Comparaison de reformulations linéaires de programmes quadratiques en nombres entiers

Alain Billionnet[1], Sourour Elloumi[2], Amélie Lambert[2]

1.CEDRIC-ENSIIE, 18 allée Jean Rostand, F-91025 Evry cedex, France
2. CEDRIC-CNAM, 292 rue Saint-Martin, F-75141 Paris cedex 03, France

1 Présentation du problème

De nombreux problèmes de recherche opérationnelle peuvent se modéliser sous la forme d'un programme mathématique à valeurs entières dont la fonction objectif est quadratique et est soumise à des contraintes linéaires. Un problème de ce type peut s'écrire sous la forme :

$$(QP)\begin{cases} Min\ f(x) = x^T Q x + c^T x \\ S.c \quad x \in X \subset \mathbb{N}^n \end{cases}$$

Avec $X = \{Ax = b,\ Dx \leq e,\ 0 \leq x_i \leq u_i,\ x_i \in \mathbb{N},\ i = 1, \ldots, n\}$.
Ce problème appartient à la classe des problèmes NP-difficiles [1].

2 Une reformulation par un programme linéaire 0-1 mixte standard

La façon la plus naturelle pour reformuler (QP) est de transformer ses variables entières en leurs décompositions binaires : $x_i = \sum_{k=0}^{\lfloor log(u_i) \rfloor} 2^k t_{ik}$, puis d'appliquer à ce nouveau problème une linéarisation connue. Ici, nous avons choisi celle de [2] qui consiste à remplacer chaque produit de variables binaires $t_{ik}t_{jl}$ par une nouvelle variable y_{ikjl}. Puis, nous ajoutons les contraintes de linéarisation du produit de variables binaires (5),(6),(7) et (8) pour forcer l'égalité entre $t_{ik}t_{jl}$ et y_{ikjl}. Nous obtenons le programme suivant :

$$(LP_{0-1})\begin{cases} Min\ f_{l0-1}(x,y) = 2\sum_{i=1}^{n}\sum_{k=0}^{\lfloor log(u_i) \rfloor}\sum_{j=i+1}^{n}\sum_{l=0}^{\lfloor log(u_j) \rfloor} 2^k 2^l q_{ij} y_{ikjl} + \sum_{i=1}^{n}\sum_{k=0}^{\lfloor log(u_i) \rfloor} 2^{2k} q_{ii} y_{ikik} + \sum_{i=1}^{n} c_i x_i \\ S.c. \quad Ax = b & (1) \\ \quad Dx \leq e & (2) \\ \quad 0 \leq x_i \leq u_i \quad \forall i & (3) \\ \quad x_i = \sum_{k=0}^{\lfloor log(u_i) \rfloor} 2^k t_{ik} \quad \forall i & (4) \\ \quad y_{ikjl} \leq t_{ik} \quad \forall i, \forall k, \forall j\, \forall l,\ i \leq j,\ q_{ij} < 0 & (5) \\ \quad y_{ikjl} \leq t_{jl} \quad \forall i, \forall k, \forall j\, \forall l,\ i \leq j,\ q_{ij} < 0 & (6) \\ \quad y_{ikjl} \geq t_{ik} + t_{jl} - 1 \quad \forall i, \forall k, \forall j\, \forall l,\ i \leq j,\ q_{ij} > 0 & (7) \\ \quad y_{ikjl} \geq 0 \quad \forall i, \forall k, \forall j\, \forall l,\ i \leq j,\ q_{ij} > 0 & (8) \\ \quad t_{ik} \in \{0,1\} \quad \forall i, \forall k & (9) \end{cases}$$

Si nous notons $N = \sum_{i=1}^{n}(\lfloor log(u_i) \rfloor + 1)$ le nombre de variables t, (LP_{0-1}) possède $O(N^2)$ variables et contraintes, ce qui est une augmentation non négligeable par rapport au problème de départ.

