PROJET OPTIMISATION

MAIN4

Rapport Projet 1

Auteurs:
Mohamed Legheraba
Lucas Gaudelet
Liliane Kissita
Felix Ernest
Elise Grosjean

January 23, 2017

Contents

1 Introduction

Le but du projet est de résoudre le problème $(\mathcal{P})$ suivant:

$\min_{s.c.} \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij}$ (1)

$\sum_{j=1}^{n} a_{ij}x_{ij} \leq b_i, i \in \{1,\ldots,m\}$, (2)

$\sum_{i=1}^{m} x_{ij} = 1, j \in \{1,\ldots,n\}$, (3)

$x \in \{0,1\}^{m*n}$ (4)

avec $A = a_{ij}$ et $C = c_{ij}$ des matrices de $M_{m*n}(\mathbb{R})$ et $b = b_i$, un vecteur dans $\mathbb{R}$

Le problème $(\mathcal{M})$ suivant est équivalent au problème $(\mathcal{P})$:

$\min_{s.c.} \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij}$ (5)

$\sum_{j=1,j\neq k}^{n} a_{ij}x_{ij}x_{ik} \leq (b_i - a_{ik})x_{ik}, i,k \in \{1,\ldots,m\}$, (6)

$\sum_{j=1,j\neq k}^{n} a_{ij}x_{ij} - \sum_{j=1}^{n} a_{ij}x_{ij}x_{ik} \leq b_i - b_i x_{ik}, i,k \in \{1,\ldots,m\}$, (7)

$\sum_{i=1}^{m} x_{ij} = 1, j \in \{1,\ldots,n\}$, (8)

$x \in \{0,1\}^{m*n}$ (9)

Le modèle suivant, linéarisé du problème $(\mathcal{M})$ est appelé $(\mathcal{L})$:

$\min_{s.c.} \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij}$

$\sum_{j=1}^{n} a_{ij}z_{jk}^{i} \leq (b_i - a_{ik})x_{ik}, i,k \in \{1,\ldots,m\}$, (10)

$\sum_{j=1,j\neq k}^{n} a_{ij}x_{ij} - \sum_{j=1}^{n} a_{ij}z_{jk}^{i} \leq b_i - b_i x_{ik}, i,k \in \{1,\ldots,m\}$, (11)

$\sum_{i=1}^{m} x_{ij} = 1, j \in \{1,\ldots,n\}$, (12)

$z_{jk}^{i} \leq x_{ij}, \forall i,j,k$ (13)

$z_{jk}^{i} \leq x_{ik}, \forall i,j,k$ (14)

$x_{ik} + x_{ij} - z_{jk}^{i} \leq 1, \forall i,j,k$ (15)

$x \in \{0,1\}^{m*n}, z \geq 0$ (16)

2 Partie A: Résolution exacte

2.1 Equivalence des modèles $(\mathcal{L})$ et $(\mathcal{P})$

Soit le problème $(\mathcal{P})$:

$\min \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij}$ (1)

$\sum_{j=1}^{n} a_{ij}x_{ij} \leq b_i, i \in \{1,\ldots,m\},$ (2)

$\sum_{i=1}^{m} x_{ij} = 1, j \in \{1,\ldots,n\},$ (3)

$x \in \{0,1\}^{m*n}$ (4)

En multipliant (2) par x_{ik} puis $(1-x_{ik})$ on obtient le problème $(\mathcal{M})$:

$\min \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij}$ (5)

$\sum_{j=1,j\neq k}^{n} a_{ij}x_{ij}x_{ik} \leq (b_i - a_{ik})x_{ik}, i,k \in \{1,\ldots,m\},$ (6)

$\sum_{j=1,j\neq k}^{n} a_{ij}x_{ij} - \sum_{j=1,j\neq k}^{n} a_{ij}x_{ij}x_{ik} \leq b_i - b_i x_{ik}, i,k \in \{1,\ldots,m\},$ (7)

$\sum_{i=1}^{m} x_{ij} = 1, j \in \{1,\ldots,n\},$ (8)

$x \in \{0,1\}^{m*n}$ (9)

Si ensuite on remplace les termes $x_{ij}x_{ik}$ par z^i_{jk}. Il faut ajouter des contraintes sur z. Si on a x_{ij} ou x_{ik} égaux à 0 alors $z^i_{jk} = 0$ et sinon $z^i_{jk} = 1$ donc $z^i_{jk} \leq x_{ik}$ et $z^i_{jk} \leq x_{ij}.z \geq 0$ puisque z=1 ou 0.

Il manque encore une contrainte qui indique que z est à valeurs dans $\{0,1\}^{m*n}$. On ajoute donc la contrainte $z^i_{jk} \geq x_{ij} + x_{ik} - 1$.
En effet, dans le premier cas, x_{ij} et x_{ik} valent tous les deux 0. Et donc $z \leq 0$ et $z \geq -1$. Comme $z \geq 0$, z=0. Dans le deuxième cas, x_{ik} ou x_{ij} valent 0 et l'autre vaut 1, donc $z \leq 0$ et $z \geq -1+1+0 = 0$, donc z=0. Et dans le dernier cas, $x_{ij} = 1$ et $x_{ik} = 1$, donc $z \leq 1$ et $z \geq -1+1+1 = 1$, donc z=1.

