cela des instances du problème d'affectation de tâches que l'on peut retrouver dans [4] et dans [6].

Environnement Expérimental

Toutes nos expérimentations ont été exécutées sur un PC ayant un processeur Intel core 2 duo cadencé à $2.8GHz$ ayant 2048 MB de RAM, et dont le système d'exploitation est Linux. Nous avons utilisé le langage de modélisation ampl et le solveur Cplex version 11 [24] pour résoudre les programmes quadratiques convexes en variables mixtes, et les solveurs CSDP [8] et SB [20] pour résoudre les programmes semi-définis. Plus précisément, nous avons utilisé le solveur CSDP pour résoudre les programmes semi-définis associés aux programmes en variables entières et SB pour ceux associés aux programmes en variables binaires. Nous avons fait ce choix pour des raisons de performances. En effet, SB est plus performant que CSDP sur des problèmes ayant un grand nombre de contraintes et lorsque les variables sont binaires, par contre dès que les variables sont entières le comportement de SB est incertain. Cela est sans doutes dû à la nécessité d'évaluer la trace de la diagonale de la matrice variable, ce qui est difficile lorsque les bornes sur les variables sont grandes. Pour les calcul de valeurs propres nous avons utilisé Scilab [25].

6.1 Résultats expérimentaux pour les problèmes quadratiques en variables entières soumis à des contraintes linéaires

6.1.1 Présentation des classes de problèmes étudiées

La première classe de problèmes à laquelle nous nous sommes intéressés est la classe $(EIQP)$, qui consiste à minimiser une fonction quadratique non convexe, soumise à 1 contrainte d'égalité et à des contraintes de bornes sur ses variables. Plus formellement, nous avons résolu des instances du problème suivant :

$$(EIQP)\begin{cases} Min & x^T Q x + c^T x & \\ s.t. & \sum_{i=1}^{n} a_i x_i = b & \\ & 0 \leq x_i \leq u_i & i \in I \\ & x_i \in \mathbb{N} & i \in I \end{cases}$$

La deuxième classe de problèmes à laquelle nous nous sommes intéressés est la classe $(IIQP)$, qui consiste à minimiser une fonction quadratique non convexe, soumise à 1 contrainte d'inégalité et à des contraintes de bornes sur ses variables. Plus formellement, nous avons résolu des instances du problème suivant :

$$(IIQP)\begin{cases} Min & x^TQx + c^Tx \\ s.t. & \sum_{i=1}^{n} d_i x_i \leq e \\ & 0 \leq x_i \leq u_i \quad i \in I \\ & x_i \in \mathbb{N} \quad\quad i \in I \end{cases}$$

Pour ces deux classes d'instances, nous avons généré aléatoirement trois types de problèmes de la façon suivante :

Les problèmes $(EIQP_1)$ et $(IIQP_1)$

- Les coefficients de la matrice Q et du vecteur c sont des entiers uniformément distribués dans l'intervalle $[-100, 100]$. Plus précisément, pour tout $i < j$, un nombre ν est généré dans l'intervalle $[-100, 100]$, et on pose alors $q_{ij} = q_{ji} = \nu$.
- Les coefficients a_i et d_i sont des entiers uniformément distribués dans l'intervalle $[1, 50]$.
- Le coefficient $b = 15 * \sum_{i=1}^{n} a_i$ et le coefficient $e = 15 * \sum_{i=1}^{n} d_i$.
- $u_i = 30$, pour tout $i \in I$.

Notons que dans de telles instances la solution $x_i = 15$ pour tout i est une solution réalisable.

Les problèmes $(EIQP_2)$ et $(IIQP_2)$

- Les coefficients de la matrice Q et du vecteur c sont des entiers générés comme dans $(EIQP_1)$ et $(IIQP_1)$.
- Les coefficients a_i et d_i sont des entiers uniformément distribués dans l'intervalle $[1, 100]$.
- Le coefficient $b = 20 * \sum_{i=1}^{n} a_i$ et le coefficient $e = 20 * \sum_{i=1}^{n} d_i$.
- $u_i = 50$, pour tout $i \in I$.

Notons que dans de telles instances la solution $x_i = 20$ pour tout i est une solution réalisable.

Les problèmes $(EIQP_3)$ et $(IIQP_3)$

- Les coefficients de la matrice Q et du vecteur c sont des entiers générés comme dans $(EIQP_1)$ et $(IIQP_1)$.
- Les coefficients a_i et d_i sont des entiers générés comme dans $(EIQP_2)$ et $(IIQP_2)$.
- Le coefficient b et e sont des entiers générés comme dans $(EIQP_2)$ et $(IIQP_2)$.
- $u_i = 70$, pour tout $i \in I$.

Notons que dans de telles instances la solution $x_i = 20$ pour tout i est une solution réalisable.

Pour chaque problème et pour chaque $n = 20$, 30 ou 40 nous avons généré 5 instances et donc un total de 90 instances.

6.1.2 Les résultats

Les résultats pour la classe de problèmes ($EIQP$) sont présentés dans les tableaux 6.1 pour les problèmes ($EIQP_1$), 6.2 pour les problèmes ($EIQP_2$), et 6.3 pour les problèmes ($EIQP_3$). Les résultats pour la classe de problèmes ($IIQP$) sont présentés dans les tableaux 6.4 pour les problèmes ($IIQP_1$), 6.5 pour les problèmes ($IIQP_2$), et 6.6 pour les problèmes ($IIQP_3$). Dans chaque tableau, une ligne correspond à la moyenne sur 5 instances. La légende des tableaux est la suivante :

Légendes des Tableaux :

- *nom* : `Problem_n`, où `n` est le nombre de variables de (QP).
- Les colonnes `BBL`, `BBLr`, `BIL`, `BILr`, `NC`, `IQCR`, `IQCRs`, `CQCR` et B&B présentent les résultats par méthode.
- *ig (gap initial)* : $\left|\frac{opt-l}{opt}\right| * 100$ où l est la valeur optimale de la relaxation continue à la racine de l'arbre de résolution.
- *temps* : temps CPU (en secondes) nécessaire à l'algorithme de Branch and Bound. Le temps limite étant fixé à 1 heure, *tps (i)* est le temps moyen *tps* de résolution de i instances sur 5 lorsque i instances ont été résolues en moins d'1 heure. Si (i) est omis alors les 5 instances ont été résolues en moins d'1 heure. Le symbole $-$ signifie qu'aucune instance n'a été résolue en moins d'1 heure. Ce temps ne tient pas compte du temps de résolution du SDP.
- *noeuds* : nombres de noeuds visités pendant l'algorithme de Branch and Bound.
- *fg (gap final)* : $\left|\frac{opt-b}{opt}\right| * 100$ où b est la valeur de la meilleure borne inférieure après 1 heure de résolution. Lorsque les instances on été résolues en moins d'1 heure, le gap final vaut 0 et n'est pas pris en compte dans les moyennes présentées ici. Notons que le gap final de la méthode de Branch and Bound $\left|\frac{opt-ent}{opt}\right| * 100$, où *ent* est la meilleure solution admissible entière trouvée après 1 heure de résolution.

TAB. 6.1 – Résultats pour le classe d'instances ($EIQP_1$)

	($EIQP_1$_20)							
	BBL	BBLr	BIL	BILr	NC	IQCR	CQCR	B&B
ig	20.57	8.81	16.50	4.72	11.93	0.09	1.07	0.09
temps	1495.94 (3)	687.43	66.44	148.58	329.29	19.97	0.99	2.4
noeuds	16506	1083	3316	491	510892	585	1772	8
fg	0.44	0	0	0	0	0	0	0
	($EIQP_1$_30)							
	BBL	BBLr	BIL	BILr	NC	IQCR	CQCR	B&B
ig	30.00	16.69	25.64	12.56	15.52	0.04	1.12	0.04
temps	-	-	2122.86 (3)	1561.4 (2)	-	65.37	2.34	43.6
noeuds	3327	664	12024	1329	7818973	407	2602	49
fg	12.53	11.46	2.97	1.02	6.03	0	0	0
	($EIQP_1$_40)							
	BBL	BBLr	BIL	BILr	NC	IQCR	CQCR	B&B
ig	30.68	19.75	26.31	15.34	15.88	0.06	1.45	0.06
temps	-	-	1993.22(1)	-	-	1002.75 (4)	12.14	172.6 (3)
noeuds	784	112	10092	412	6140591	3586	10687	416
fg	24.96	19.11	19.02	12.04	9.16	0	0	0.06

TAB. 6.2 – Résultats pour le classe d'instances ($EIQP_2$)

	($EIQP_2$_20)							
	BBL	BBLr	BIL	BILr	NC	IQCR	CQCR	B&B
ig	54.65	33.72	31.40	8.94	18.50	0.09	2.79	0.09
temps	-	-	1675 (3)	690.09	210 (1)	57.2	2.35	31
noeuds	9250	1508	64716	1582	6757122	1174	3948	43
fg	19.22	18.11	1.74	0	3.31	0	0	0
	($EIQP_2$_30)							
	BBL	BBLr	BIL	BILr	NC	IQCR	CQCR	B&B
ig	58.31	47.26	34.90	12.30	19.81	0.12	2.00	0.12
temps	-	-	-	-	-	349.2	7.35	645.25 (4)
noeuds	1437	242	17949	1083	5814106	2394	8313	900
fg	47.26	38.11	20.91	5.96	10.52	0	0	0.08
	($EIQP_2$_40)							
	BBL	BBLr	BIL	BILr	NC	IQCR	CQCR	B&B
ig	62.31	45.30	37.83	16.17	17.64	0.05	1.37	0.05
temps	-	-	-	2826 (1)	-	1173.33 (3)	11.26	539.5 (4)
noeuds	302	7	5282	201	4449236	2316	9932	256
fg	50.16	44.36	36.89	14.92	10.64	0.01	0	0

TAB. 6.3 – Résultats pour le classe d'instances ($EIQP_3$)

	($EIQP_3$_20)							
	BBL	BBLr	BIL	BILr	NC	IQCR	CQCR	B&B
ig	171.50	139.61	54.28	30.56	19.61	0.38	6.20	0.38
temps	-	-	851.5 (4)	631.75 (4)	741 (2)	35.40	3.01	8.75 (4)
noeuds	7179	715	21566	2644	4038944	609	4750	7334
fg	75.90	84.87	2.96	2.92	2.81	0	0	0.5
	($EIQP_3$_30)							
	BBL	BBLr	BIL	BILr	NC	IQCR	CQCR	B&B
ig	179.51	135.65	56.00	31.20	20.20	0.23	5.55	0.23
temps	-	-	-	-	-	484.85	591.72	164.6 (3)
noeuds	665	82	11702	900	4651172	3518	565067	2674
fg	144.47	127.97	26.13	25.59	9.78	0	0	0.3
	($EIQP_3$_40)							
	BBL	BBLr	BIL	BILr	NC	IQCR	CQCR	B&B
ig	182.10	143.29	56.82	36.24	20.26	0.11	4.24	0.11
temps	-	-	-	-	-	1549.5 (4)	877.12	405.5 (4)
noeuds	249	0	2957	200	3510785	2589	748317	635
fg	163.18	138.60	44.88	35.11	13.09	0.03	0	0.26

TAB. 6.4 – Résultats pour la classe d'instances ($IIQP_1$)

	($IIQP_1$_20)							
	BBL	BBLr	BIL	BILr	NC	IQCR	IQCRs	CQCR
ig	32.65	22.96	27.78	17.97	11.33	0.66	0.15	4.25
temps	1678 (3)	648.98	89.54	132.63	393.5 (4)	11.69	6.77	0.74
noeuds	12212	733	1598	288	2004810	155	69	1154
fg	1.05	0	0	0	0	0	0	0
	($IIQP_1$_40)							
	BBL	BBLr	BIL	BILr	NC	IQCR	IQCRs	CQCR
ig	49.59	39.56	44.22	33.83	10.85	0.59	0.09	3.27
temps	-	-	787.5 (2)	1677.75 (2)	2 (1)	114.15	51.20	1.49
noeuds	1903	573	9646	1175	6111656	619	132	1467
fg	20.13	29.37	17.76	4.46	3.02	0	0	0
	($IIQP_1$_40)							
	BBL	BBLr	BIL	BILr	NC	IQCR	IQCRs	CQCR
ig	62.22	53.10	56.74	47.58	11.27	0.90	0.13	4.90
temps	-	-	-	-	-	751.55	471.10	13.88
noeuds	355	63	3356	418	5753881	2067	884	12658
fg	48.20	51.14	46.49	42.86	4.87	0	0	0

TAB. 6.5 – Résultats pour la classe d'instances ($IIQP_2$)

	($IIQP_2$_20)							
	BBL	BBLr	BIL	BILr	NC	IQCR	IQCRs	CQCR
ig	71.08	43.01	43.97	17.68	19.66	6.20	0.09	13.25
temps	-	-	267 (2)	679.01	24.5 (2)	207.6	12.41	1.56
noeuds	6537	1248	39060	1278	4860729	7415	119	2555
fg	34.46	26.74	10.17	0	5.60	0	0	0
	($IIQP_2$_20)							
	BBL	BBLr	BIL	BILr	NC	IQCR	IQCRs	CQCR
ig	56.64	38.06	32.37	9.06	18.08	3.48	0.02	13.47
temps	-	-	-	1733.33 (3)	-	949.5	279.05	396.34
noeuds	1699	249	15573	678	5892848	7045	1279	440033
fg	43.01	35.95	16.79	3.45	9.23	0	0	0
	($IIQP_2$_20)							
	BBL	BBLr	BIL	BILr	NC	IQCR	IQCRs	CQCR
ig	63.76	46.27	38.97	16.53	19.62	2.99	0.02	15.10
temps	-	-	-	-	-	2194 (3)	330.49	984 (1)
noeuds	270	4	5530	171	5137274	3239	457	3009496
fg	53.74	46.21	33.56	15.14	13.11	0.65	0	0.92

TAB. 6.6 – Résultats pour la classe d'instances ($IIQP_3$)

	($IIQP_3$_20)							
	BBL	BBLr	BIL	BILr	NC	IQCR	IQCRs	CQCR
ig	155.73	103.58	44.65	17.54	20.54	4.06	0.05	13.01
temps	-	3312 (1)	777.25 (4)	611.95	76.5 (2)	391.30	17.42	11.90
noeuds	6571	846	20110	1036	3716265	15707	198	20795
fg	86.02	47.93	1.76	0	4.53	0	0	0
	($IIQP_3$_30)							
	BBL	BBLr	BIL	BILr	NC	IQCR	IQCRs	CQCR
ig	171.47	131.42	53.07	32.86	19.29	1.38	0.04	10.92
temps	-	-	-	-	-	1644.66	77.03	45.75
noeuds	608	41	11049	749	4688296	14846	203	54026
fg	136.23	115.50	23.43	24.11	8.87	0	0	0
	($IIQP_3$_40)							
	BBL	BBLr	BIL	BILr	NC	IQCR	IQCRs	CQCR
ig	178.05	147.25	53.14	36.17	20.75	1.18	0.11	11.41
temps	-	-	-	-	-	3439 (1)	713.31	1476 (1)
noeuds	401	0	3103	137	3990954	2017	620	1969425
fg	159.32	143.26	40.84	35.77	13.66	0.95	0	0.74

Comparaison des performances des linéarisations BBL, BBLr, BIL, et BILr

D'abord, nous pouvons remarquer que pour les deux classes de problèmes, les 4 linéarisations suivent la même tendance en termes de gap initial moyen, c'est-à-dire qu'en moyenne c'est la méthode BILr qui fournit le meilleur gap initial, puis la méthode BIL (sauf pour les instances de $(EIQP_1)$ et $(IIQP_1)$ où c'est la méthode BBLr), puis la méthode BBLr et enfin la méthode BBL. Comme cela à été prouvé dans le Chapitre 3, pour toutes les instances des deux classes de problèmes $(EIQP)$ et $(IIQP)$, nous pouvons constater que le gap initial associé à la reformulation BIL est meilleur que celui associé à la reformulation BBL. De plus, le programme $(LP_{\texttt{BIL}})$ a moins de variables et contraintes que le programme $(LP_{\texttt{BBL}})$. Par exemple, les instances de $(IIQP_2)$ où $n = 20$ fournissent des programmes reformulés $(LP_{\texttt{BIL}})$ (resp. $(LP_{\texttt{BBL}})$) avec 2820 (resp. 7260) variables et 5061 (resp. 14421) contraintes. En conséquence, la méthode BIL a de meilleures performances que la méthode BBL dans les trois domaines : gap initial, nombre de noeuds et temps de calcul.

Pour les deux méthodes BBL et BIL, leurs versions renforcées améliorent significativement la valeur de la borne obtenue par relaxation continue. En conséquence, le nombre de noeuds décroît dans ces versions renforcées. Cependant, pour la méthode BBLr l'amélioration du gap n'est pas suffisant pour compenser l'augmentation de la taille du problème reformulé, et finalement, le temps de résolution nécessaire à la méthode BBLr est plus grand que celui nécessaire à la méthode BIL.

Pour la méthode BIL, la version renforcée entraîne une amélioration importante du gap, mais dans ce cas l'amélioration du gap compense largement l'augmentation de taille du programme reformulé, et finalement le temps de résolution nécessaire par la méthode BILr est significativement plus petit que celui nécessaire à la méthode BIL.

Comparaison des performances des convexifications NC, IQCR, IQCRs, et CQCR

Les Tableaux 6.1, 6.2 et 6.3 présentent les résultats pour la classe de problèmes $(EIQP)$. Pour toutes les instances nous pouvons observer que la valeur du gap initial moyen de la méthode IQCR est est bien meilleur que ceux obtenus par les méthodes NC et CQCR. Par exemple, pour $(EIQP_2)$, la moyenne des gaps initiaux sur les 15 instances est de 18.50% pour la méthode NC, 2.79% pour la méthode CQCR et 0.09% pour la méthode IQCR. Cette moyenne est donc divisée par un facteur d'environ 200 entre les méthodes NC et IQCR, et par un facteur d'environ 5 entre les méthodes CQCR et IQCR. Sans surprise, la moyenne du nombres de noeuds est divisé par un facteur 3000 entre les méthodes NC et CQCR et par un facteur 1000 entre les méthodes CQCR et IQCR. De plus, les gaps initiaux moyens pour les méthodes IQCR et CQCR sont assez stables puisqu'ils varient entre 0% et 0.29%, et entre 0.78% et 3.89% respectivement, tandis que ceux de la méthode

NC varient entre 11.22% et 24.44%. Il est intéressant de mentionner le temps de prétraitement associé à la résolution des programmes semi-définis. En moyenne, pour les problème ($EIQP_2$) cela a pris 70 secondes, 600 secondes, and 3500 secondes pour les instances de taille 20, 30, et 40, respectivement. Ce temps est assez conséquent, mais d'après la Remarque 4.3.1 la résolution des programmes semi-définis peut être arrêtée après un temps fixé. Cela pourrait être intéressant pour des instances de grande taille puisque les solveurs SDP trouvent en général une solution de bonne qualité assez rapidement. Toujours pour les problèmes ($EIQP_2$), la résolution des programmes semi-définis pour la méthode CQCR a pris en moyenne 1 seconde, 2 secondes, et 5 secondes pour les instances de taille 20, 30, et 40, respectivement.