3 Une nouvelle reformulation par un programme linéaire 0-1 mixte

Nous proposons une reformulation linéaire plus compacte de (QP) en un problème équivalent. L'idée est de remplacer chaque produit de variables entières $x_i x_j$ par une nouvelle variable y_{ij}. Ensuite, par un jeu de variables et contraintes linéaires supplémentaires, il faut assurer l'égalité $y_{ij} = x_i x_j$. Cependant, contrairement à la linéarisation du produit de deux variables binaires, on ne sait pas linéariser facilement le produit de deux variables entières. Ici, nous avons choisi de le

faire par une décomposition en puissances de 2 de x_i, $x_i = \sum_{k=0}^{\lfloor log(u_i) \rfloor} 2^k t_{ik}$. Il en découle l'égalité non linéaire $y_{ij} = x_i x_j = \sum_{k=0}^{\lfloor log(u_i) \rfloor} 2^k t_{ik} x_j$, que nous linéarisons en introduisant les variables positives $z_{ijk} = t_{ik} x_j$ ainsi que les contraintes de linéarisation du produit d'une variable binaire par une variable entière (10),(11),(12) et (13). Nous obtenons le programme linéaire suivant :

$$(LP'_{0-1}) \begin{cases} Min\ f_l(x,y) = \sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij} y_{ij} + \sum_{i=1}^{n} c_i x_i \\ S.c \quad (1)\,(2)\,(3)\,(4)\,(9) \\ \quad z_{ijk} \leq u_j t_{ik} \quad \forall i, \forall k, \forall j,\ q_{ij} < 0 & (10) \\ \quad z_{ijk} \leq x_j \quad \forall i, \forall k, \forall j,\ q_{ij} < 0 & (11) \\ \quad z_{ijk} \geq x_j - u_j(1 - t_{ik}) \quad \forall i, \forall k, \forall j,\ q_{ij} > 0 & (12) \\ \quad z_{ijk} \geq 0 \quad \forall i, \forall k, \forall j,\ q_{ij} > 0 & (13) \\ \quad y_{ij} = \sum_{k=0}^{\lfloor log(u_i) \rfloor} 2^k z_{ijk} \quad i,j \in I,\ i < j,\ q_{ij} \neq 0 & (14) \\ \quad \sum_{k=0}^{\lfloor log(u_i) \rfloor} 2^k z_{ijk} = \sum_{k=0}^{\lfloor log(u_j) \rfloor} 2^k z_{jik} \quad i,j \in I,\ q_{ij} \neq 0 & (15) \end{cases}$$

(LP'_{0-1}) possède $O(nN)$ variables et contraintes. Sa taille est donc nettement plus petite que celle de (LP_{0-1}), ce qui permet d'améliorer le temps de résolution. Notons également que la contrainte (15), qui n'est pas nécessaire à la validité de (LP'_{0-1}), force l'égalité entre y_{ij} et y_{ji} et permet d'affiner la valeur de la borne obtenue par relaxation continue de (LP'_{0-1}).

4 Résultats expérimentaux

Pour évaluer la qualité de chaque linéarisation, nous évaluons le temps de résolution des différentes méthodes et les valeurs des bornes obtenues par relaxations continues des problèmes linéarisés. Les tests sur l'efficacité des différentes méthodes ont étés effectués avec le solver Xpress-MP [3]. Ils ont été réalisés sur des instances du problème du sac à dos quadratique où les valeurs de Q sont tirées aléatoirement dans l'intervalle $[-100, 100]$ et sa diagonale est nulle. La contrainte du sac à dos est sous la forme $\sum_{i=1}^{n} d_i x_i \leq \sum_{i=1}^{n} d_i$ où les valeurs des d_i sont tirées dans l'intervalle $[1, 50]$. Les bornes supérieures u_i sur les variables x_i sont toutes égales à 50. Quelques résultats sont présentés dans le tableau suivant. Les colonnes *gap*(%) (resp. *nb noeuds*) représentent l'erreur de la borne à la racine (resp. le nombre de noeuds) de l'algorithme de Branch and Bound.

	(LP_{0-1})			(LP'_{0-1})		
n	gap (%)	nb noeuds	temps(s)	gap (%)	nb noeuds	temps(s)
15	219.658899	7350	32	153.726030	2578	8
25	236.913881	2832	491	167.554454	644	51
50	127.930015	NC	> 3600	80.557525	1241	3145

Ces résultats montrent que (LP'_{0-1}) fournit une bien meilleure borne à la racine, et un temps de résolution plus court que (LP_{0-1}). De plus, (LP'_{0-1}) permet de résoudre des instances qui ont jusqu'à 50 variables entières en moins d'une heure, ce que ne permet pas de faire (LP_{0-1})

Références

1. M.R. Garey and D.S. Johnson, *Computers and Intractability : A guide to the theory of NP-Completness*, Freeman, (1979).
2. R. Fortet *Applications de l'Algèbre de Boole en Recherche Opérationelle*, Revue Française De Recherche Opérationelle [4] 17-25, (1960)
3. Dash Optimization, *Xpress-Mosel version 1.6.1*, Xpress-Mosel language Reference Manual 1.4, 2004, http ://www.dashoptimization.com/, (2005).