Les deux modèles sont donc équivalents.

2.2 Valeur optimale de relaxation

Soit opt^P et opt^L les valeurs optimales de la relaxation des problèmes $(\mathcal{P})$ et $(\mathcal{L})$.

On vérifie bien que $opt^P \leq opt^L$ sur toute les instances, opt^L étant lui même inférieur à la valeur entière de la solution.

2.3 Performances des solveurs sur les modèles $(\mathcal{L})$ et $(\mathcal{P})$

Nous avons réussi à résoudre les instances des modèles L et P avec les solveurs de AMPL. Les résultats sont contenues dans les fichiers logsampl_L.txt et logsampl_P.txt . Nous avons aussi les résultats des relaxations de ces mêmes problèmes.

On observe que les solveurs sont très performants car pour les petites instances nous obtenons des résultats en moins d'une seconde. Par contre pour de grosses instances le temps augmente très fortement, pour atteindre plus de 100 secondes.

2.4 Performances des solveurs sur le modèle $(\mathcal{M})$

Les solveurs marchent également sur le modèle M, même si ils sont plus lents. Pour la petite instance, il faut un peu plus d'une seconde pour la résoudre et entre 10 et 30 secondes pour les moyennes instances.

3 Partie B : Résolution approchée

3.1 Performances sur $(\mathcal{P})$

La fonction loadfile permet de récupérer les jeux de données A,b et C du problème $(\mathcal{P})$.
Nous avons ensuite réécrit x qui est de la même taille que A sous la forme d'un vecteur colonne de taille m*n pour pouvoir écrire la contrainte $\sum_{j=1}^{n} a_{ij}x_{ij} \leq b_i$ sous forme matricielle. On a donc écrit dans "script.m" une nouvelle matrice B de taille (m,m*n) diagonale où

$$B = \begin{pmatrix} a_{11} & \cdots & a_{1n} & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & \cdots & 0 & a_{21} & \cdots & a_{2n} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \ddots & \ddots & \ddots & 0 & \cdots & 0 \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & a_{m1} & \cdots & a_{mn} \end{pmatrix}$$

On a $x = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1n} \\ \vdots \\ x_{m1} \\ x_{mn} \end{pmatrix}$

Donc on retrouve bien la contrainte d'inégalité.
Nous avons coder sur Matlab $ga(fun, 500, B, b, [], [], lb, ub, @nonlinearga, IntCon)$, avec fun la fonction objective,500 le nombre de variables pour le premier jeu de données (m=5 et n=100) et B la nouvelle matrice des contraintes, où IntCon indique que l'on souhaite des variables entières et @nonlinearga est la fonction qui définie les contraintes d'égalités car ga ne permet pas d'utiliser Aeq et beq simultanément avec IntCon.
Nous trouvons comme résultat z= 3481 et x pour remettre sous la forme initiale (une matrice de taille m*n) avec des 0 et des 1.
Pour Patternsearch, nous avons utilisé la commande $x = patternsearch(fun, X0, B, b, Aeq, beq, lb, u$
Aeq est la contrainte d'égalité.
Comme on a $\sum_{i=1}^{m} x_{ij} = 1$, Aeq est une matrice de taille n,n*m (m blocs de la matrice Identité).

On a par exemple pour m=2, $Aeq = \begin{pmatrix} 1 & 0 & \cdots & 1 & 0 & \cdots \\ 0 & \ddots & 0 & 0 & \ddots & 0 \\ 0 & \cdots & 1 & 0 & \cdots & 1 \end{pmatrix}$

et on garde le même x en vecteur colonne que tout à l'heure; beq est un vecteur colonne de taille n dont les composants valent 1.
On trouve z= 3212 pour le premier jeu de donnée avec m=5 et n=100.
Les autres résultats sont détaillés dans Mat.txt
Les résultats en relâchant la contrainte (10) de L ont été générés avec le solveur cplex avec la commande display j in 1..._nvars (_varname[j],_var[j],_var[j].rc) . Les résultats sont présentés dans "Mat2.txt".

4 Partie C: Algorithme du sous-gradient

Nous n'avons malheureusement pas eu le temps de nous pencher sur l'algorithme du sous gradient.

5 Conclusion

En conclusion, grâce à ce projet nous avons compris l'intérêt de la relaxation lagrangienne pour les problèmes compliquées. En effet si l'on essaie de résoudre le problème M directement cela prend beaucoup de temps alors que le problème P équivalent est résolut beaucoup plus rapidement par les solveurs AMPL.

De même nous avons vu que l'algorithme ga de MathLab est beaucoup plus adapté pour résoudre les problèmes avec un mélange de variables continues et entières alors que le patternsearch ne peut pas résoudre ce type de problème.

Nous n'avons pas eu le temps de regarder en profondeur l'algorithme du sous gradiant mais il ne fait pas de doute que cet algorithme peut être très efficace pour résoudre des problèmes du même style que M.