Les Tableaux 6.4, 6.5 et 6.6 présentent les résultats pour la classe de problèmes ($IIQP$). Ces résultats suivent les mêmes tendances que ceux de la classe de problèmes ($EIQP$) en termes de gap initial, nombre de noeuds et temps de résolution moyens. Cependant, nous pouvons remarquer que les gaps initiaux des méthodes IQCR et CQCR sont beaucoup moins stables que ceux de la classe de problèmes ($EIQP$). En effet, pour les problèmes ($IIQP_2$) ce gap initial moyen varie pour la méthode IQCR (resp. pour la méthode CQCR) entre 0.03% et 17.80%, (resp. entre 4.36% et 26.86%) avec une moyenne de 4.22% (resp. de 13.94%). Par contre pour la méthode NC, la moyenne du gap initial est à peu près identique à celle des problèmes ($EIQP_2$), puisqu'elle est de 19.12%. Toujours pour les problèmes ($IIQP_2$), le temps de résolution des programmes semi-définis pour la méthode IQCR est en moyenne de 40 secondes, 300 secondes, et 1600 secondes pour les instances de taille 20, 30, et 40, respectivement. Pour la méthode CQCR, ce temps est en moyenne égal à 1 seconde, 2 secondes, et 3 secondes pour les instances de taille 20, 30, et 40, respectivement

Dans ces expérimentations, nous avons également testé l'impact de l'intégration des contraintes d'inégalités au processus de convexification en résolvant les 45 instances de la classe ($IIQP$) avec la méthode IQCRs. Plus précisément, comme cela a été présenté dans la Section 4.3.4, nous avons reformulé chaque instance de ($IIQP_1$), ($IIQP_2$) et ($IIQP_3$) en une instance d'un programme mixte entier soumis à 1 contrainte d'égalité. Nous l'avons ensuite résolu par la méthode décrite dans la Section 4.3.5. Nous pouvons voir clairement l'amélioration en termes de valeur de gap moyen, puisque dans ($IIQP_2$) le gap moyen décroît de 4.22% pour la méthode IQCR à 0.06% pour la méthode IQCRs. De plus, le nombre de noeuds de la méthode IQCRs est réduit d'un facteur 10 en comparaison avec celui de la méthode IQCR. La méthode IQCRs permet de résoudre toutes les instances de la classe ($IIQP$), alors que la méthode IQCR ne résout que 39 instances sur 45. Toujours pour les problèmes ($IIQP_2$), le temps de résolution des programmes semi-définis pour la méthode IQCRs est en moyenne de 250 secondes, 1850 secondes and 9000 secondes, pour les instances de taille 20, 30, et 40, respectivement.

Commentaires sur les résultats du Branch and Bound

Dans ces expérimentations, nous donnons quelques résultats préliminaires de notre algorithme de Branch and Bound qui est basé sur la propriété de projection de la méthode `IQCR`. Pour le moment nous n'avons testé cet algorithme que sur les classes de problèmes ($EIQP$). L'intérêt est de comparer le temps de résolution de cet algorithme avec celui de la méthode `IQCR`. Sur ces instances, le temps de résolution moyen de l'algorithme de Branch and Bound est la plupart du temps plus court que celui de la méthode `IQCR`. Cependant, sur quelques instances, l'instance $EIQP_2_30_1$ par exemple, la méthode `IQCR` est beaucoup plus rapide, 135 secondes, alors que l'algorithme de Branch and Bound ne peut résoudre cette instance en moins d'1 heure de résolution. Il est important de noter que l'implantation de ce Branch and Bound est naïve, puisque pour le moment nous effectuons un parcours en profondeur de l'arbre de résolution. En résolvant cette même instance, $EIQP_2_30_1$, par la méthode `IQCR` avec le parcours en profondeur de son arbre de résolution, celle-ci n'est pas résolue en moins d'une heure de résolution. Ainsi, avec une implantation plus fine cet algorithme sera encore plus efficace : il est donc très prometteur.

Nous résumons les performances de nos différentes méthodes sur les classes de problèmes ($EIQP$) et ($IIQP$) dans les Tableaux 6.7 et 6.7 respectivement.

Légendes des Tableaux 6.7 et 6.7 :

- Les colonnes `BBL`, `BBLr`, `BIL`, `BILr`, `NC`, `IQCR`, `IQCRs`, `CQCR` et B&B présentent les résultats par méthode.
- *nb. inst. résolues* : est le nombres d'instances résolues en moins d'1 heure sur les 45 instances proposées.
- *gap initial* : $\left|\frac{opt - l}{opt}\right| * 100$ où l est la valeur optimale de la relaxation continue à la racine de l'arbre de résolution. Ce gap est le gap moyen sur les 45 instances des classes d'instances ($EIQP$) et ($IIQP$).
- *temps* : temps CPU (en secondes) nécessaire à l'algorithme de Branch and Bound. Ce temps est est le temps moyen sur le nombres d'instances résolues en moins d'1 heure.
- *taille max* : la taille maximale des problèmes que la méthode a pu résoudre en 1 heure de résolution.

TAB. 6.7 – Résultats globaux pour les classes d'instances ($EIQP$)

	($EIQP$)							
	BBL	BBLr	BIL	BILr	NC	IQCR	CQCR	B&B
nb. inst. résolues	3	5	16	17	8	41	45	37
gap initial	87.73	65.56	37.74	18.67	17.70	0.13	2.87	0.13
temps	1495.94	687.43	1341.79	1171.6	426.76	515.28	167.59	223.36
taille max	20	20	40	40	20	40	40	40

TAB. 6.8 – Résultats globaux pour les classes d'instances ($IIQP$)

	($IIQP$)							
	BBL	BBLr	BIL	BILr	NC	IQCR	IQCRs	CQCR
nb. inst. résolues	3	6	13	20	9	39	45	37
gap initial	93.02	69.47	43.89	25.47	16.82	2.38	0.08	9.83
temps	1678	1980.49	480.32	966.874	124.12	1078.16	217.65	325.73
taille max	20	20	40	40	20	40	40	40

Dans ces tableaux nous constatons que les résultats suivent les mêmes tendances pour les deux classes d'instances ($EIQP$) et ($IIQP$). En effet, pour les classes ($EIQP$) (resp. ($IIQP$)), les linéarisations ne peuvent pas résoudre plus de 17 instances (resp. 20 instances) sur les 45 proposées, alors que les convexifications en résolvent jusqu'à 41 instances (resp. 45 instances). En ce qui concerne la qualité des gaps initiaux, nous pouvons remarquer sur ces moyennes que pour les deux classes d'instances ($EIQP$) et ($IIQP$), nous pouvons définir un ordre sur nos méthodes qui est le suivant : `BBL`, `BBLr`, `BIL`, `BILr`, `NC`, `CQCR`, `IQCR`, `IQCRs`.

6.2 Résultats expérimentaux pour les problèmes quadratiques en variables 0-1

Nous comparons ici les résultats de la méthode `IQCR` avec ceux des méthodes `UQCR` et `QCR` sur un problème non contraint et sur deux problèmes soumis à des contraintes d'égalité.

6.2.1 Résultats pour le problème non contraint : les instances de Pardalos et Rodgers (1990)

Nous avons testé `IQCR` sur des instances générées aléatoirement par le générateur de problèmes creux de Pardalos et Rodgers [33] et résolues par la méthode `UQCR` dans [3]. Ces instances avaient été générées de la façon suivante :

- Les termes linéaires du vecteur c, ainsi que les termes diagonaux de la matrice Q ont été tiré aléatoirement dans l'intervalle $[-100, 100]$.
- Les termes quadratiques de la matrice Q ont été tirés aléatoirement dans l'intervalle $[-50, 50]$.
- La matrice Q a une densité égale à $d(\leq 1)$. La densité représente ici la probabilité qu'un élément non nul soit tiré pour les termes non diagonaux de Q.

Pour chacune des 15 paires (n, d), où n est le nombre de variables et d la densité de la matrice Q, Billionnet et Elloumi ont généré 10 instances.

Nous avons refait les expérimentations de [3] pour comparer le temps de calculs entre les méthodes `UQCR` et `IQCR`. Leurs performances sur ces instances sont présentées dans le Tableau 6.9, où chaque ligne correspond à une moyenne sur 10 instances.

TAB. 6.9 – Résultats pour les instances de Pardalos and Rodgers

	UQCR				IQCR			
n_densité	ig	temps (s)	noeuds	fg	ig	temps (s)	noeuds	fg
50_d40	6.1	0.3	910	0	0.56	1.4	69	0
50_d60	5.8	0.3	1034	0	0.56	1.4	78	0
50_d100	6.6	0.5	1367	0	0.99	2.3	140	0
60_d20	5.5	0.5	1768	0	0.14	1.5	20	0
60_d40	9.0	1.4	4366	0	0.90	6.5	236	0
70_d30	6.1	2.8	8841	0	0.72	11.5	250	0
70_d80	7.3	5.4	10899	0	1.31	17.5	443	0
80_d20	5.9	5.4	16046	0	0.45	13.5	185	0
100_d100	7.6	372.7	370358	0	1.86	475.6	5502	0
120_d30	7.1	1263.6	1405507	0	1.53	985	6082	0
120_d80	8.7	3905 (6)	2703558	0.52	2.64	2516 (7)	32902	0.34

Sur toutes ces instances, nous voyons clairement l'amélioration en termes de gap moyen entre la méthode `UQCR` et la méthode `IQCR`. En effet, ce gap moyen sur toutes les instances passe de 6.88% pour la méthode `UQCR` à 1.06% pour la méthode `IQCR`, et est donc divisé par un facteur 6. Par contre, le temps de résolution de la méthode `UQCR` est plus court que celui de la méthode `IQCR` pour les instances de tailles inférieures à 100. Pour les instances de plus grande taille, c'est-à-dire pour celles où la taille est égale à 120, l'amélioration en termes de gap compense l'augmentation de taille du problème reformulé, et `IQCR` a un temps de résolution plus court.

6.2.2 Résultats pour des problèmes avec contraintes d'égalités : le problème d'affectation de tâches

Soit $P = \{P_1, \ldots, P_n\}$ un ensemble de processeurs d'un système distribué, et $T = \{T_1, \ldots, T_m\}$ un ensemble de tâches qui doivent être exécutées sur ce système distribué, sachant qu'une partie des ces tâches doit communiquer.

Le problème d'affectation de tâches peut être formulé comme un problème quadratique en variables $0-1$ soumis à une contrainte d'égalité linéaire :

$$(TA)\begin{cases} min & l(x) = \sum_{i=1}^{m}\sum_{k=1}^{n} e_{ik}x_{ik} + \sum_{i=1}^{m-1}\sum_{j=i+1}^{m}\sum_{k=1}^{n}\sum_{l=1}^{n} c_{ikjl}x_{ik}x_{jl} & \\ s.c. & \sum_{k=1}^{n} x_{ik} = 1 & i = 1, \ldots n \\ & x_{ik} \in \{0,1\} & i = 1, \ldots n, k = 1, \ldots m \end{cases}$$

où le coefficient réel e_{ik} représente le coût d'exécution de la tâche T_i lorsqu'elle est affectée sur le processeur P_k, et le coefficient c_{ikjl} représente le coût de communication entre les tâches T_i et T_j lorsqu'elles sont affectées sur les processeurs P_k et P_l respectivement. Les variables de décision x_{ik} valent 1 si et seulement si la tâche T_i est affectée au processeur P_k.

Afin de comparer `IQCR` avec la méthode `QCR`, nous avons résolu les mêmes instances que celles de [4]. Ces instances sont générées aléatoirement. Les coefficients e_{ik} et c_{ikjl} sont générés aléatoirement dans l'intervalle $[-50, 50]$. Ces instances ont également été utilisées dans [6].

Pour 3 valeurs de $n = (4, 5, 6)$ et 4 valeurs de $m = (15, 18, 20, 25)$, 10 instances ont été générées. Les performances de `QCR` et `IQCR` sur ces instances sont présentées dans le Tableau 6.10, où chaque ligne correspond à une moyenne sur 10 instances.

TAB. 6.10 – Résultats pour le problème d'affectation de tâches

	QCR			IQCR		
nbTaches_nbProc	ig	temps (s)	noeuds	ig	temps (s)	noeuds
15_5	30.44	8.6	22770	3.98	13.0	339
18_4	22.51	4.0	10870	5.50	17.3	472
20_4	24.14	11.5	25621	6.64	48.9	1104
20_5	33.26	346.8	521932	7.81	282.5	4859
25_4	23.95	222.1	313049	8.37	630.2	8686
25_5	35.02	1875.2 (1)	4118083	11.07	1658.5 (3)	50395
25_6	45.14	-	3281184	12.97	1454.7 (1)	54872

Les résultats pour ce problème sont similaires à ceux du problème non contraint. Ici encore nous voyons clairement l'amélioration en termes de gap moyen entre la méthode `QCR` et la mé-

thode IQCR. En effet, ce gap moyen sur toutes les instances passe de 30.63% pour la méthode QCR à 8.04% pour la méthode IQCR. Par contre, le temps de résolution de la méthode QCR est plus court que celui de la méthode IQCR pour les instances de petite taille. Pour les instances de grande taille, ici encore, l'amélioration en termes de gap compense l'augmentation de taille du problème reformulé, et IQCR a un temps de résolution plus court.

6.3 Conclusion

Pour conclure, sur ces deux classes de problèmes la méthode BILr est la meilleure approche de linéarisation pour les trois critères : gap initial, nombre de noeuds et temps de résolution. Cependant, les expérimentations montrent que cette méthode ne peut pas résoudre d'instances ayant 40 variables ou plus en moins d'une heure.

En ce qui concerne les reformulations convexes, pour ces deux classes d'instances NC n'a pas pu résoudre les instances ayant plus de 20 variables en moins d'une heure, tandis que les méthode IQCR, IQCRs et CQCR ont résolu des instances allant jusqu'à 40 variables. De plus, ces expérimentations montrent que IQCR et IQCRs sont les meilleures approches en termes de gap initial. Par contre en termes de temps de calcul, la méthode CQCR peut être plus rapide car la qualité du gap initial ne compense pas toujours la taille du problème reformulé dans les méthodes IQCR et IQCRs.

L'algorithme spécifique de Branch and Bound est prometteur puisqu'il est, le plus souvent, plus rapide que la méthode IQCR alors qu'il est implanté de façon très rudimentaire.

Enfin, la méthode IQCR appliquée aux problèmes en variables binaires améliore en termes de gap initial les méthodes existantes de convexification.

Conclusion

Dans cette thèse nous avons présenté plusieurs approches de résolution de programmes mathématiques en variables entières dont la fonction objectif est quadratique non convexe et est soumise à des contraintes linéaires. Ce type de problème, noté (QP), appartient à la classe des problèmes $\mathcal{NP}$-difficiles. Il existe des algorithmes et des solveurs capables de résoudre efficacement (QP) dans le cas particulier où sa fonction objectif est convexe. C'est pour cela que pour résoudre (QP), nous avons proposé une méthode en deux phases : la première consiste en sa reformulation en un programme équivalent dont la fonction objectif est convexe, et la deuxième consiste en la soumission de ce problème reformulé à un solveur efficace. Deux cas de figures se présentent alors : soit on reformule (QP) en un programme équivalent dont la fonction objectif est linéaire, soit on reformule (QP) en un programme équivalent dont la fonction objectif est quadratique et convexe. Ainsi, nous avons proposé dans cette thèse 4 reformulations linéaires et 3 reformulations quadratiques convexes de (QP) que nous avons étudiées théoriquement et expérimentalement. La Figure 1 récapitule ces différentes reformulations en présentant pour chacune : son nom, son principe, son interprétation en programmation quadratique binaire, et les outils logiciels utilisés pour la résoudre.

Les reformulations linéaires :

Tout d'abord, nous avons proposé la reformulation linéaire `BBL` qui consiste à remplacer chaque variable entière par sa décomposition en variables binaires puis à linéariser les nouveaux produits de deux variables binaires par la linéarisation de Fortet [13]. Nous avons ensuite renforcé cette reformulation en lui ajoutant des inégalités valides dans la méthode appelée `BBLr`. L'étude de cette approche nous montre que les programmes obtenus par ces reformulations possèdent un très grand nombre de variables et de contraintes. De plus, nos expérimentations montrent que leurs bornes obtenues par relaxation continue sont trop faibles. En effet, en moyenne sur les 90 instances proposées, les méthodes `BBL` et `BBLr` ont un gap respectif de 90.37% et de 67.52%. Pénalisées par leurs gaps et par leurs tailles, elles n'ont pu résoudre respectivement que 6 et 11 instances de 20 variables entières en moins d'une heure de temps de résolution.

Type	Nom	Fondements	Interprétation en 0-1	Outils logiciels
LINEARISATION	`BBL`	Décomposition binaire de chaque variable entière et linéarisation des produits de variables binaires	Linéarisation de Fortet [13]	Solveur linéaire Mixte-01
	`BBLr`	Renforcement par des inégalités valides		
	`BIL`	Décomposition binaire de chaque variable entière et linéarisation des produits d'une variable binaire par une variable entière	Linéarisation de Fortet [13]	Solveur linéaire Mixte-01
	`BILr`	Renforcement par des inégalités valides		
CONVEXIFICATION	`NC`	1. Décomposition unaire de chaque variable entière et expression linéaire des carrés des variables 2. Calcul de la plus petite valeur propre	Méthode de la plus petite valeur propre [19]	Scilab + Solveur quadratique Mixte-01
	`IQCR`	1. Décomposition binaire de chaque variable entière et expression linéaire des produits d'une variable binaire par une variable entière 2. Utilisation des contraintes d'égalité pour perturber l'objectif 3. Résolution d'un SDP pour obtenir une convexification optimale	Amélioration de `QCR` [4]	Solveur SDP + Solveur quadratique Mixte-01 ou Branch and Bound spécifique qui utilise un solveur quadratique Mixte-01
	`IQCRs`	Adaptation de `ICQR` à la programmation mixte entière et utilisation des toutes les contraintes (égalités et inégalités) pour perturber l'objectif		
	`CQCR`	Version simplifiée de `IQCR` : expression linéaire des carrés des variables uniquement	`QCR` [4]	Solveur SDP + Solveur quadratique Mixte-01

FIG. 1 – Tableau récapitulatif des méthodes proposées dans cette thèse

Ensuite, nous avons proposé la reformulation linéaire `BIL` qui consiste à remplacer, dans chaque produit de variables entières une des deux variables par sa décomposition binaire. Puis, nous linéarisons chaque nouveau produit d'une variable entière par un variable binaire par la linéarisation de McCormick [31]. Nous avons ensuite renforcé cette reformulation en lui ajoutant des inégalités valides dans la méthode appelée `BILr`. Cette approche réduit significativement le nombre de variables et de contraintes additionnelles en comparaison avec l'approche de la reformulation `BBL`. De plus, nous avons démontré que la borne obtenue par relaxation continue fournie par la reformulation `BIL` est toujours de meilleure qualité que celle fournie par la reformulation `BBL`. Ce résultat est intéressant car il allie une réduction de taille du problème reformulé et une amélioration de sa borne. Nos expérimentations corroborent nos résultats théoriques. En effet, en moyenne sur les 90 instances proposées, les méthodes `BIL` et `BILr` ont un gap respectif de 43.32% et de 22.07%. Elles ont résolu respectivement, en moins d'une heure de temps de résolution, 29 et 37 instances ayant jusqu'à 40 variables.

Les reformulations quadratiques convexes :

Nous avons d'abord proposé la reformulation quadratique convexe `NC` qui consiste à exprimer linéairement les carrés des variables, en utilisant leurs décompositions unaires, puis à convexifier en utilisant la plus petite valeur propre du Hessien de l'objectif de (QP). Cette approche, très facile à mettre en oeuvre, s'est avérée peu efficace expérimentalement. En effet, malgré un gap moyen de 17.26%, qui est de meilleure qualité que ceux des quatre linéarisations, cette approche n'a résolu que 17 instances de 20 variables entières en moins d'une heure de temps de résolution.

Nous avons ensuite proposé une deuxième approche de reformulation quadratique convexe, que nous avons appelée `IQCR`. Elle consiste en la définition d'un schéma général de reformulation quadratique convexe. Dans ce schéma nous ajoutons à l'objectif des fonctions qui s'annulent sur le domaine de notre problème reformulé. Ces fonctions sont construites grâce à l'expression linéaire des produits de variables entières, ainsi qu'à l'aide des contraintes d'égalité. Ensuite, nous cherchons à l'intérieur de ce schéma la meilleure reformulation convexe du point de vue de la valeur de la borne obtenue par relaxation continue. Nous avons démontré que cette meilleure reformulation, `IQCR`, peut être déduite de la solution optimale duale d'une relaxation semi-définie de (QP). Nos expérimentations montrent que cette approche est très efficace en pratique. En effet, sur les 90 instances proposées, le gap moyen fourni par la méthode `IQCR` est de 1.26%. Rappelons que ce gap est aussi celui de la relaxation SDP associée. `IQCR` a résolu 80 instances sur 90 ayant jusqu'à 40 variables en moins d'une heure de temps de résolution.

Puis, nous avons montré comment adapter `IQCR` à une large classe de problèmes de la programmation mixte entière, dans la méthode appelée `IQCRs`. Cette adaptation permet également

d'utiliser les contraintes d'inégalité, en plus des contraintes d'égalité, pour perturber la fonction objectif. En effet, il est toujours possible de transformer les contraintes d'inégalité en contraintes d'égalité à l'aide de variables d'écarts réelles. Cette transformation fournit un programme en variables mixtes entières qu'il est possible de résoudre avec `IQCRs`. Expérimentalement, cette approche s'est avérée très prometteuse. En effet, avec un gap moyen de 0.08%, elle a résolu les 45 instances des problèmes ($IIQP$), alors que la méthode `IQCR` n'a résolu que 39 instances en moins d'une heure de temps de résolution. Ces résultats montrent l'impact et la pertinence d'intégrer toutes les contraintes dans la perturbation de la fonction objectif.

Ensuite, nous avons présenté la méthode `CQCR` qui est une simplification de la méthode `IQCR`. Cette méthode fournit une reformulation quadratique convexe de (QP) choisie dans un sous-ensemble des reformulations possibles dans le schéma associé à `IQCR`. En effet, au lieu de perturber la fonction objectif avec les expressions linéaires de tous les produits de variables, elle ne la perturbe qu'avec les expressions linéaires des carrés des variables. Cette reformulation étant plus compacte, elle induit sur certaines instances un temps de résolution réduit. Cependant, par construction, elle fournit une borne par relaxation continue qui est inférieure ou égale à celle fournie par la méthode `IQCR`. Dans nos expérimentations le gap associé à `CQCR` est de 6.35% en moyenne sur les 90 instances proposées. De plus, elle ne peut ni être adaptable à la programmation en variables mixtes entières, ni intégrer les contraintes d'inégalité dans la perturbation de la fonction objectif. L'intérêt de cette approche est donc expérimental, puisqu'elle est nettement plus rapide que `IQCR` pour certaines instances.

Finalement, nous avons appliqué les trois méthodes `NC`, `CQCR` et `IQCR` à la programmation quadratique binaire. Nous avons montré que les méthodes `NC` et `CQCR` sont équivalentes respectivement à la méthode de la plus petite valeur propre et à la méthode `QCR`. Un résultat intéressant est que `IQCR` constitue une amélioration des méthodes existantes pour la programmation quadratique binaire. Nos expérimentations confirment notre étude théorique, puisque le gap moyen sur les instances de la programmation quadratique binaire proposées dans cette thèse est divisé par un facteur 5 en moyenne entre la méthode `QCR` et la méthode `IQCR`.

Un algorithme spécifique de Branch and Bound :

Nous avons présenté un algorithme de Branch and Bound de résolution de `IQCR` basé sur la propriété de projection utilisée dans la preuve de la méthode `IQCR`. Malgré une implantation naïve, cet algorithme s'est avéré compétitif avec la résolution directe du problème non projeté de la reformulation `IQCR`, par un solveur standard MIQP. En effet, nos résultats expérimentaux sur les classes de problèmes ($EIQP$) montrent que cet algorithme a un temps moyen de résolution divisé par un facteur 2 par rapport à celui de la résolution par un solveur.

Pour conclure, cette thèse étudie théoriquement et compare expérimentalement des reformulations linéaires et des reformulations quadratiques convexes des programmes quadratiques en nombres entiers. Les résultats montrent que les reformulations quadratiques convexes sont les plus performantes. De plus, ces convexifications initialement construites pour les problèmes en nombres entiers améliorent les méthodes existantes pour le cas particulier de la programmation quadratique binaire.

Résumé des principaux résultats de cette thèse :

- Reformulations linéaires :

  La reformulation `BIL` fournit un programme linéaire mixte-01, de taille plus petite que celui qui est fourni par la reformulation `BBL`, et de façon surprenante, cette réduction de taille vient avec un borne obtenue par relaxation continue de meilleure qualité.

- Reformulations quadratiques convexes :
  - La reformulation `NC` est une reformulation convexe de taille raisonnable qui fournit, en moyenne et sur les instances étudiées dans cette thèse, une borne obtenue par relaxation continue de meilleure qualité que toutes les linéarisations proposées dans cette thèse.
  - La reformulation `IQCR` est une reformulation optimale au sein de son schéma de reformulations quadratiques convexes. Ce schéma perturbe la fonction objectif à la fois avec tous les produits de variables et avec les contraintes d'égalités. De plus il s'avère que `IQCR` est efficace expérimentalement et qu'elle est la meilleure reformulation proposée dans cette thèse en termes de borne obtenue par relaxation continue.
  - La borne par relaxation du programme reformulé par `IQCR` est identique à celle d'une relaxation SDP.
  - La reformulation `IQCR` appliquée à la programmation quadratique binaire constitue une amélioration de la reformulation `QCR`.
  - La reformulation `IQCRs` est une adaptation de la méthode `IQCR` à une classe importante de problèmes mixtes entiers. De plus, `IQCRs` appliquée à la programmation mixte-01 constitue une amélioration de l'existant. En effet, à notre connaissance, la reformulation quadratique convexe d'un programme mixte-01 n'a jamais été étudiée.
  - La reformulation `IQCRs` permet une intégration directe des contraintes d'inégalité dans la perturbation de la fonction objectif.

- Propriété de projection :

  La propriété de projection du problème reformulé par les méthodes `IQCR` ou `BIL` est à la fois

centrale dans la preuve d'optimalité de la méthode `IQCR`, mais aussi, permet de concevoir un algorithme de Branch and Bound spécifique. Cet algorithme s'avère expérimentalement, dans une implantation rudimentaire, concurrentiel avec la résolution directe du problème reformulé par Cplex MIQP.

– Solveur :

  Le solveur `SIQP (Solution of Integer Quadratic Programming)` de programmes quadratiques en variables mixtes entières.

Bibliographie

[1] Audet, C., Hansen, P., Savard, G. : Essays and Surveys in Global Optimization. GERAD 25th Anniversary Series, Springer, New York, (2005)

[2] Bellman, R., Fan, K. : On systems of linear inequalities in Hermitian matrix variables. Proc. Sympos. Pure Math, (VII), 1-11 (1963)

[3] Billionnet, A., Elloumi, S. : Using a mixed integer quadratic programming solver for the unconstrained quadratic 0-1 problem.Mathematical Programming, (109), 55-68 (2007)

[4] Billionnet, A., Elloumi, S., Plateau, M.-C. : Improving the performance of standard solvers for quadratic 0-1 programs by a tight convex reformulation : the QCR method. Discrete Applied Mathematic, (6), 1185-1197 (2009)

[5] Billionnet, A., Elloumi, S., Lambert, A. : Linear Reformulations of Integer Quadratic Programs. MCO 2008, september 8-10, 43-51 (2008)

[6] Billionnet, A., Elloumi, S. : Best reduction of the quadratic semi-assignment problem. Discrete Applied Mathematics. (109), 197-213 (2001)

[7] Bonami,P., Biegler, L., Conn, A., Cornuéjols, G., Grossmann, I., Laird, C., Lee, J., Lodi, A., Margot, F., Sawaya, N., Waechter, A. : An algorithmic framework for convex mixed integer nonlinear programming. Discrete Optimization. 5, 186-204, (2005)

[8] Borchers, B. : CSDP, A C Library for Semidefinite Programming. Optimization Methods and Software. 11(1), 613-623 (1999)

[9] Cui, Y. : Dynamic programming algorithms for the optimal cutting of equal rectangles. Appl.Math. Model. 29, 1040-1053 (2005)

[10] Dash Optimization, Xpress-Mosel version 1.6.1. : Xpress-Mosel language Reference Manual 1.4., http ://www.dashoptimization.com/ (2005)

[11] Duffin, R.J. : Infinite programs. In Linear inequalities and related systems. Annals of Mathematics. (38), 157-170 (1956)

[12] Floudas, C.A. : Deterministic Global Optimization. Kluwer Academic Publishing, Dordrecht, The Netherlands, (2000)

[13] Fortet, R. : Applications de l'Algèbre de Boole en Recherche Opérationnelle. Revue Française d'Informatique et de Recherche Opérationnelle. (4), 17-25 (1960)

[14] Fu, H.L., Shiue, L., Cheng, X., Du, D.Z., Kim, J.M. : Quadratic Integer Programming with Application in the Chaotic Mappings of Complete Multipartite Graphs. J. Optim. Theory Appl. 110 (3), 545-556 (2001)

[15] Gallo, G., Hammer, P.L., Simeone, B. : Quadratic Knapsack problems. Math. programming study. (12), 132-149 (1980)

[16] Garey, M.R., Johnson, D.S. : Computers and Intractability : A guide to the theory of NP-Completness. W.H. Freeman, San Francisco, CA, (1979)

[17] GLOBALLib, www.gamsworld.org/global/globallib/globalstat.htm

[18] Gupta, O.K., Ravindran, V. : Branch and bound experiments in convex nonlinear integer programming. Management Science. (31), 1533-1546 (1985)

[19] Hammer, P.L., Rubin, A.A. : Some remarks on quadratic programming with 0-1 variables. Revue Française d'Informatique et de Recherche Opérationnelle, (1970).

[20] Helmberg, C. : A c++ implementation of the spectral bundle method. Manual version 1.1.1. http ://www-user.tu-chemnitz.de/ helmberg/semidef.html (2000)

[21] Helmberg, C., Rendl, F. : Solving quadratic (0-1)-problems by semidefinite programs and cutting planes. Mathematical Prgramming. (82), 291-35 (1998)

[22] Horn, R.A., Johnson, C.R. : Matrix analysis. Cambridge University Press, Cambridge, (1985)

[23] Hua, Z.S., Banerjee, P. : Aggregate line capacity design for PWB assembly systems. Int. J.Prod. Res. 38 (11), 2417-2441 (2000)

[24] ILOG. ILOG CPLEX 11.0 Reference Manual. ILOG CPLEX Division, Gentilly, (2008)

[25] INRIA, Scilab version 5.1.1, http ://www.scilab.org/ (2006)

[26] Karmarkar, N. : A new polynomial time algorithm for linear programming. Combinatorica. (4), 373-395 (1984)

[27] Körner, F. : A New Bound for the Quadratic Knapsack Problem and Its Use in a Branch and Bound Algorithm. Optimization. (17), 643-648 (1986)

[28] Kovács :, L.B. : Combinatorial methods of discrete programming. Akadémiai Kiadó. Budapest. (1980)

[29] Leyffer, S. : Integrating SQP and branch-and-bound for mixed integer nonlinear programming. Computational Optimization and Applications. (18), 295-309 (2001)

[30] Liberti, L., Maculan, N. : Global Optimization : From Theory to Implementation, Chapter : Nonconvex Optimization and Its Applications. Springer, New York, (2006)

[31] McCormick, G.P. : Computability of global solutions to factorable nonconvex programs : Part I - Convex underestimating problems. Mathematical Programming. 10(1), 147-175 (1976)

[32] Nemirovskii, N., Nemirovskii, A. : Interior-point polynomial algorithms in convex programming. SIAM. (13) (1994)

[33] Pardalos, P.M., Rodgers, G.P. : Computational aspects of a branch and bound algorithm for quadratic 0-1 progrmming. Comput, (45), 131-144 (1990)

[34] Plateau, M.-C. : Reformulations quadratiques convexes pour la programmation quadratique en variables 0-1. Thèse CEDRIC, (2006)

[35] Rockafellar, R.T. : Convex analysis. Mathematical Series, (28), (1970).

[36] Roupin, F. : Semidefinite relaxations of the Quadratic Assignment Problem in a Lagrangian Framework. Int. J. of Mathematics in Operational Research. 1(1), 144-162 (2009)

[37] Saxena, A., Bonami, P., Lee, J. : Disjunctive Cuts for Non-Convex Mixed Integer Quadratically Constrained Programs. IPCO. Bologna, (2008)

[38] Shelari, H.D., Adams, W.P. : A tight linearization and an algorithm for zero-one quadratic programming problems. Management Science. 32(10), 1274-90 (1986)

[39] Tawarmalani, M., Sahinidis, N.V. : Global optimization of mixed-integer nonlinear programs : A theoretical and computational study. Mathematical Programming. 99 (3), 563-591 (2004)

[40] Tawarmalani, M., Sahinidis, N.V. : Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming. Kluwer Academic Publishing, Dordrecht, The Netherlands, (2002)

[41] Vandenberghe, L., Boyd, S. : Semidefinite programming. SIAM. 38(1), 49-95 (1996)

[42] Volkovich, O.V., Roshchin, V.A., Sergienko, I.V. : Models and methods of solution of quadratic integer programming problems. Cybernetics. (23), 289-305 (1987)

[43] Wiegele, A. : Nonlinear Optimization Techniques Applied to Combinatorial Optimization Problems. Dissertation. (2006)

Annexes

Annexe A

Algorithmes et outils de calcul de la programmation semi-définie

Dans ce chapitre nous présentons un rappel théorique sur la programmation semi-définie ainsi que les deux algorithmes que nous utilisons pour résoudre nos programmes semi-définis. Nous présentons d'abord l'algorithme CSDP [8], qui est une méthode de points intérieurs, puis l'algorithme SB [20], qui est une méthode de faisceaux.

A.1 Rappel : les matrices semi-définies positives

Dans cette section nous reprenons l'état de l'art de la thèse d'Angelika Wiegele [43] sur la programmation semi-définie positive.

En algèbre linéaire, la notion de matrice semi-définie positive est analogue à celle de nombre réel positif. Cette notion s'applique aux matrices symétriques réelles ou complexes. Dans cette partie nous rappelons les définitions ainsi que quelques propriétés des matrices semi-définies positives.

Théorème A.1.1 *Soit X une matrice symétrique réelle carrée, i.e. $X \in \mathbf{S_n}$, il y a équivalence entre :*

i) X *est semi-définie positive (SDP).*

ii) $y^T X y \geq 0 \ \forall y \in \mathbb{R}^n$.

iii) $\lambda_{min}(X) \geq 0$.

iv) *Tous les déterminants des mineurs principaux de* X *sont* ≥ 0.

Rappelons également quelques propriétés qui caractérisent les matrices semi-définies positives :

Lemme A.1.1 *Toute sous matrice carrée d'une matrice définie positive est définie positive.*

Définition A.1.1 $A \succeq 0$ *ssi* $tr(AB) \geq 0 \ \forall B \succeq 0$.

Lemme A.1.2 *Si* $A, B \succeq 0$ *alors* $\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}b_{ij} \geq 0$, *et* $\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}b_{ij} = 0$ *ssi* $AB = \mathbf{0}$

Observation A.1.1 *Si* $A \succeq 0$ *et* $a_{ii} = 0$, *pour un certain* i, *alors* $a_{ij} = a_{ji} = 0 \ \forall j \in \{1, \ldots, n\}$

A.2 La programmation semi-définie

La programmation semi-définie (SDP) est une extension de la programmation linéaire (LP). C'est une discipline qui a été étudiée pour la première fois dans les années 60, par Bellman et Fan [2]. Elle consiste à optimiser dans le cône des matrices SDP. Un problème SDP est donc un programme dont l'objectif est linéaire en les termes de la matrice variable X et dont les contraintes sont également linéaires en les termes de cette matrice. Le problème primal SDP peut se formuler de la façon suivante. Étant donné $C, A^1, \ldots, A^m \in \mathbf{S_n}$, et $b \in \mathbb{R}^m$, et en notant a_{ij}^k le terme (i,j) de la matrice A^k, et c_{ij} le terme (i,j) de la matrice C, le primal $(PSDP)$ s'écrit sous la forme :

$$(PSDP) \begin{cases} Max & \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}X_{ij} \\ S.c. & \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}^k X_{ij} = b_k \quad k = 1, \ldots m \\ & X \in \mathbf{S_n}, X \succeq 0 \end{cases}$$

Comme en programmation linéaire, on définit le problème dual SDP. En introduisant $y \in \mathbb{R}^m$, les variables duales de $(PSDP)$, le dual de $(PSDP)$ s'écrit sous la forme suivante :

$$(DSDP) \begin{cases} Min & \sum_{k=1}^{m} b_k y_k \\ S.c. & \sum_{k=1}^{m} y_k A^k - C \succeq 0 \\ & y \in \mathbb{R}^m \end{cases}$$

Notons que $\sum_{k=1}^{m} y_k A^k - C \in \mathbf{S_n}$.

Tout d'abord remarquons que, comme en programmation linéaire, $(DSPD)$ provient du dual Lagrangien, en effet, $(PSDP)$ s'écrit aussi :

$$\max_X \{\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}X_{ij} \ : \ \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}^k X_{ij} = b_k, k = 1, \ldots m \, , \, X \succeq 0\}$$

Et son Lagrangien est :

$$\mathcal{L}(X,y) = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}X_{ij} - \sum_{k=1}^{m}(\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}^{k}X_{ij} - b_k)y_k$$

$$\mathcal{L}(X,y) = \sum_{i=1}^{n}\sum_{j=1}^{n}(c_{ij} - \sum_{k=1}^{m} a_{ij}^{k}y_k)X_{ij} + \sum_{k=1}^{m} b_k y_k$$

Dont la fonction duale est :

$$L(y) = \max_{X\succeq 0} \mathcal{L}(X,y) = \sum_{i=1}^{n}\sum_{j=1}^{n}(c_{ij} - \sum_{k=1}^{m} a_{ij}^{k}y_k)X_{ij} + \sum_{k=1}^{m} b_k y_k$$

Son dual Lagrangien est donc :

$$(DL) = \min_{y\in\mathbb{R}^m}\max_{X\succeq 0} \mathcal{L}(X,y) = \sum_{i=1}^{n}\sum_{j=1}^{n}(c_{ij} - \sum_{k=1}^{m} a_{ij}^{k}y_k)X_{ij} + \sum_{k=1}^{m} b_k y_k$$

Or il est connu que :

$$\max_{X\succeq 0} \mathcal{L}(X,y) = \begin{cases} \sum_{k=1}^{m} b_k y_k \ ssi \ \sum_{i=1}^{n}\sum_{j=1}^{n}(c_{ij} - \sum_{k=1}^{m} a_{ij}^{k}y_k) \succeq 0 \\ \infty \ sinon \end{cases}$$

donc :

$$(DL) = \min_{y\in\mathbb{R}^m}\{\sum_{k=1}^{m} b_k y_k \ : \ \sum_{i=1}^{n}\sum_{j=1}^{n}(c_{ij} - \sum_{k=1}^{m} a_{ij}^{k}y_k) \succeq 0\}$$

qui est exactement $(DSDP)$.

Pour des solutions satisfaisants les contraintes des $(PSDP)$ et $(DSDP)$, la notion d'admissibilité est définie de la façon suivante en programmation semi-définie :

Définition A.2.1 *(Admissibilité)*

1. *<u>primale</u> : Une matrice $X \succeq 0$ est réalisable pour $(PSDP)$ ssi : $\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}^{k}X_{ij} = b_k,\ k = 1,\dots m.$*
2. *<u>duale</u> : Le vecteur $y \in \mathbb{R}^m$ est réalisable pour $(DSDP)$ ssi $\sum_{k=1}^{m} y_k A^k - C \succeq 0.$*

Définition A.2.2 *(Admissibilité stricte)*

1. *primale* : *Une matrice* $X \succ 0$ *est strictement réalisable pour* $(PSDP)$ *ssi* $\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}^k X_{ij} = b_k,\ k = 1, \dots m.$

2. *duale* : *Le vecteur* $y \in \mathbb{R}^m$ *est strictement réalisable pour* $(DSDP)$ *ssi* $\sum_{k=1}^{m} y_k A^k - C \succ 0.$

A.3 Théorie de la dualité

Définition A.3.1 *(Dualité forte)*
Étant donné X *et* y *solutions réalisables de* $(PSDP)$ *et* $(DSDP)$*, la différence entre les valeurs des objectifs de* $(PSDP)$ *et* $(DSDP)$ *est appelée le saut de dualité. Il est donné par :*

$$\sum_{k=1}^{m} b_k y_k - \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} X_{ij}$$

Proposition A.3.1 *(Dualité faible)*
Étant donné X *et* y *solutions réalisables de* $(PSDP)$ *et* $(DSDP)$ *alors*

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} X_{ij} \leq \sum_{k=1}^{m} b_k y_k$$

Preuve.

$$\sum_{k=1}^{m} b_k y_k - \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} X_{ij} = \sum_{k=1}^{m} (\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}^k X_{ij}) y_k - \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} X_{ij}$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n} X_{ij} (\sum_{k=1}^{m} a_{ij}^k y_k - c_{ij})$$

Or par définition $X \succeq 0$ et $\sum_{k=1}^{m} A^k y_k - C \succeq 0$, donc, par le Lemme A.4.1 :

$$\sum_{i=1}^{n}\sum_{j=1}^{n} X_{ij} (\sum_{k=1}^{m} a_{ij}^k y_k - c_{ij}) \geq 0$$

donc

$$\sum_{k=1}^{m} b_k y_k - \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} X_{ij} \geq 0$$

□

Observation A.3.1 *Si pour X et y solutions admissibles de $(PSDP)$ et $(DSDP)$, le saut de dualité vaut 0, alors il y a dualité forte et X et y sont les valeurs optimales de $(PSDP)$ et $(DSDP)$. Et on a :*

$$\sum_{k=1}^{m} b_k y_k - \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} X_{ij} = 0 \Longleftrightarrow \sum_{i=1}^{n}\sum_{j=1}^{n} X_{ij}(\sum_{k=1}^{m} a_{ij}^k y_k - c_{ij}) = 0$$

Cependant, il n'y a pas nécessairement dualité forte en SDP, comme l'illustre l'exemple suivant de Vandenberghe et Boyd [41] :

Exemple A.3.1

$$(P)\begin{cases} Max & x_{12} \\ S.c. & \begin{pmatrix} 0 & x_{12} & 0 \\ x_{12} & x_{22} & 0 \\ 0 & 0 & 1+x_{12} \end{pmatrix} \succeq 0 \end{cases}$$

Ce problème SDP est posé sous une forme différente de la façon dont nous avons défini un primal SDP, mais on peut le réécrire sous la forme $(PSDP)$:

$$(PSDP)\begin{cases} Max & \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} X_{ij} \\ S.c. & \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}^k X_{ij} = b_k \;\; k = 1, \dots m \\ & X \in \mathbf{S_n}, X \succeq 0 \end{cases}$$

Avec :

$$C = \begin{pmatrix} 0 & 0.5 & 0 \\ 0.5 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad X = \begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{12} & x_{22} & x_{23} \\ x_{13} & x_{23} & x_{33} \end{pmatrix}$$

$$A^1 = \begin{pmatrix} 0 & -0.5 & 0 \\ -0.5 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad A^2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$A^3 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \qquad A^4 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

$$b = (1, 0, 0, 0)$$

Le dual de ce problème sous la forme $(DSDP)$ *est donc :*

$$(DSDP)\begin{cases} Min & y_1 + 0y_2 + 0y_3 + 0y_4 \\ S.c. & y_1A^1 + y_2A^2 + y_3A^3 + y_4A^4 - C \succeq 0 \end{cases}$$

ou encore

$$(D)\begin{cases} Min & y_1 \\ S.c. & \begin{pmatrix} y_2 & (1-y_1)/2 & y_3 \\ (1-y_1)/2 & 0 & y_4 \\ y_3 & y_4 & y_1 \end{pmatrix} \succeq 0 \end{cases}$$

Du fait de l'Observation A.1.1, x_{12}*, doit être égal à 0, la valeur optimale de* $(PSDP)$ *est donc* 0*. Pour les mêmes raisons,* $(1-y_1)/2$ *doit valoir* 0*, dans* $(DSDP)$*, donc* y_1 *doit valoir* 1*, et la valeur optimale de* $(DSDP)$ *est* 1*. Le saut de dualité vaut donc* 1*.*

Ainsi, en SDP, contrairement à la programmation linéaire, le saut de dualité ne vaut pas obligatoirement 0 à l'optimum. Pour déterminer si un problème SDP a un saut de dualité nul à l'optimum, on peut utiliser les contraintes de qualification de Slater.

Définition A.3.2 *(Les contraintes de qualifications de Slater)*

- $(PSDP)$ *satisfait les conditions de Slater si il existe* $X \succ 0$ *tel que* $\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}^k X_{ij} = b_k$, $k = 1, \dots m$.
- $(DSDP)$ *satisfait les conditions de Slater si il existe* $y \in \mathbb{R}^m$ *tel que* $\sum_{k=1}^{m} y_k A^k - C \succ 0$.

Nous pouvons donc en déduire le théorème suivant :

Théorème A.3.1 *Soit* p^* *la valeur optimale de* $(PSDP)$ *et* d^* *la valeur optimale de* $(DSDP)$

- *si* $(PSDP)$ *satisfait les conditions de Slater avec* p^* *fini, alors* $p^* = d^*$ *et cette valeur est atteinte pour* $(DSDP)$ *;*
- *si* $(DSDP)$ *satisfait les conditions de Slater avec* d^* *fini, alors* $p^* = d^*$ *et cette valeur est atteinte pour* $(PSDP)$ *;*
- *si* $(PSDP)$ *et* $(DSDP)$ *satisfont les conditions de Slater alors* $p^* = d^*$ *et cette valeur est atteinte pour les deux problèmes.*

Remarque A.3.1 *Ici,* p^**(respectivement* d^**) est fini si le sup de* $(PSDP)$ *(respectivement* $(DSDP)$*) est atteint.*

Une preuve se trouve par exemple dans Duffin [11], Nesterov et Nemirovskii [32], ou Rockafellar [35]. Evidemmemnt ces conditions ne sont pas respectées dans l'exemple A.3.1.

Définition A.3.3 *Les matrices $Z \succeq 0$ et $X \succeq 0$ sont complémentaires si $ZX = \mathbf{0}$*

Pour les problèmes à dualité forte, nous avons les conditions nécessaires et suffisantes d'optimalité suivantes :

$$\begin{cases} \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}^k X_{ij} = b_k \, \forall k,\ X \succeq 0 \text{ (admissibilité du primal)} \\ \sum_{k=1}^{m} A^k y_k - C \succeq 0,\ y \in \mathbb{R}^m \text{ (admissibilité du dual)} \\ (\sum_{k=1}^{m} A^k y_k - C)X = \mathbf{0} \ \text{ (complémentarité de X et } (\sum_{k=1}^{m} A_k y_k - C)) \end{cases}$$

Preuve. Si le saut de dualité est nul, nous avons vu qu'il existe X et y, solutions optimales de $(PSDP)$ et $(DSDP)$, tels que $\sum_{i=1}^{n}\sum_{j=1}^{n} X_{ij}(\sum_{k=1}^{m} a_{ij}^k y_k - c_{ij}) = 0$ et donc que $(\sum_{k=1}^{m} A^k y_k - C)X = \mathbf{0}$, et donc les matrices $(\sum_{k=1}^{m} A^k y_k - C)$ et X sont complémentaires. □

A.4 Optimisation fondée sur les valeurs propres

Remarquons que la propriété $X \succeq 0 \Leftrightarrow \lambda_{min}(X) \geq 0$ indique une forte relation entre l'optimisation fondée sur les valeurs propres et la SDP. Nous définissons le problème d'optimisation fondé sur les valeurs propres pour $a \in \mathbb{R}, C, A^1, \ldots A^m \in \mathbf{S_n}, b \in \mathbb{R}^m$:

$$(EVP) \min_{y \in \mathbb{R}^m} a\lambda_{max}(C - \sum_{k=1}^{m} A^k y_k) + b^T y$$

Nous allons montrer qu'un problème (EVP) peut se poser comme une instance particulière d'un problème $(DSDP)$ de dimension $m+1$.

Notons tout d'abord qu'il est connu que :

$$\lambda = \lambda_{max}(Z) \Longleftrightarrow Z - \lambda I \preceq 0$$

Nous pouvons donc réécrire (EVP) de la façon suivante :

$$(EVP) \begin{cases} Min\ a\lambda + b^T y \\ \qquad C - \sum_{k=1}^{m} A^k y_k - \lambda I \preceq 0 \\ \qquad y \in \mathbb{R}^m\ ,\ \lambda \in \mathbb{R} \end{cases}$$

Montrons maintenant qu'une instance de $(PSDP)$ peut se formuler sous la forme d'un (EVP)

Faisons l'hypothèse que : $\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}^k X_{ij} = b_k, \forall k \Longrightarrow tr(X) = a \geq 0 \Longleftrightarrow \sum_{i=1}^{n} X_{ii} = a$, en sachant que cette hypothèse est vraie dans beaucoup de relaxations. Ajoutons donc à $(PSDP)$ la contrainte : $\sum_{i=1}^{n} X_{ii} = a$, dont la variable duale associée est $\lambda \in \mathbb{R}$, on obtient les problèmes $(PSDP')$ et $(DSDP')$ suivant :

$$(PSDP') \begin{cases} Max & \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} X_{ij} \\ S.c. & \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}^k X_{ij} = b_k \ k = 1, \dots m \\ & \sum_{i=1}^{n} X_{ii} = a \\ & X \in \mathbf{S_n}, X \succeq 0 \end{cases}$$

$$(DSDP') \begin{cases} Min & \sum_{k=1}^{m} b_k y_k + \sum_{k=1}^{m} a\lambda \\ S.c. & \sum_{k=1}^{m} y_k A^k - C + \lambda I \succeq 0 \\ & y \in \mathbb{R}^m, \ \lambda \in \mathbb{R} \end{cases}$$

Si le vecteur $(y_1, \dots, y_m, \lambda)$ est une solution admissible de $(DSDP')$, alors :

$$Q = \sum_{k=1}^{m} y_k A^k - C + \lambda I \succeq 0$$

et donc toutes les valeurs propres de $-Q$ sont négatives, on a donc la plus grande valeur propre de $-Q$ qui vaut 0, et donc :

$$(\sum_{k=1}^{m} A^k y_k - C + \lambda I) \succeq 0 \Longleftrightarrow \lambda_{max}(-(\sum_{k=1}^{m} A^k y_k - C + \lambda I)) = 0$$

$$\Longleftrightarrow \lambda_{max}(C - \sum_{k=1}^{m} A^k y_k - \lambda I) = 0 \Longleftrightarrow \lambda_{max}(C - \sum_{k=1}^{m} A^k y_k) - \lambda = 0$$

$$\Longleftrightarrow \lambda = \lambda_{max}(C - \sum_{k=1}^{m} A^k y_k)$$

En remplaçant λ dans l'objectif de $(DSDP)$ on obtient :

$$\min_{y\in\mathbb{R}^m} a\lambda_{max}(C-\sum_{k=1}^{m}A^k y_k)+b^T y$$

qui est exactement (EVP)

Un problème de la forme $(PSDP)$ peut donc s'écrire sous la forme d'un problème (EVP) sous l'hypothèse que la trace de sa variable X soit constante et positive.

Conditions d'optimalité :

Pour simplifier l'écriture, notons $f(y)=\lambda_{max}(C-\sum_{k=1}^{m}A_k y_k)+b^T y$ la fonction à minimiser, et considérons que $a=1$ On sait que :

$$(\lambda_{max}(Q))=\begin{cases} Max\sum_{i=1}^{n}\sum_{j=1}^{n}Q_{ij}x_i x_j^T \\ S.c. \quad \sum_{k=1}^{m}y_k A_k+\lambda I-C\succeq 0 \\ \quad\quad ||x||=1 \end{cases}$$

Lemme A.4.1 *$conv(\{xx^T,||x||=1\})=\{W\succeq 0, tr(W)=1\}$ donc $\lambda_{max}(X)=max\{\sum_{i=1}^{n}\sum_{j=1}^{n}W_{ij}X_{ij}$: $tr(W)=1, w\succeq 0\}$*

A.5 Les algorithmes de résolution de la SDP

Les programmes SDP sont des problèmes de minimisation convexe, ils peuvent donc être résolus par des algorithmes polynômiaux, dont les deux plus connus sont d'une part, la méthode des points intérieurs (IPMs), étudiée surtout dans les années 90 mais peu performante pour des problèmes ayant un grand nombre de contraintes ; et d'autre part la méthode Spectral Bundle qui permet de résoudre des problèmes ayant un grand nombre de contraintes.

A.5.1 La méthode des points intérieurs, Borchers, 1999

De nombreuses variantes de cette méthode existent dans la littérature, nous présentons ici la plus "efficace". L'idée est de suivre un chemin central approximatif, dans la région admissible, qui mène à l'optimum. Nous supposons que les contraintes de qualification de Slater sont valides

pour (PSDP) et (DSDP) , c'est-à-dire que :

$$(OPT)\begin{cases} \mathbf{A}(X) = b\,,\ X \succeq 0 \text{ (faisabilité du primal)} \\ \mathbf{A^T}(y) - C = Z\,,\ y \in \mathbf{R^m}\,,\ Z \succeq 0 \text{ (faisabilité du dual)} \\ ZX = 0 \text{ (complémentarité de X et Z)} \end{cases}$$

L'idée est de remplacer $ZX = 0$, par $ZX = \mu I$, avec $\mu \geq 0$, et $\mu \to 0$

D'abord, on définit donc le problème auxiliaire suivant :

$$(PSDP_\mu)\begin{cases} Max & \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}X_{ij} - \mu log(det(X)) \\ S.c. & \mathbf{A}(X) = b \\ & X \succ 0, X \succeq 0 \end{cases}$$

avec μ le paramètre de frontière et $-\mu log(det(X))$, la fonction de frontière et dont le Lagrangien est :

$$\mathbf{L}_\mu(X, y) = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}X_{ij} - \mu log(det(X)) + \sum_{k=1}^{m}(b_k - \mathbf{A}_k(X))y_k$$

Ensuite, on calcule les gradients par rapport à X et y, dans le but de dériver les conditions de KKT, qui sont nécessaires à l'optimalité.

$$\nabla_X \mathbf{L}_\mu(X, y) = C - \mu X^{-1} - \mathbf{A^T}(y)$$

$$\nabla_y \mathbf{L}_\mu(X, y) = b - \mathbf{A}(X)$$

car $\nabla_X log(det(X)) = X^{-1}$

En posant les gradients égaux à 0, on obtient :

$$(OPT_\mu)\begin{cases} \mathbf{A}(X) = b\,,\ X \succ 0 \\ \mathbf{A^T}(y) - C = Z\,,\ y \in \mathbf{R^m}\,,\ Z \succ 0 \\ XZ = \mu I \end{cases}$$

Il a été prouvé que (OPT_μ) a pour chaque $\mu \geq 0$ une unique solution $(X(\mu), y(\mu), Z(\mu))$: le chemin central primal dual.

Appelons $\epsilon = (X, y, Z), X \succ 0, Z \succ 0$, un point quelconque, pas nécessairement sur le chemin central. Le but est de trouver $\nabla\epsilon = (\nabla X, \nabla y, \nabla Z)$ tel que $\epsilon + \nabla\epsilon$ se rapproche du chemin central et qu'à chaque itération μ soit de plus en plus petit jusqu'à l'être suffisamment.

Le système résolu pour trouver la direction et le $\nabla\epsilon$ approprié est le suivant :

$$\mathbf{A}(X + \nabla X) = b$$

$$\mathbf{A^T}(y + \nabla y) - C = Z + \nabla Z$$

$$(X + \nabla X)(Z + \nabla Z) = \mu I$$

Une solution primale-duale (X, y) peut être trouvée en $O(\sqrt{n}|log(\epsilon)|)$

A.5.2 La méthode des faisceaux (Spectral Bundle), Helmberg, 2000

Cette méthode utilise les relations entre la SDP et l'optimisation par la valeur propre. En fait, elle résout le problème (EVP), et donc (DSPD) et (PSDP). Cette méthode a un avantage sur la méthode (IPMs), car elle n'a pas à résoudre à chaque itération un système de taille m, mais résout des problèmes de taille plus petite à chaque itération. Deux éléments composent cette méthode de résolution, le concept de faisceaux et l'idée du point approché.

Considérons que $a = 1$, et notons $f(y) = \lambda_{max}(C - \mathbf{A^T}(y)) + b^T y$ la fonction à minimiser. Définissons $\hat{f}$, une fonction qui approxime f, dans le voisinage de l'itération courante. Introduisons $\mathbf{L}(w, y) = \sum_{i=1}^{n}\sum_{j=1}^{n}(c_{ij} - A_j(y))W_{ij} + b^T y$. Comme $\lambda_{max}(X) = max\{\sum_{i=1}^{n}\sum_{j=1}^{n}(W_{ij}X_{ij}) : W \succeq 0, tr(W) = 1\}$, on peut donc réécrire $f(y) = max\{\mathbf{L}(w, y) : W \succeq 0, tr(W) = 1\}$

Le concept de faisceaux :

Remplaçons la région admissible $\{W : W \succeq 0, tr(W) = 1\}$ par un sous-ensemble plus facilement calculable $\hat{W} = \{\alpha\bar{W} + PVP^T : \alpha + tr(V) = 1, \alpha \geq 0, V \in \mathbf{S_k^+}\}$, où k est le nombre de colonnes de P. On a donc $\hat{f}(y) = max\{\mathbf{L}(w, y) : W \in \hat{W}\}$. P est construit de manière à ce que son nombre maximum de colonnes, r, soit assez petit pour simplifier la résolution informatique, et pour qu'il contienne des informations sur le sous-gradient de l'itération courante. Le paramètre r contrôle également la dimension de V. Pour utiliser toutes les informations sans augmenter la valeur de r, $\bar{W}$ est utilisé comme sous-gradient global.

L'idée du point approché :

Étant donné que nous travaillons avec une approximation de f qui est valide uniquement dans la zone de voisinage de l'itération courante, nous devons restreindre le déplacement du

point courant par :

$$min_y\ \hat{f}(y) + u/2||y - \hat{y}||^2,$$

avec $u \geq 0$, le paramètre de pénalité.

Construction de P et $\bar{W}$:

Une itération de cet algorithme consiste à trouver un nouveau point y_{new}, soit $\hat{y}$ le point courant, y_{new} est obtenu en résolvant $min_y\ \hat{f}(y) + u/2||y - \hat{y}||^2$, avec $u \geq 0$. Ce minorant est obtenu en résolvant d'abord : $max_{W \in \hat{W}}\ \langle C - \mathbf{A^T}(\hat{y}), W\rangle + b^T y - 1/(2u)\langle \mathbf{A}(W) - b, \mathbf{A}(W) - b\rangle$, à l'aide d'un (IPMs). Nous avons donc $W_{new} = \alpha^* \bar{W} + PV^*P^T$.

Algorithme :

La nouvelle itération est calculée facilement : $y_{new} = \hat{y} + 1/u(\mathbf{A}(W_{new}) - b)$. On met à jour la matrice P : tant que P n'a pas r colonnes, on orthogonalise le nouveau vecteur propre de P, par rapport à P, et on l'ajoute comme une nouvelle colonne. Lorsque P contient r colonnes, on obtient les α^* et V^* de cette itération.

Nous présentons maintenant nos expérimentations qui concernent à la fois des problèmes dont les variables sont binaires, mais aussi deux classes de problèmes dont les variables sont à valeurs entières.

Annexe B

Details des résulats expérimentaux

Dans cette annexe nous présentons le détail des résultats expérimentaux. La légende des tableaux est la suivante :

Légendes des Tableaux :

- *nom* : `Problem_n`, où `n` est le nombre de variables de (QP).
- Les colonnes `BBL`, `BBLr`, `BIL`, `BILr`, `NC`, `IQCR`, `IQCRs`, `CQCR` et Branch and Bound présentent les résultats par méthode.
- *ig (gap initial)* : $\left|\frac{opt-l}{opt}\right| * 100$ où l est la valeur optimale de la relaxation continue à la racine de l'arbre de résolution.
- *temps* : temps CPU (en secondes) nécessaire à l'algorithme de Branch and Bound. Le temps limite étant fixé à 1 heure, *tps (i)* est le temps moyen *tps* de résolution de i instances sur 5 lorsque i instances ont été résolues en moins d'1 heure. Si (i) est omis alors les 5 instances ont été résolues en moins d'1 heure. Le symbole $-$ signifie qu'aucune instance n'a été résolue en moins d'1 heure. Ce temps ne tient pas compte du temps de résolution du SDP.
- *noeuds* : nombres de noeuds visités pendant l'algorithme de Branch and Bound.
- *fg (gap final)* : $\left|\frac{opt-b}{opt}\right| * 100$ où b est la valeur de la meilleure borne inférieure après 1 heure de résolution. Lorsque les instances on été résolues en moins d'1 heure, le gap final vaut 0 et n'est pas pris en compte dans les moyennes présentées ici. Notons que le gap final de la méthode de Branch and Bound $\left|\frac{opt-ent}{opt}\right| * 100$, où *ent* est la meilleure solution admissible entière trouvée après 1 heure de résolution.

B.1 Résultats pour la classe de problèmes $(EIQP)$

TAB. B.1 – Résolution de $(EIQP_1)$ par les méthodes `BBL` et `BBLr`

		BBL				BBLr			
	opt	ig	time (s)	nodes	fg	ig	time (s)	nodes	fg
$EIQP_1$_20_1	-5311070	17.83	-	38378	2.42	5.13	1367	2495	0
$EIQP_1$_20_2	-5098379	16.51	2205	17803	0	6.44	558	937	0
$EIQP_1$_20_3	-4554397	27.40	1412	8081	0	14.00	551	842	0
$EIQP_1$_20_4	-5614860	18.62	870	6172	0	3.42	48	22	0
$EIQP_1$_20_5	-4354396	22.50	-	12097	0.12	15.05	910	1119	0
moyenne		**20.57**			**0.44**	**8.81**			**0**
$EIQP_1$_30_1	-10210390	29.60	-	3923	10.45	16.19	-	760	9.09
$EIQP_1$_30_2	-11243370	23.54	-	3998	5.86	10.19	-	769	4.96
$EIQP_1$_30_3	-9862120	28.84	-	3916	11.13	19.46	-	625	13.12
$EIQP_1$_30_4	-10720488	34.95	-	2024	19.53	19.41	-	454	16.29
$EIQP_1$_30_5	-10835084	33.07	-	2776	15.67	18.19	-	716	13.84
moyenne		**30.00**			**12.53**	**16.69**			**11.46**
$EIQP_1$_40_1	-20907112	14.05	-	1412	9.83	5.79	-	174	3.92
$EIQP_1$_40_2	-21274411	27.62	-	501	20.27	14.68	-	242	13.39
$EIQP_1$_40_3	-17033610	40.06	-	908	32.63	29.40	-	0	29.40
$EIQP_1$_40_4	-18268074	30.79	-	482	26.74	19.68	-	132	19.63
$EIQP_1$_40_5	-17373411	40.87	-	621	35.32	29.23	-	13	19.21
moyenne		**30.68**			**24.96**	**19.75**			**19.11**

- : Cplex a été arrêté avec un gap final positif après 1 heure de résolution

TAB. B.2 – Résolution de ($EIQP_1$) par les méthodes `BIL` et `BILr`

		BIL				BILr			
	opt	ig	time (s)	nodes	fg	ig	time (s)	nodes	fg
$EIQP_1$_20_1	-5311070	14.07	82	8622	0	1.33	426	1511	0
$EIQP_1$_20_2	-5098379	12.31	57	2096	0	2.26	57	282	0
$EIQP_1$_20_3	-4554397	23.20	70	997	0	9.70	60	126	0
$EIQP_1$_20_4	-5614860	14.58	42	1233	0	0	2	0	0
$EIQP_1$_20_5	-4354396	18.36	78	3634	0	10.28	195	539	0
moyenne		**16.50**			**0**	**4.72**			**0**
$EIQP_1$_30_1	-10210390	24.97	3224	19495	0	11.84	2826	1482	0
$EIQP_1$_30_2	-11243370	19.39	938	3181	0	6.25	593	275	0
$EIQP_1$_30_3	-9862120	24.41	2205	14012	0	14.62	-	1585	0.04
$EIQP_1$_30_4	-10720488	30.79	-	12341	9.66	15.82	-	1254	5.08
$EIQP_1$_30_5	-10835084	28.63	-	11093	5.21	14.26	-	2052	0.01
moyenne		**25.64**			**2.97**	**12.56**			**1.02**
$EIQP_1$_40_1	-20907112	9.98	1993	32299	0	0.83	-	456	0.04
$EIQP_1$_40_2	-21274411	23.20	-	4422	18.92	10.56	-	371	5.73
$EIQP_1$_40_3	-17033610	35.42	-	3901	30.52	24.95	-	400	22.08
$EIQP_1$_40_4	-18268074	26.60	-	3701	20.53	15.62	-	395	11.61
$EIQP_1$_40_5	-17373411	36.34	-	6138	25.10	24.71	-	440	20.74
moyenne		**26.31**			**19.02**	**15.34**			**12.04**

- : Cplex a été arrêté avec un gap final positif après 1 heure de résolution

TAB. B.3 – Résolution de ($EIQP_1$) par les méthodes `NC` et `CQCR`

		NC				CQCR			
	opt	ig	time (s)	nodes	fg	ig	time (s)	nodes	fg
$EIQP_1$_20_1	-5311070	12.89	146	342802	0	1.24	2	4048	0
$EIQP_1$_20_2	-5098379	10.98	29	81583	0	0.23	0	1184	0
$EIQP_1$_20_3	-4554397	10.15	7	19396	0	1.38	0	54	0
$EIQP_1$_20_4	-5614860	11.92	43	117812	0	0.21	0	6	0
$EIQP_1$_20_5	-4354396	13.70	1419	1992867	0	2.29	1	3571	0
moyenne		**11.93**			**0**	**1.07**			**0**
$EIQP_1$_30_1	-10210390	13.76	-	7689021	4.94	1.36	1	1091	0
$EIQP_1$_30_2	-11243370	13.40	-	7916032	3.65	0.49	0	105	0
$EIQP_1$_30_3	-9862120	12.23	-	8133101	3.32	1.55	2	3536	0
$EIQP_1$_30_4	-10720488	18.23	-	7666601	8.06	1.10	1	1097	0
$EIQP_1$_30_5	-10835084	19.97	-	7690110	10.16 0	1.09	5	7182	0
moyenne		**15.52**			**6.03**	**1.12**			**0**
$EIQP_1$_40_1	-20907112	7.73	-	6459201	1.31	0.50	1	879	0
$EIQP_1$_40_2	-21274411	16.64	-	6037101	10.00	1.54	6	5903	0
$EIQP_1$_40_3	-17033610	17.83	-	6022601	10.84	1.34	6	6236	0
$EIQP_1$_40_4	-18268074	16.02	-	6119381	9.27	1.70	22	20000	0
$EIQP_1$_40_5	-17373411	21.15	-	6064671	14.40	2.19	22	20420	0
moyenne		**15.88**			**9.16**	**1.45**			**0**

- : Cplex a été arrêté avec un gap final positif après 1 heure de résolution

TAB. B.4 – Résolution de ($EIQP_1$) par les méthodes `IQCR` et le Branch and Bound

		IQCR				Branch and Bound			
	opt	ig	time (s)	nodes	fg	ig	time (s)	nodes	fg
$EIQP_1$_20_1	-5311070	0.09	68	2496	0	0.09	3	5	0
$EIQP_1$_20_2	-5098379	0.13	5	16	0	0.13	0	3	0
$EIQP_1$_20_3	-4554397	0.05	7	93	0	0.05	2	3	0
$EIQP_1$_20_4	-5614860	0	0	0	0	0	0	1	0
$EIQP_1$_20_5	-4354396	0.15	18	321	0	0.15	7	27	0
moyenne		**0.09**			**0**	**0.09**			**0**
$EIQP_1$_30_1	-10210390	0.04	21	50	0	0.04	19	18	0
$EIQP_1$_30_2	-11243370	0	4	0	0	0	0	1	0
$EIQP_1$_30_3	-9862120	0.04	89	713	0	0.04	24	19	0
$EIQP_1$_30_4	-10720488	0.05	31	56	0	0.05	16	12	0
$EIQP_1$_30_5	-10835084	0.09	179	1217	0	0.09	159	193	0
moyenne		**0.04**			**0**	**0.04**			**0**
$EIQP_1$_40_1	-20907112	0.04	-	3961	0.01	0.04	241	19	0
$EIQP_1$_40_2	-21274411	0.04	773	1356	0	0.04	276	94	0
$EIQP_1$_40_3	-17033610	0	157	0	0	0	1	1	0
$EIQP_1$_40_4	-18268074	0.03	744	3091	0	0.03	-	802	0
$EIQP_1$_40_5	-17373411	0.21	2337	9526	0	0.21	-	1162	0.3
moyenne		**0.06**			**0**	**0.06**			**0.06**

- : Cplex a été arrêté avec un gap final positif après 1 heure de résolution

TAB. B.5 – Résolution de ($EIQP_2$) par les méthodes `BBL` et `BBLr`

		BBL				BBLr			
	opt	ig	time	nodes	fg	ig	time	nodes	fg
$EIQP_2$_20_1	-9321876	66.14	-	12550	21.30	38.24	-	1550	31.26
$EIQP_2$_20_2	-9013418	79.76	-	5633	23.25	43.06	-	1163	29.47
$EIQP_2$_20_3	-15337225	34.24	-	10377	7.61	25.61	2762	1683	0
$EIQP_2$_20_4	-11863777	44.99	-	11428	13.89	33.70	-	1531	16.11
$EIQP_2$_20_5	-12095004	48.12	-	6486	30.05	27.98	-	1615	13.73
moyenne		**54.65**			**19.22**	**33.72**			**18.11**
$EIQP_2$_30_1	-23592535	64.18	-	1161	51.81	47.31	-	251	46.42
$EIQP_2$_30_2	-25924713	52.48	-	1227	42.45	36.19	-	230	34.07
$EIQP_2$_30_3	-21938906	64.08	-	893	53.18	45.56	-	228	45.51
$EIQP_2$_30_4	-29913305	42.86	-	1906	36.00	27.03	-	251	22.08
$EIQP_2$_30_5	-22422891	67.92	-	1999	52.88	45.84	-	250	42.64
moyenne		**58.31**			**47.26**	**40.39**			**38.11**
$EIQP_2$_40_1	-42548497	59.54	-	209	44.76	43.18	-	7	43.17
$EIQP_2$_40_2	-35957464	78.57	-	67	63.82	52.50	-	1	52.50
$EIQP_2$_40_3	-40116963	62.95	-	280	51.35	48.24	-	0	48.24
$EIQP_2$_40_4	-51306080	40.65	-	513	34.68	30.22	-	27	25.52
$EIQP_2$_40_5	-38090192	69.83	-	445	56.17	52.38	-	0	52.37
moyenne		**62.31**			**50.16**	**45.30**			**44.36**

- : Cplex a été arrêté avec un gap final positif après 1 heure de résolution

TAB. B.6 – Résolution de ($EIQP_2$) par les méthodes `BIL` et `BILr`

		BIL				BILr			
	opt	ig	time	nodes	fg	ig	time	nodes	fg
$EIQP_2$_20_1	-9321876	39.68	1369	43518	0	14.55	1805	4547	0
$EIQP_2$_20_2	-9013418	50.10	-	68454	7.19	22.86	212	379	0
$EIQP_2$_20_3	-15337225	14.65	181	4857	0	0.09	37	97	0
$EIQP_2$_20_4	-11863777	25.18	3475	88451	0	4.41	909	2041	0
$EIQP_2$_20_5	-12095004	27.41	-	118302	1.51	2.81	485	850	0
moyenne		**31.40**			**1.74**	**8.94**			**0**
$EIQP_2$_30_1	-23592535	41.05	-	18235	29.45	17.80	-	978	12.70
$EIQP_2$_30_2	-25924713	30.40	-	16407	15.59	7.51	-	1067	0.11
$EIQP_2$_30_3	-21938906	39.17	-	16535	23.81	18.11	-	971	14.04
$EIQP_2$_30_4	-29913305	21.65	-	21568	8.82	0.36	-	1161	0.05
$EIQP_2$_30_5	-22422891	42.23	-	17000	26.88	17.73	-	1242	6.91
moyenne		**34.90**			**20.91**	**12.30**			**5.96**
$EIQP_2$_40_1	-42548497	34.02	-	4604	33.98	12.55	-	189	10.47
$EIQP_2$_40_2	-35957464	52.02	-	5935	50.22	26.20	-	199	26.07
$EIQP_2$_40_3	-40116963	38.48	-	4437	37.35	18.14	-	93	17.72
$EIQP_2$_40_4	-51306080	20.09	-	6345	20.05	1.85	2826	284	0
$EIQP_2$_40_5	-38090192	44.54	-	5090	42.83	22.11	-	241	20.34
moyenne		**37.83**			**36.89**	**16.17**			**14.92**

- : Cplex a été arrêté avec un gap final positif après 1 heure de résolution

TAB. B.7 – Résolution de ($EIQP_2$) avec les méthodes `NC` et `CQCR`

		NC				CQCR			
	opt	ig	time	nodes	fg	ig	time	nodes	fg
$EIQP_2$_20_1	-9321876	17.84	-	7751701	1.76	3.10	3	5799	0
$EIQP_2$_20_2	-9013418	23.11	-	8366287	4.73	3.89	2	3523	0
$EIQP_2$_20_3	-15337225	12.53	210	335313	0	1.34	0	360	0
$EIQP_2$_20_4	-11863777	17.75	-	8436182	3.48	2.67	4	8763	0
$EIQP_2$_20_5	-12095004	21.28	-	8896130	6.53	2.95	3	4509	0
moyenne		**18.50**			**3.31**	**2.79**			**0**
$EIQP_2$_30_1	-23592535	20.18	-	5832801	11.61	3.06	6	6780	0
$EIQP_2$_30_2	-25924713	15.05	-	5700091	5.98	0.65	12	13018	0
$EIQP_2$_30_3	-21938906	22.81	-	5814721	13.09	2.32	6	7449	0
$EIQP_2$_30_4	-29913305	16.57	-	5754501	7.17	0.78	7	7925	0
$EIQP_2$_30_5	-22422891	24.44	-	5968416	14.73	3.20	6	5965	0
moyenne		**19.81**			**10.52**	**2.00**			**0**
$EIQP_2$_40_1	-42548497	16.52	-	4474098	10.40	1.18	7	6235	0
$EIQP_2$_40_2	-35957464	24.16	-	4453801	17.04	1.45	17	13869	0
$EIQP_2$_40_3	-40116963	17.00	-	4526300	10.50	1.17	20	16122	0
$EIQP_2$_40_4	-51306080	11.22	-	4242061	4.98	0.54	1	903	0
$EIQP_2$_40_5	-38090192	19.31	-	4549921	13.27	2.53	18	15388	0
moyenne		**17.64**			**10.64**	**1.37**			**0**

- : Cplex a été arrêté avec un gap final positif après 1 heure de résolution

TAB. B.8 – Résolution de ($EIQP_2$) avec les méthodes `IQCR` et le Branch and Bound

		IQCR				Branch and Bound			
	opt	ig	time	nodes	fg	ig	time	nodes	fg
$EIQP_2$_20_1	-9321876	0.14	140	3282	0	0.14	54	34	0
$EIQP_2$_20_2	-9013418	0	4	14	0	0	1	1	0
$EIQP_2$_20_3	-15337225	0.03	42	926	0	0.03	1	3	0
$EIQP_2$_20_4	-11863777	0.19	90	1339	0	0.19	92	168	0
$EIQP_2$_20_5	-12095004	0.08	14	252	0	0.08	7	10	0
moyenne		**0.09**			**0**	**0.09**			**0**
$EIQP_2$_30_1	-23592535	0.29	135	1073	0	0.29	-	3192	0.4
$EIQP_2$_30_2	-25924713	0.11	431	3345	0	0.11	83	44	0
$EIQP_2$_30_3	-21938906	0.02	331	2416	0	0.02	48	17	0
$EIQP_2$_30_4	-29913305	0.05	506	2653	0	0.05	85	26	0
$EIQP_2$_30_5	-22422891	0.12	343	2482	0	0.12	2365	1221	0
moyenne		**0.12**			**0**	**0.12**			**0.08**
$EIQP_2$_40_1	-42548497	0.02	1162	1315	0	0.02	157	46	0
$EIQP_2$_40_2	-35957464	0.17	-	4236	0.03	0.17	-	509	0
$EIQP_2$_40_3	-40116963	0.05	-	2496	0.03	0.05	209	35	0
$EIQP_2$_40_4	-51306080	0	49	3	0	0	3	2	0
$EIQP_2$_40_5	-38090192	0.01	2309	3528	0	0.01	1789	687	0
moyenne		**0.05**			**0.01**	**0.05**			**0**

- : Cplex a été arrêté avec un gap final positif après 1 heure de résolution

TAB. B.9 – Résolution de ($EIQP_3$) par les méthodes `BBL` et `BBLr`

		BBL				BBLr			
	opt	ig	time	nodes	fg	ig	time	nodes	fg
$EIQP_3$_20_1	-13226046	147.03	-	5489	77.06	111.46	-	768	50.98
$EIQP_3$_20_2	-16400092	140.70	-	8891	50.35	111.47	-	705	43.59
$EIQP_3$_20_3	-13372984	166.38	-	8498	48.47	138.66	-	684	100.56
$EIQP_3$_20_4	-9904855	236.75	-	4999	103.88	202.93	-	781	130.37
$EIQP_3$_20_5	-10903367	166.66	-	8020	99.77	133.52	-	641	98.84
moyenne		**171.50**			**75.90**	**139.61**			**84.87**
$EIQP_3$_30_1	-24412436	163.50	-	539	134.29	100.57	-	105	94.73
$EIQP_3$_30_2	-25640775	189.63	-	1135	166.92	168.37	-	82	161.71
$EIQP_3$_30_3	-23342586	186.92	-	479	151.85	123.18	-	60	113.95
$EIQP_3$_30_4	-29843855	182.01	-	400	116.75	156.61	-	113	146.79
$EIQP_3$_30_5	-26911633	175.51	-	772	152.52	129.52	-	51	122.66
moyenne		**179.51**			**144.47**	**135.65**			**127.97**
$EIQP_3$_40_1	-50352748	159.99	-	75	149.79	116.96	-	0	111.77
$EIQP_3$_40_2	-46862608	170.84	-	91	151.48	138.40	-	0	138.40
$EIQP_3$_40_3	-51680153	168.54	-	336	154.66	126.35	-	0	121.61
$EIQP_3$_40_4	-49068049	188.24	-	581	165.07	165.59	-	0	159.81
$EIQP_3$_40_5	-36454613	222.91	-	164	194.90	169.15	-	0	161.40
moyenne		**182.10**			**163.18**	**143.29**			**138.60**

- : Cplex a été arrêté avec un gap final positif après 1 heure de résolution

TAB. B.10 – Résolution de ($EIQP_3$) par les méthodes `BIL` et `BILr`

		BIL				BILr			
	opt	ig	time	nodes	fg	ig	time	nodes	fg
$EIQP_3$_20_1	-13226046	41.29	601	10169	0	12.62	485	756	0
$EIQP_3$_20_2	-16400092	36.95	289	5367	0	10.58	78	111	0
$EIQP_3$_20_3	-13372984	47.25	471	10574	0	27.17	827	1302	0
$EIQP_3$_20_4	-9904855	93.45	-	47717	14.78	67.37	-	8583	14.59
$EIQP_3$_20_5	-10903367	52.45	2045	34007	0	35.06	1137	2470	0
moyenne		**54.28**			**2.96**	**30.56**			**2.92**
$EIQP_3$_30_1	-24412436	48.26	-	11895	35.36	23.72	-	942	18.01
$EIQP_3$_30_2	-25640775	57.97	-	11353	14.93	37.99	-	906	34.98
$EIQP_3$_30_3	-23342586	62.36	-	12748	33.12	34.23	-	990	29.69
$EIQP_3$_30_4	-29843855	55.21	-	10690	17.60	30.40	-	861	21.93
$EIQP_3$_30_5	-26911633	56.20	-	11825	34.70	29.64	-	804	23.36
moyenne		**56.00**			**26.13**	**31.20**			**25.59**
$EIQP_3$_40_1	-50352748	46.09	-	3289	32.54	23.80	-	151	23.14
$EIQP_3$_40_2	-46862608	51.64	-	2707	47.61	35.55	-	282	35.43
$EIQP_3$_40_3	-51680153	48.24	-	2982	38.13	27.75	-	157	27.60
$EIQP_3$_40_4	-49068049	54.64	-	3089	36.90	34.73	-	252	30.69
$EIQP_3$_40_5	-36454613	83.51	-	2722	69.22	59.35	-	162	58.69
moyenne		**56.82**			**44.88**	**36.24**			**35.11**

- : Cplex a été arrêté avec un gap final positif après 1 heure de résolution

TAB. B.11 – Résolution de ($EIQP_3$) avec les méthodes `NC` et `CQCR`

		NC				CQCR			
	opt	ig	time	nodes	fg	ig	time	nodes	fg
$EIQP_3$_20_1	-13226046	19.27	1283	1459042	0	2.93	0	1078	0
$EIQP_3$_20_2	-16400092	18.04	-	5950448	2.21	4.31	1	1372	0
$EIQP_3$_20_3	-13372984	13.50	199	269074	0	5.01	3	4601	0
$EIQP_3$_20_4	-9904855	24.70	-	6577410	8.04	11.27	7	11918	0
$EIQP_3$_20_5	-10903367	22.54	-	5938750	3.56	7.49	1	1364	0
moyenne		**19.61**			**2.81**	**6.20**			**0**
$EIQP_3$_30_1	-24412436	18.01	-	4471796	7.01	6.85	27	26701	0
$EIQP_3$_30_2	-25640775	19.48	-	4580736	10.27	7.47	27	28315	0
$EIQP_3$_30_3	-23342586	22.13	-	4671160	10.35	3.60	9	9465	0
$EIQP_3$_30_4	-29843855	12.80	-	4814101	4.22	3.06	4	4363	0
$EIQP_3$_30_5	-26911633	28.57	-	4718071	17.05	6.76	725	701933	0
moyenne		**20.20**			**9.78**	**5.55**			**0**
$EIQP_3$_40_1	-50352748	20.23	-	3503721	12.42	2.09	14	7722	0
$EIQP_3$_40_2	-46862608	18.07	-	3473701	11.48	2.73	10	6635	0
$EIQP_3$_40_3	-51680153	20.47	-	3529911	13.33	2.30	12	10054	0
$EIQP_3$_40_4	-49068049	18.17	-	3539901	11.43	6.08	1797	1116467	0
$EIQP_3$_40_5	-36454613	24.35	-	3506691	16.77	7.98	1688	1183003	0
moyenne		**20.26**			**13.09**	**4.24**			**0**

- : Cplex a été arrêté avec un gap final positif après 1 heure de résolution

TAB. B.12 – Résolution de ($EIQP_3$) avec les méthodes `IQCR` et le Branch and Bound

		IQCR				Branch and Bound			
	opt	ig	time	nodes	fg	ig	time	nodes	fg
$EIQP_3$_20_1	-13226046	0.06	6	42	0	0.06	4	6	0
$EIQP_3$_20_2	-16400092	0.03	5	11	0	0.03	1	2	0
$EIQP_3$_20_3	-13372984	0.16	33	512	0	0.16	25	119	0
$EIQP_3$_20_4	-9904855	1.60	156	3897	0	1.60	-	36533	2.47
$EIQP_3$_20_5	-10903367	0.06	26	344	0	0.06	5	8	0
moyenne		**0.38**			**0**	**0.38**			**0.5**
$EIQP_3$_30_1	-24412436	0.02	85	335	0	0.02	8	8	0
$EIQP_3$_30_2	-25640775	0.27	474	3078	0	0.27	-	3564	0.1
$EIQP_3$_30_3	-23342586	0.05	1184	2184	0	0.05	394	285	0
$EIQP_3$_30_4	-29843855	0.03	76	229	0	0.03	92	130	0
$EIQP_3$_30_5	-26911633	0.76	634	4647	0	0.76	-	9384	1.4
moyenne		**0.23**			**0**	**0.23**			**0.3**
$EIQP_3$_40_1	-50352748	0.01	865	928	0	0.01	92	67	0
$EIQP_3$_40_2	-46862608	0	320	158	0	0	16	3	0
$EIQP_3$_40_3	-51680153	0.04	2049	2876	0	0.04	121	42	0
$EIQP_3$_40_4	-49068049	0.03	3126	5524	0	0.03	1393	557	0
$EIQP_3$_40_5	-36454613	0.48	-	6141	0.13	0.48	-	2507	1.3
moyenne		**0.11**			**0.03**	**0.11**			**0.26**

- : Cplex a été arrêté avec un gap final positif après 1 heure de résolution

B.2 Résultats pour la classe de problèmes ($IIQP$)

TAB. B.13 – Résolution de ($IIQP_1$) par les méthodes `BBL` et `BBLr`

		BBL				BBLr			
	opt	ig	time (s)	nodes	fg	ig	time (s)	nodes	fg
$IIQP_1$_20_1	-4763626	25.39	-	26242	0.94	12.62	404	547	0
$IIQP_1$_20_2	-3693618	40.04	2290	10092	0	32.05	1303	1241	0
$IIQP_1$_20_3	-3783803	39.06	2552	9152	0	25.47	405	604	0
$IIQP_1$_20_4	-5624785	7.74	192	1177	0	3.26	47	0	0
$IIQP_1$_20_5	-2749979	51.05	-	14397	4.29	41.40	1083	1277	0
moyenne		**32.65**			**1.05**	**22.96**			**0**
$IIQP_1$_30_1	-9104372	32.07	-	1991	5.90	25.03	-	556	8.56
$IIQP_1$_30_2	-8632815	50.43	-	679	21.27	36.91	-	479	29.99
$IIQP_1$_30_3	-7738098	61.38	-	512	23.67	52.09	-	372	45.94
$IIQP_1$_30_4	-6070518	81.19	-	545	42.57	70.48	-	645	54.63
$IIQP_1$_30_5	-10514171	22.89	-	5790	7.25	13.29	-	816	7.75
moyenne		**49.59**			**20.13**	**39.56**			**29.37**
$IIQP_1$_40_1	-14186867	61.48	-	491	47.45	52.73	-	38	52.71
$IIQP_1$_40_2	-15447872	47.69	-	481	37.71	36.42	-	79	34.66
$IIQP_1$_40_3	-11635872	78.51	-	0	65.13	69.18	-	63	64.73
$IIQP_1$_40_4	-13820601	57.76	-	482	46.10	49.87	-	25	49.87
$IIQP_1$_40_5	-12820428	65.65	-	324	44.60	57.32	-	110	53.71
moyenne		**62.22**			**48.20**	**53.10**			**51.14**

- : Cplex a été arrêté avec un gap final positif après 1 heure de résolution

TAB. B.14 – Résolution de ($IIQP_1$) par les méthodes `BIL` et `BILr`

		BIL				BILr			
	opt	ig	time (s)	nodes	fg	ig	time (s)	nodes	fg
$IIQP_1$_20_1	-4763626	20.94	81	1699	0	8.20	77	229	0
$IIQP_1$_20_2	-3693618	35.30	112	1399	0	27.36	129	225	0
$IIQP_1$_20_3	-3783803	33.87	128	2495	0	20.21	111	243	0
$IIQP_1$_20_4	-5624785	3.71	7	422	0	0.02	7	0	0
$IIQP_1$_20_5	-2749979	45.10	117	1975	0	34.08	337	743	0
moyenne		**27.78**			**0**	**17.97**			**0**
$IIQP_1$_30_1	-9104372	27.19	790	4296	0	19.75	2025	782	5.51
$IIQP_1$_30_2	-8632815	44.97	-	10920	14.46	31.47	-	1148	16.76
$IIQP_1$_30_3	-7738098	56.17	-	8204	39.72	47.32	-	1168	0
$IIQP_1$_30_4	-6070518	74.09	-	13058	34.61	61.90	-	2061	0
$IIQP_1$_30_5	-10514171	18.68	785	11756	0	8.71	1330	719	0.04
moyenne		**44.22**			**17.76**	**33.83**			**4.46**
$IIQP_1$_40_1	-14186867	56.16	-	2860	48.53	47.33	-	385	40.15
$IIQP_1$_40_2	-15447872	42.65	-	3112	38.68	31.27	-	419	27.00
$IIQP_1$_40_3	-11635872	72.45	-	3688	54.42	63.34	-	456	59.40
$IIQP_1$_40_4	-13820601	52.67	-	3270	39.39	44.88	-	360	40.94
$IIQP_1$_40_5	-12820428	59.77	-	3853	51.45	51.08	-	470	46.83
moyenne		**56.74**			**46.49**	**47.58**			**42.86**

- : Cplex a été arrêté avec un gap final positif après 1 heure de résolution

TAB. B.15 – Résolution de ($IIQP_1$) par les méthodes `NC` et `IQCR`

		NC				IQCR			
	opt	ig	time (s)	nodes	fg	ig	time (s)	nodes	fg
$IIQP_1$_20_1	-4763626	15.28	-	7414712	0.03	1.07	21	342	0
$IIQP_1$_20_2	-3693618	8.22	3	8009	0	0.09	10	69	0
$IIQP_1$_20_3	-3783803	15.10	1557	2560447	0	0.83	11	144	0
$IIQP_1$_20_4	-5624785	7.42	3	8105	0	0.24	4	17	0
$IIQP_1$_20_5	-2749979	10.61	11	32780	0	1.09	11	206	0
moyenne		**11.33**			**0**	**0.66**			**0**
$IIQP_1$_30_1	-9104372	4.66	2	-	4.66	0.12	32	13	0
$IIQP_1$_30_2	-8632815	12.41	-	7602932	2.69	0.95	174	1192	0
$IIQP_1$_30_3	-7738098	11.73	-	7531701	1.53	0.72	179	996	0
$IIQP_1$_30_4	-6070518	13.53	-	7558301	3.48	1.11	156	867	0
$IIQP_1$_30_5	-10514171	11.94	-	7861657	2.75	0.05	27	30	0
moyenne		**10.85**			**3.02**	**0.59**			**0**
$IIQP_1$_40_1	-14186867	11.43	-	6017301	5.10	0.33	798	1259	0
$IIQP_1$_40_2	-15447872	10.27	-	5456401	3.76	1.55	1015	2886	0
$IIQP_1$_40_3	-11635872	11.79	-	5670701	5.29	0.87	754	2447	0
$IIQP_1$_40_4	-13820601	11.34	-	5757501	4.78	0.81	389	934	0
$IIQP_1$_40_5	-12820428	11.49	-	5867501	5.43	0.95	799	2810	0
moyenne		**11.27**			**4.87**	**0.90**			**0**

- : Cplex a été arrêté avec un gap final positif après 1 heure de résolution

TAB. B.16 – Résolution de ($IIQP_1$) par les méthodes `IQCRs` et `CQCR`

		IQCRs				CQCR			
	opt	ig	time (s)	nodes	fg	ig	time (s)	nodes	fg
$IIQP_1$_20_1	-4763626	0.48	15	246	0	6.64	1	2564	0
$IIQP_1$_20_2	-3693618	0.06	5	28	0	3.88	0	404	0
$IIQP_1$_20_3	-3783803	0.08	4	20	0	4.49	0	386	0
$IIQP_1$_20_4	-5624785	0.02	2	0	0	2.69	0	762	0
$IIQP_1$_20_5	-2749979	0.12	6	54	0	3.55	0	1658	0
moyenne		**0.15**			**0**	**4.25**			**0**
$IIQP_1$_30_1	-9104372	0.09	22	22	0	0.44	0	96	0
$IIQP_1$_30_2	-8632815	0.05	75	294	0	4.09	1	1407	0
$IIQP_1$_30_3	-7738098	0.01	69	26	0	5.78	1	1363	0
$IIQP_1$_30_4	-6070518	0.27	65	301	0	5.09	1	1362	0
$IIQP_1$_30_5	-10514171	0.05	23	17	0	0.95	2	3107	0
moyenne		**0.09**			**0**	**3.27**			**0**
$IIQP_1$_40_1	-14186867	0.06	296	296	0	3.50	5	4158	0
$IIQP_1$_40_2	-15447872	0.03	407	602	0	6.40	20	19816	0
$IIQP_1$_40_3	-11635872	0.39	607	1041	0	4.68	10	8510	0
$IIQP_1$_40_4	-13820601	0.02	341	472	0	4.69	8	7140	0
$IIQP_1$_40_5	-12820428	0.13	702	2010	0	5.25	25	23667	0
moyenne		**0.13**			**0**	**4.90**			**0**

- : Cplex a été arrêté avec un gap final positif après 1 heure de résolution

TAB. B.17 – Résolution de ($IIQP_2$) par les méthodes `BBL` et `BBLr`

		BBL				BBLr			
	opt	ig	time	nodes	fg	ig	time	nodes	fg
$IIQP_2$_20_1	-11368326	47.36	-	4991	10.87	31.58	-	1547	16.07
$IIQP_2$_20_2	-8187686	90.61	-	6010	50.06	53.20	-	1026	39.21
$IIQP_2$_20_3	-8318158	71.70	-	7813	41.33	47.79	-	990	32.07
$IIQP_2$_20_4	-8362534	96.34	-	4462	57.11	51.86	-	1252	29.75
$IIQP_2$_20_5	-11277208	49.38	-	9409	12.95	30.65	-	1425	16.58
moyenne		**71.08**			**34.46**	**43.01**			**26.74**
$IIQP_2$_30_1	-28283032	42.46	-	2617	33.05	34.79	-	240	28.40
$IIQP_2$_30_2	-24952930	51.29	-	904	37.94	32.47	-	253	31.70
$IIQP_2$_30_3	-20248934	79.20	-	499	64.37	57.15	-	208	55.65
$IIQP_2$_30_4	-24404197	53.73	-	2677	38.55	33.47	-	223	33.30
$IIQP_2$_30_5	-26490741	56.50	-	1799	41.15	32.40	-	324	30.70
moyenne		**56.64**			**43.01**	**38.06**			**35.95**
$IIQP_2$_40_1	-45892717	54.08	-	503	45.49	40.71	-	7	40.70
$IIQP_2$_40_2	-42436784	63.42	-	210	51.04	45.30	-	7	45.30
$IIQP_2$_40_3	-37091366	77.09	-	130	65.03	56.72	-	0	56.72
$IIQP_2$_40_4	-41852435	62.08	-	100	54.69	42.71	-	7	42.47
$IIQP_2$_40_5	-40834676	62.15	-	407	52.43	45.89	-	3	45.88
moyenne		**63.76**			**53.74**	**46.27**			**46.21**

- : Cplex a été arrêté avec un gap final positif après 1 heure de résolution

TAB. B.18 – Résolution de ($IIQP_2$) par les méthodes `BIL` et `BILr`

		BIL				BILr			
	opt	ig	time	nodes	fg	ig	time	nodes	fg
$IIQP_2$_20_1	-11368326	23.13	291	9990	0	4.02	86	157	0
$IIQP_2$_20_2	-8187686	60.91	-	62958	17.58	31.12	1209	1943	0
$IIQP_2$_20_3	-8318158	46.44	-	58434	8.67	20.57	207	323	0
$IIQP_2$_20_4	-8362534	64.88	-	57140	24.55	29.34	1822	3860	0
$IIQP_2$_20_5	-11277208	24.51	243	6778	0	3.35	69	108	0
moyenne		**43.97**			**10.17**	**17.68**			**0**
$IIQP_2$_30_1	-28283032	19.01	-	19047	7.77	1.01	901	387	0
$IIQP_2$_30_2	-24952930	30.48	-	15057	14.30	6.41	-	1086	0.05
$IIQP_2$_30_3	-20248934	51.80	-	13259	33.81	26.51	-	762	17.18
$IIQP_2$_30_4	-24404197	29.50	-	16635	8.41	6.78	2889	710	0
$IIQP_2$_30_5	-26490741	31.06	-	13867	19.67	4.58	1410	447	0
moyenne		**32.37**			**16.79**	**9.06**			**3.45**
$IIQP_2$_40_1	-45892717	28.61	-	5907	23.21	8.70	-	181	6.70
$IIQP_2$_40_2	-42436784	38.39	-	5592	31.30	14.52	-	170	12.76
$IIQP_2$_40_3	-37091366	50.03	-	5125	47.89	27.96	-	185	27.75
$IIQP_2$_40_4	-41852435	38.18	-	5929	31.30	15.02	-	162	13.84
$IIQP_2$_40_5	-40834676	39.65	-	5101	34.07	16.44	-	160	14.64
moyenne		**38.97**			**33.56**	**16.53**			**15.14**

- : Cplex a été arrêté avec un gap final positif après 1 heure de résolution

TAB. B.19 – Résolution de ($IIQP_2$) par les méthodes `NC` et `IQCR`

		NC				IQCR			
	opt	ig	time (s)	nodes	fg	ig	time (s)	nodes	fg
$IIQP_2$_20_1	-11368326	9.62	18	34880	0	0.13	11	100	0
$IIQP_2$_20_2	-8187686	24.73	-	8116218	7.94	8.54	800	33547	0
$IIQP_2$_20_3	-8318158	19.69	-	7916261	3.58	4.49	36	286	0
$IIQP_2$_20_4	-8362534	34.17	-	8178097	16.47	17.80	184	3145	0
$IIQP_2$_20_5	-11277208	10.07	31	58193	0	0.03	7	0	0
moyenne		**19.66**			**5.60**	**6.20**			**0**
$IIQP_2$_30_1	-28283032	12.21	-	5727121	4.67	0.77	176	450	0
$IIQP_2$_30_2	-24952930	20.73	-	6101801	11.61	2.94	2336	18480	0
$IIQP_2$_30_3	-20248934	22.67	-	6125254	12.86	8.52	1268	11495	0
$IIQP_2$_30_4	-24404197	15.14	-	5436644	6.48	2.51	474	2493	0
$IIQP_2$_30_5	-26490741	19.66	-	6073422	10.55	2.68	494	2309	0
moyenne		**18.08**			**9.23**	**3.48**			**0**
$IIQP_2$_40_1	-45892717	14.43	-	4770061	8.25	0.62	676	787	0
$IIQP_2$_40_2	-42436784	20.20	-	5285301	13.52	1.22	3132	2610	0
$IIQP_2$_40_3	-37091366	24.38	-	5276161	17.49	7.12	-	3259	2.31
$IIQP_2$_40_4	-41852435	20.86	-	5176123	14.35	3.83	-	4258	0.94
$IIQP_2$_40_5	-40834676	18.22	-	5178724	11.92	2.16	2774	5283	0
moyenne		**19.62**			**13.11**	**2.99**			**0.92**

- : Cplex a été arrêté avec un gap final positif après 1 heure de résolution

TAB. B.20 – Résolution de ($IIQP_2$) avec les méthodes `IQCRs` et `CQCR`

		IQCRs				CQCR			
	opt	ig	time (s)	nodes	fg	ig	time (s)	nodes	fg
$IIQP_2$_20_1	-11368326	0.02	5	31	0	4.36	1	1315	0
$IIQP_2$_20_2	-8187686	0.14	23	264	0	18.25	2	3062	0
$IIQP_2$_20_3	-8318158	0.02	12	111	0	11.54	0	668	0
$IIQP_2$_20_4	-8362534	0.37	17	189	0	26.86	7	13753	0
$IIQP_2$_20_5	-11277208	0	3	3	0	5.24	0	260	0
moyenne		**0.09**			**0**	**13.25**			**0**
$IIQP_2$_30_1	-28283032	0.05	61	238	0	6.52	9	9277	0
$IIQP_2$_30_2	-24952930	0.08	728	3527	0	15.80	1762	1925355	0
$IIQP_2$_30_3	-20248934	0.04	498	2142	0	19.21	79	89579	0
$IIQP_2$_30_4	-24404197	0.03	43	113	0	11.49	31	33521	0
$IIQP_2$_30_5	-26490741	0.08	63	379	0	14.32	104	132591	0
moyenne		**0.02**			**0**	**13.47**			**0**
$IIQP_2$_40_1	-45892717	0.01	161	81	0	9.55	984	827741	0
$IIQP_2$_40_2	-42436784	0	130	100	0	15.26	-	3690179	1.05
$IIQP_2$_40_3	-37091366	0.03	354	642	0	19.87	-	3435981	2.07
$IIQP_2$_40_4	-41852435	0.04	460	490	0	16.87	-	3712671	1.09
$IIQP_2$_40_5	-40834676	0.01	544	975	0	13.97	-	3190516	0.41
moyenne		**0.02**			**0**	**15.10**			**0.92**

- : Cplex a été arrêté avec un gap final positif après 1 heure de résolution

TAB. B.21 – Résolution de ($IIQP_3$) par les méthodes `BBL` et `BBLr`

		BBL				BBLr			
	opt	ig	time	nodes	fg	ig	time	nodes	fg
$IIQP_3$_20_1	-17240565	116.08	-	4450	79.05	71.91	3312	1029	0
$IIQP_3$_20_2	-10712606	174.44	-	6299	96.78	119.21	-	836	111.87
$IIQP_3$_20_3	-14193757	152.52	-	8856	80.36	121.20	-	808	62.61
$IIQP_3$_20_4	-16358540	125.65	-	5172	47.52	89.76	-	781	21.73
$IIQP_3$_20_5	-9979912	209.94	-	8082	126.41	115.81	-	776	43.41
moyenne		**155.73**			**86.02**	**103.58**			**47.93**
$IIQP_3$_30_1	-27168517	177.92	-	1048	136.04	156.27	-	37	129.49
$IIQP_3$_30_2	-25311234	158.91	-	301	129.49	110.81	-	41	103.54
$IIQP_3$_30_3	-20968526	203.91	-	555	166.52	141.47	-	38	133.09
$IIQP_3$_30_4	-22015154	184.00	-	520	148.21	147.58	-	41	116.64
$IIQP_3$_30_5	-26905764	132.63	-	616	100.90	100.96	-	48	94.73
moyenne		**171.47**			**136.23**	**131.42**			**115.50**
$IIQP_3$_40_1	-45889156	176.29	-	312	163.74	134.54	-	0	129.08
$IIQP_3$_40_2	-51750130	178.34	-	469	160.04	149.02	-	0	145.08
$IIQP_3$_40_3	-46860430	184.81	-	302	166.66	154.84	-	0	148.96
$IIQP_3$_40_4	-49474826	178.02	-	425	152.05	152.60	-	0	152.60
$IIQP_3$_40_5	-44447580	172.79	-	500	154.10	145.27	-	0	140.57
moyenne		**178.05**			**159.32**	**147.25**			**143.26**

- : Cplex a été arrêté avec un gap final positif après 1 heure de résolution

TAB. B.22 – Résolution de ($IIQP_3$) par les méthodes `BIL` et `BILr`

		BIL				BILr			
	opt	ig	time	nodes	fg	ig	time	nodes	fg
$IIQP_3$_20_1	-17240565	20.94	113	2071	0	1.03	292	616	0
$IIQP_3$_20_2	-10712606	59.84	2672	47442	0	30.54	1326	2312	0
$IIQP_3$_20_3	-14193757	40.01	236	3821	0	17.23	297	508	0
$IIQP_3$_20_4	-16358540	22.67	88	2199	0	2.79	168	234	0
$IIQP_3$_20_5	-9979912	79.81	-	45019	8.80	36.11	975	1513	0
moyenne		**44.65**			**1.76**	**17.54**			**0**
$IIQP_3$_30_1	-27168517	52.12	-	10324	17.76	32.30	-	828	22.04
$IIQP_3$_30_2	-25311234	48.14	-	9823	21.79	25.44	-	690	17.49
$IIQP_3$_30_3	-20968526	70.61	-	13099	42.37	47.86	-	742	41.43
$IIQP_3$_30_4	-22015154	63.50	-	9793	30.84	40.80	-	711	28.14
$IIQP_3$_30_5	-26905764	30.96	-	12206	4.40	17.92	-	774	11.47
moyenne		**53.07**			**23.43**	**32.86**			**24.11**
$IIQP_3$_40_1	-45889156	56.49	-	3101	42.76	37.19	-	142	36.54
$IIQP_3$_40_2	-51750130	51.61	-	3073	37.26	34.33	-	186	33.91
$IIQP_3$_40_3	-46860430	54.64	-	3084	41.85	38.09	-	124	37.89
$IIQP_3$_40_4	-49474826	51.78	-	3109	43.62	35.32	-	120	34.86
$IIQP_3$_40_5	-44447580	51.19	-	3150	38.69	35.90	-	113	35.66
moyenne		**53.14**			**40.84**	**36.17**			**35.77**

- : Cplex a été arrêté avec un gap final positif après 1 heure de résolution

[ht !]

TAB. B.23 – Résolution de ($IIQP_3$) avec les méthodes `NC` et `IQCR`

		NC				IQCR			
	opt	ig	time	nodes	fg	ig	time	nodes	fg
$IIQP_3$_20_1	-17240565	10.42	8	11544	0	0.07	23	209	0
$IIQP_3$_20_2	-10712606	29.91	-	6651951	12.87	8.46	965	44154	0
$IIQP_3$_20_3	-14193757	20.28	-	5157759	0.75	2.12	77	2354	0
$IIQP_3$_20_4	-16358540	12.80	145	205174	0	0.47	17	199	0
$IIQP_3$_20_5	-9979912	29.30	-	6554901	9.05	9.17	329	6574	0
moyenne		**20.54**			**4.53**	**4.06**			**0**
$IIQP_3$_30_1	-27168517	14.29	-	4474449	4.62	0.12	164	361	0
$IIQP_3$_30_2	-25311234	18.44	-	4641510	8.51	0.54	1364	5010	0
$IIQP_3$_30_3	-20968526	25.99	-	5022601	14.67	3.37	3568	36437	0
$IIQP_3$_30_4	-22015154	22.01	-	4774014	11.35	1.90	893	5162	0
$IIQP_3$_30_5	-26905764	15.74	-	4528910	5.18	0.99	290	884	0
moyenne		**19.29**			**8.87**	**1.38**			**0**
$IIQP_3$_40_1	-45889156	21.60	-	3942200	14.21	0.37	3439	2457	0
$IIQP_3$_40_2	-51750130	21.40	-	3958031	14.01	0.84	-	3300	0.53
$IIQP_3$_40_3	-46860430	20.26	-	3931901	12.95	2.09	-	2275	1.92
$IIQP_3$_40_4	-49474826	19.84	-	4015801	13.20	1.09	-	1800	1.07
$IIQP_3$_40_5	-44447580	20.66	-	4106841	13.91	1.50	-	3147	1.25
moyenne		**20.75**			**13.66**	**1.18**			**0.95**

- : Cplex a été arrêté avec un gap final positif après 1 heure de résolution

TAB. B.24 – Résolution de ($IIQP_3$) avec les méthodes `IQCRs` et `CQCR`

		IQCRs				CQCR			
	opt	ig	time	nodes	fg	ig	time	nodes	fg
$IIQP_3$_20_1	-17240565	0.03	23.74	12008	0	7.23	10	12669	0
$IIQP_3$_20_2	-10712606	0.03	44.75	640	0	19.02	17	29539	0
$IIQP_3$_20_3	-14193757	0.08	15.40	168	0	12.98	4.44	5964	0
$IIQP_3$_20_4	-16358540	0.02	16.45	1878	0	5.47	2	2829	0
$IIQP_3$_20_5	-9979912	0.07	8.22	29	0	20.33	4	8081	0
moyenne		**0.05**			**0**	**13.01**			**0**
$IIQP_3$_30_1	-27168517	0.04	43.12	45	0	7.09	12	10930	0
$IIQP_3$_30_2	-25311234	0	38.52	3	0	9.20	551	3719047	0
$IIQP_3$_30_3	-20968526	0.11	191.37	800	0	16.93	202	227475	0
$IIQP_3$_30_4	-22015154	0	56.26	12	0	12.86	22	21768	0
$IIQP_3$_30_5	-26905764	0.02	55.86	156	0	8.53	81	82644	0
moyenne		**0.11**			**0**	**10.92**			**0**
$IIQP_3$_40_1	-45889156	0.19	703.60	447	0	10.68	1476	1081979	0
$IIQP_3$_40_2	-51750130	0.01	547.67	670	0	11.17	-	2922753	0.49
$IIQP_3$_40_3	-46860430	0.01	844.64	665	0	11.81	-	2927102	0.77
$IIQP_3$_40_4	-49474826	0.13	726.79	574	0	12.39	-	3140976	1.64
$IIQP_3$_40_5	-44447580	0.22	743.87	748	0	11.01	-	3064654	0.81
moyenne		**0.04**			**0**	**11.41**			**0.74**

- : Cplex a été arrêté avec un gap final positif après 1 heure de résolution

B.3 Résultats pour les instances de Pardalos and Rodgers

TAB. B.25 – Résultats pour les instances de Pardalos and Rodgers

		IQCR			UQCR		
	opt	ig	time (s)	nodes	ig	time (s)	nodes
PARD_50_d40	-5239	0.00	0.3	0	2.0	0.1	34
PARD_50_d40	-3739	1.01	3.3	169	8.2	0.3	1042
PARD_50_d40	-4504	0.32	1.8	53	6.5	0.2	627
PARD_50_d40	-5570	0.02	0.8	2	2.9	0.1	150
PARD_50_d40	-3909	0.10	1.3	12	4.7	0.1	386
PARD_50_d40	-3569	1.04	2.4	84	8.2	0.2	820
PARD_50_d40	-3753	0.19	1.3	25	6.1	0.1	421
PARD_50_d40	-5948	0.03	1.2	1	3.6	0.1	375
PARD_50_d40	-2904	1.84	3.7	180	11.3	0.5	1933
PARD_50_d40	-4212	1.02	3.2	168	8.0	0.8	3310
moyenne		**0.56**	**1.4**	**69**	**6.1**	**0.3**	**910**
PARD_50_d60	-5758	0.25	1.8	57	5.3	0.2	547
PARD_50_d60	-4500	0.43	1.8	56	6.2	0.2	411
PARD_50_d60	-5517	0.08	1.1	2	5.5	0.1	210
PARD_50_d60	-4832	2.01	4.2	237	8.7	0.8	2826
PARD_50_d60	-5285	0.04	1.1	3	4.3	0.1	189
PARD_50_d60	-5028	0.08	1.2	15	4.7	0.1	204
PARD_50_d60	-6270	0.00	0.2	0	1.1	0	18
PARD_50_d60	-6449	0.01	0.3	0	3.4	0.1	113
PARD_50_d60	-3589	1.38	2.6	118	11.2	0.5	1911
PARD_50_d60	-5335	1.30	4.8	295	7.4	1	3911
moyenne		**0.56**	**1.4**	**78**	**5.8**	**0.3**	**1034**
PARD_50_d100	-6371	1.05	2.8	114	7.8	0.4	1067
PARD_50_d100	-5037	2.46	5.7	276	12.7	1	2861
PARD_50_d100	-7163	0.13	1	5	3.7	0.1	120
PARD_50_d100	-7344	0.05	1	0	3.7	0.1	92
PARD_50_d100	-5328	2.74	5.5	349	10.8	1.2	3704
PARD_50_d100	-7202	0.17	1.4	28	4.0	0.1	185
PARD_50_d100	-7165	0.02	1.1	0	4.8	0.1	223
PARD_50_d100	-8698	0.03	1	2	1.9	0.6	61 0.6
PARD_50_d100	-4788	3.15	7.6	624	12.9	1.7	5275
PARD_50_d100	-6448	0.07	1.2	4	3.5	0.1	84
moyenne		**0.99**	**2.3**	**140**	**6.6**	**0.5**	**1367**
PARD_60_d20	-3796	0.11	2.4	15	6.5	0.4	1337
PARD_60_d20	-3930	0.28	2.9	53	7.3	0.8	3159
PARD_60_d20	-3591	0.00	0.5	0	5.0	0.2	646
PARD_60_d20	-5464	0.01	0.5	0	2.3	0.1	123
PARD_60_d20	-4012	0.02	0.5	0	5.2	0.4	1428
PARD_60_d20	-3673	0.01	0.5	0	5.0	0.2	546
PARD_60_d20	-3227	0.08	2.5	20	6.4	0.6	2162
PARD_60_d20	-6600	0.00	0.4	0	0.8	0.4	25
PARD_60_d20	-2817	0.69	4.5	91	10.6	1.5	6524
PARD_60_d20	-3671	0.16	2.7	24	6.3	0.5	1730
moyenne		**0.14**	**1.5**	**20**	**5.5**	**0.5**	**1768**

- : Cplex a été arrêté avec un gap final positif après 1 heure de résolution

TAB. B.26 – Résultats pour les instances de Pardalos and Rodgers (suite)

		IQCR			UQCR		
	opt	ig	time (s)	nodes	ig	time (s)	nodes
PARD_60_d40	-7052	0.01	0.4	20	3.0	0.2	230
PARD_60_d40	-5047	1.10	6.5	196	8.1	1	2958
PARD_60_d40	-6101	0.13	2.9	27	4.9	0.3	560
PARD_60_d40	-5671	0.62	4.7	80	7.8	1.1	3446
PARD_60_d40	-5324	0.09	2.8	19	24.9	0.5	1141
PARD_60_d40	-5329	1.07	5.5	133	7.0	1.4	4484
PARD_60_d40	-4935	0.18	3.5	43	4.9	03	635
PARD_60_d40	-6555	0.12	3.4	46	5.2	0.5	1439
PARD_60_d40	-3337	4.90	33.7	1718	16.9	7.7	25867
PARD_60_d40	-5239	0.76	5.3	100	6.9	0.9	2899
moyenne		**0.90**	**6.5**	**238**	**9.0**	**1.4**	**4366**
PARD_70_d30	-6585	1.11	16.3	356	7.8	5.5	17913
PARD_70_d30	-5742	1.23	14.6	251	8.4	4	12528
PARD_70_d30	-7331	0.12	3.6	5	1.9	0.1	97
PARD_70_d30	-6743	0.37	6.9	96	5.9	1.61	5128
PARD_70_d30	-6393	0.44	7.3	126	5.9	1.5	5057
PARD_70_d30	-5701	2.40	40.8	1389	9.1	11	35975
PARD_70_d30	-6603	0.09	4.8	17	4.1	0.4	1125
PARD_70_d30	-7740	0.04	3.8	1	3.3	0.3	854
PARD_70_d30	-4396	1.00	10.6	170	8.8	1.7	5073
PARD_70_d30	-6130	0.37	7.8	91	5.7	1.7	4657
moyenne		**0.72**	**11.5**	**250**	**6.1**	**2.8**	**8841**
PARD_70_d80	-9792	0.64	9.6	162	6.4	2.4	5349
PARD_70_d80	-9111	0.48	8.0	121	6.8	1.5	2207
PARD_70_d80	-8692	1.70	19.8	418	7.9	7.1	14442
PARD_70_d80	-10733	1.50	26.3	844	7.6	8.1	19178
PARD_70_d80	-7843	0.67	8.4	110	5.7	1	1970
PARD_70_d80	-9102	1.48	18.2	418	6.7	3.4	7524
PARD_70_d80	-8706	2.26	35.9	1061	8.9	10.6	23672
PARD_70_d80	-11704	0.78	10.2	228	5.3	2.7	6129
PARD_70_d80	-7186	2.25	24.6	610	9.5	6.1	13184
PARD_70_d80	-9033	1.37	20.9	467	8.0	6.8	15337
moyenne		**1.31**	**17.5**	**443**	**7.3**	**5.4**	**10899**
PARD_80_d20	-6719	0.04	6.6	19	5.1	1.6	4376
PARD_80_d20	-6615	0.13	7	33	5.1	1.8	5342
PARD_80_d20	-6605	0.06	5.5	3	3.9	0.6	1672
PARD_80_d20	-6491	0.10	9.1	52	5.5	2.1	6144
PARD_80_d20	-6141	0.25	9.8	60	5.4	1.9	5537
PARD_80_d20	-6313	1.24	41.7	909	7.9	28.7	85797
PARD_80_d20	-6057	0.50	15.5	199	7.1	4.9	15014
PARD_80_d20	-7362	0.02	6.7	1	4.4	1.5	4562
PARD_80_d20	-4090	1.95	31.3	522	9.7	9.5	27537
PARD_80_d20	-6242	0.21	8.9	57	5.0	1.5	4478
moyenne		**0.45**	**13.5**	**185**	**5.9**	**5.4**	**16046**
PARD_100_d100	-19412	0.78	74	585	5.3	27	30676
PARD_100_d100	-17290	1.26	159	1501	6.3	59	61394
PARD_100_d100	-17565	0.76	64	458	6.6	43	46629
PARD_100_d100	-19125	1.14	131	1403	5.2	52	57035
PARD_100_d100	-15868	2.54	469	5188	9.0	330	392129
PARD_100_d100	-17368	1.28	128	1031	6.7	59	61584
PARD_100_d100	-18629	1.65	271	2858	7.9	405	399809
PARD_100_d100	-18649	2.75	1384	17901	9.0	1671	1664154
PARD_100_d100	-13294	3.96	1436	18025	10.8	574	546701
PARD_100_d100	-15352	2.50	640	6076	9.5	444	443466
moyenne		**1.86**	**475.6**	**5502**	**7.6**	**372.7**	**370358**

- : Cplex a été arrêté avec un gap final positif après 1 heure de résolution

TAB. B.27 – Résultats pour les instances de Pardalos and Rodgers (suite)

		IQCR			UQCR		
	opt	ig	time (s)	nodes	ig	time (s)	nodes
PARD_120_d30	-13067	2.19	2356	16382	8.3	3495	3881966
PARD_120_d30	-13046	1.01	298	1423	6.6	349	427563
PARD_120_d30	-12418	1.63	830	5097	6.6	838	1052495
PARD_120_d30	-13867	1.31	570	3371	6.4	727	903643
PARD_120_d30	-11403	1.44	408	2077	7.6	500	602976
PARD_120_d30	-12915	0.93	171	629	6.4	260	316448
PARD_120_d30	-14068	0.52	116	375	5.1	168	235217
PARD_120_d30	-14701	0.83	241	904	5.1	324	416262
PARD_120_d30	-10458	3.58	3820	23924	10.3	4467	4673214
PARD_120_d30	-12201	1.85	1046	6645	8.4	1308	1545289
moyenne		**1.53**	**985**	**6082**	**7.1**	**1263.6**	**1405507**

		IQCR				UQCR			
	opt	ig	time (s)	nodes	fg	ig	time (s)	nodes	fg
PARD_120_d80	-18691	3.80	-	63434	1.21	10.5	-	8millions	1.7
PARD_120_d80	-18827	2.85	5282	33057	0	9.1	9898	6358472	0
PARD_120_d80	-19302	2.61	2863	17891	0	8.5	4096	2990730	0
PARD_120_d80	-20765	1.52	601	2943	0	6.9	1147	896834	0
PARD_120_d80	-20417	1.39	467	2524	0	6.5	322	277408	0
PARD_120_d80	-18482	2.63	3116	21387	0	9.0	6696	4637045	0
PARD_120_d80	-22194	2.69	-	87540	0.18	9.0	-	8millions	1.2
PARD_120_d80	-19515	4.38	-	60641	1.97	10.8	-	8millions	2.2
PARD_120_d80	-18195	2.76	4459	31847	0	9.7	-	7millions	0.1
PARD_120_d80	-19049	1.81	1132	7764	0	7.3	1298	1060856	0
moyenne		**2.64**	**2516 (7)**	**32902**	**0.34**	**8.7**	**3905 (6)**	**2703558**	**0.52**

- : Cplex a été arrêté avec un gap final positif après 1 heure de résolution

B.4 Résultats pour le problème d'affectation de tâche

TAB. B.28 – Résultats pour le problème d'affectation de tâches

		QCR			IQCR		
	opt	ig	time (s)	nodes	ig	time (s)	nodes
Task_15_5	-1985	18.98	1.5	3394	0.16	3.3	1
Task_15_5	-1568	37.40	13.4	36328	7.16	26.2	799
Task_15_5	-1892	28.50	7.3	19095	4.30	10.9	273
Task_15_5	-1806	30.07	5.4	13665	3.60	8.5	183
Task_15_5	-1881	25.66	3.7	9483	1.75	6.3	95
Task_15_5	-1950	25.83	2.9	7297	2.09	5.4	71
Task_15_5	-1893	27.19	4.1	10261	2.66	7.4	128
Task_15_5	-1733	36.29	9.4	25101	5.70	20.4	602
Task_15_5	-1798	39.32	17.4	47597	6.93	25.4	797
Task_15_5	-1763	35.15	20.4	55485	5.43	15.7	445
moyenne		**30.44**	**8.6**	**22770**	**3.98**	**13.0**	**339**
Task_18_4	-2136	24.41	4.5	12210	6.20	24.1	702
Task_18_4	-1936	22.00	1.8	4778	6.27	17.1	467
Task_18_4	-2304	19.00	1.1	2723	1.67	5.0	48
Task_18_4	-1926	30.88	6.4	17512	7.84	24.1	725
Task_18_4	-2044	18.20	2.3	6292	4.32	7.9	157
Task_18_4	-2076	24.53	5.3	15117	6.54	20.5	549
Task_18_4	-2251	15.15	1.2	2896	3.61	6.9	111
Task_18_4	-2141	20.60	3.2	8209	4.57	14.9	364
Task_18_4	-1925	27.07	7.4	19899	7.83	35.7	1162
Task_18_4	-2144	23.29	7.0	19069	6.20	16.6	435
moyenne		**22.51**	**4.0**	**10870**	**5.50**	**17.3**	**472**
Task_20_4	-2355	26.07	15.0	32946	8.89	64.6	1508
Task_20_4	-2324	24.06	3.4	7124	3.67	13.6	179
Task_20_4	-2504	25.47	18.7	42184	7.07	54.0	1164
Task_20_4	-2385	27.96	8.5	18347	5.21	22.5	381
Task_20_4	-2225	25.68	11.1	24505	9.00	77.6	1913
Task_20_4	-2474	24.96	8.7	19128	6.15	37.2	746
Task_20_4	-2415	27.92	4.8	10376	9.17	111.7	2776
Task_20_4	-2671	13.95	28.6	66284	4.22	20.3	320
Task_20_4	-2609	23.71	6.0	13086	6.50	43.5	1027
Task_20_4	-2609	21.65	9.9	22230	6.50	43.5	1027
moyenne		**24.14**	**11.5**	**25621**	**6.64**	**48.9**	**1104**
Task_20_5	-2587	37.69	333.2	491174	8.48	306.1	5312
Task_20_5	-2810	34.60	417.7	619072	7.11	187.7	2905
Task_20_5	-2512	40.73	687.3	1019235	12.76	910.9	17408
Task_20_5	-2777	31.72	550.9	821001	9.90	461.0	8138
Task_20_5	-2744	34.44	415.0	622377	6.59	122.7	1737
Task_20_5	-3122	25.13	119.1	186947	4.96	82.7	1046
Task_20_5	-2914	32.65	298.0	463393	6.40	191.3	3123
Task_20_5	-2962	30.41	161.4	250781	5.39	101.2	1471
Task_20_5	-2958	28.41	220.9	342137	7.55	229.9	3718
Task_20_5	-2690	36.78	264.1	403207	8.95	231.6	3735
moyenne		**33.26**	**346.8**	**521932**	**7.81**	**282.5**	**4859**

- : Cplex a été arrêté avec un gap final positif après 1 heure de résolution

TAB. B.29 – Résultats pour le problème d'affectation de tâches (suite)

		QCR			IQCR		
	opt	ig	time (s)	nodes	ig	time (s)	nodes
Task_25_4	-3104	31.06	312.2	447399	12.22	868.6	12234
Task_25_4	-3570	19.70	47.7	65836	5.36	101.1	1022
Task_25_4	-3167	31.30	250.3	352779	12.69	786.5	10103
Task_25_4	-3613	18.16	30.2	42050	5.10	91.1	869
Task_25_4	-3756	17.22	18.2	24134	3.54	46.8	357
Task_25_4	-3449	28.77	650.4	920089	12.18	1707.1	25881
Task_25_4	-3268	29.34	595.9	821218	13.17	1656.7	23044
Task_25_4	-3855	24.60	191.4	282546	8.86	797.0	10698
Task_25_4	-3688	23.45	106.5	149782	6.90	207.3	2371
Task_25_4	-3952	15.95	18.1	24662	3.73	40.2	283
moyenne		**23.95**	**222.1**	**313049**	**8.37**	**630.2**	**8686**
Task_25_5	-3833	37.96	-	4182836	10.71	-	42870
Task_25_5	-3763	36.03	-	4610400	13.18	-	65866
Task_25_5	-4138	29.59	1875.2	1611263	5.49	352.5	2611
Task_25_5	-4001	33.82	-	4411615	10.85	-	43138
Task_25_5	-4045	30.59	-	3895066	10.47	-	35051
Task_25_5	-4112	30.54	-	4311141	8.53	2262.1	20242
Task_25_5	-3751	38.89	-	4645304	15.19	-	96994
Task_25_5	-3795	42.00	-	4619345	14.19	-	106500
Task_25_5	-4153	31.51	-	4310981	8.94	2360.9	20568
Task_25_5	-3885	39.29	-	4582884	13.18	-	70118
moyenne		**35.02**	**1875.2 (1)**	**4118083**	**11.07**	**1658.5 (3)**	**50395**
Task_25_6	-4087	45.87	-	3341578	12.03	-	61235
Task_25_6	-4262	45.16	-	3147476	13.03	-	54074
Task_25_6	-4319	44.84	-	3313781	13.49	-	60600
Task_25_6	-4640	42.08	-	3339751	10.43	-	59285
Task_25_6	-4160	49.67	-	3361442	16.07	-	58727
Task_25_6	-4471	40.68	-	3261752	10.65	-	71630
Task_25_6	-4208	49.10	-	3347215	14.48	-	58800
Task_25_6	-4481	36.45	-	3132881	6.46	1454.7	7369
Task_25_6	-4007	51.68	-	3307351	18.06	-	59200
Task_25_6	-4182	45.81	-	3258622	15.00	-	57800
moyenne		**45.14**	**-**	**3281184**	**12.97**	**1454.7 (1)**	**54872**

- : Cplex a été arrêté avec un gap final positif après 1 heure de résolution

Annexe C

Le logiciel SIQP (Solution of Integer Quadratic Programs)

Le logiciel `SIQP (Solution of Integer Quadratic Programs)` résout des instances de programmes quadratiques soumis à des contraintes linéaires dont la fonction objectif est non convexe. Il implante les méthodes de la plus petite valeur propres, notée ici `EV(EigenValue method)`, `QCR`, `IQCR` pour la programmation quadratique en variables binaires, et les méthodes `BBL`, `BBLr`, `BIL`, `BILr`, `IQCR`, `IQCRs`, et `CQCR` pour la programmation quadratique en variables entières. De plus, `SIQP` fournit un module de génération aléatoire d'instances des classes de problème ($EIQP$) et ($IIQP$). Ce logiciel est distribué sous la licence GNU GPL et ses sources sont disponibles à l'adresse `http ://cedric.cnam.fr/oc/siqp/siqp.tar.gz`. Il est également disponible via une interface internet sur le site `http ://cedric.cnam.fr/oc/siqp/siqp.php`.

Pour tous les algorithmes de résolution, le logiciel `SIQP` peut résoudre les problèmes reformulés MIQP :(Mixed Integer Quadratic Program), ou MILP :(Mixed Integer Linear Program) avec soit le logiciel Bonmin [7], soit le logiciel Cplex [24], soit le logiciel Xpress [10]. En fonction de la méthode utilisée, le logiciel `SIQP` utilise différents logiciels pour reformuler (QP). Le détail des méthodes et des logiciels utilisés est donné dans la suite :

1. **Algorithmes pour résoudre la programmation quadratique en variables binaires :**
   - `EV (EigenValue method)` : Algorithme de résolution de Hammer et Rubin [19]. Cet algorithme utilise Scilab [25] pour calculer les valeurs propres du Hessien de la fonction objectif, puis un solveur MIQP.
   - `QCR (Quadratic Convex Reformulation)` : Algorithme de résolution de Billionnet, Elloumi et Plateau [4]. Cet algorithme utilise Sb [20] pour résoudre le programme semi-défini, puis un solveur MIQP.

– `EQCR (Extended Quadratic Convex Reformulation)` : cet algorithme utilise Sb [20] pour résoudre le programme semi-défini, puis un solveur MIQP.

2. **Algorithmes pour résoudre la programmation quadratique en variables entières :**
   – *Les linéarisations :*
     – `BBL (Binary Binary Linearization)` : cet algorithme utilise uniquement un solveur MILP.
     – `BBLr (Binary Binary Linearization reinforced)` : cet algorithme utilise uniquement un solveur MILP.
     – `BIL (Binary Integer Linearization)` : cet algorithme utilise uniquement un solveur MILP.
     – `BILr (Binary Integer Linearization reinforced)` : cet algorithme utilise uniquement un solveur MILP.
   – *Les convexifications :*
     – `NC (Naive Convexification)` : cet algorithme utilise Scilab [25] pour calculer les valeurs propres du Hessien de la fonction objectif, puis un solveur MIQP.
     – `IQCR (Integer Quadratic Convex Reformulation)` : cet algorithme utilise CSDP [8] pour résoudre le programme semi-défini, puis un solveur MIQP.
     – `IQCRs (Integer Quadratic Convex Reformulation)` : cet algorithme utilise CSDP [8] pour résoudre le programme semi-défini, puis un solveur MIQP.
     – `CQCR (Compact Quadratic Convex Reformulation)` : cet algorithme utilise CSDP [8] pour résoudre le programme semi-défini, puis un solveur MIQP.

Format de fichier d'instance :

Le format de fichier d'entrée du logiciel `SIQP` est le suivant :

$$(QP)\begin{cases} Min & f(x) = x^T Qx + c^T x & \\ s.t & Ax = b & \text{m egalités} \\ & Dx \leq e & \text{p inégalités} \\ & 0 \leq x \leq u & \text{n variables} \\ & x \in \mathbb{N}^n & \end{cases}$$

Où $Q \in \mathbf{S}^n$, $c \in \mathbb{R}^n$, $A \in \mathbf{M}_{m\times n}$, $b \in \mathbb{R}^m$, $D \in \mathbf{M}_{p\times n}$, $e \in \mathbb{R}^p$, et $u \in \mathbb{N}^n$.

Le format de fichier qui représente (QP) est le suivant :

```
n m p
u
u_1 u_2 ... u_n
Q
Nombre d'éléments non nuls de la matrice Q
i j Q_ij (éléments de la matrice Q)
c
Nombre d'éléments non nuls du vecteur c
i c_i (éléments du vecteur c)
A
Nombre d'éléments non nuls de la matrice A
r i A_ri (éléments de la matrice A)
b
Nombre d'éléments non nuls du vecteur b
r b_r  (éléments du vecteur b)
D
Nombre d'éléments non nuls de la matrice E
s i D_si (éléments de la matrice D)
e
Nombre d'éléments non nuls du vecteur e
s e_s (éléments du vecteur e)
```

Si il n'y a pas de contrainte d'égalité ou d'inégalité il ne faut pas mettre les lignes correspondantes dans le fichier d'instance.

Générateur d'instances :

Le logiciel `SIQP` génère aléatoirement des instances des classes de problème $(EIQP)$ et $(IIQP)$ grâce aux paramètres n, m, p, min, max, med et $mult$ utilisés de la façon suivante :

- n est le nombre de variables, m le nombre de contraintes d'égalité, et p le nombre de contraintes d'inégalité.
- Les coefficients de Q et c sont des réels générés aléatoirement dans l'intervalle $[min, max]$. Plus précisément, pour tout $i < j$, un nombre ν est généré dans l'intervalle $[min, max]$, et nous avons $q_{ij} = q_{ji} = \nu$.
- Les coefficients a_{ri} et d_{si} sont des entiers générés aléatoirement dans l'intervalle $[1, med]$.
- Nous avons $b_r = mult * \sum_{i=1}^{n} a_{ri}$ et $e_s = mult * \sum_{i=1}^{n} d_{si}$.

– Nous avons $u_i = med$, pour tout $i \in I$.

Dans une telle instance, la solution $x_i = mult$ pour tout i est réalisable.