MPRO

Étude de Cas et Management de la Recherche Opérationnelle

ECMA

Plus Court Chemin Robuste

Antoine Desjardins
Martin Lainée

Table des matières

1 Introduction

Pour cette partie du projet, nous avons reprit la modélisation proposée dans le corrigé. Nous travaillons en Julia 1.6.3. Notre repo peut être trouvé à l'adresse https ://github.com/boumpteryx/ECMA

2 Résolution par Dualisation

Nous avons encodé le dual proposé dans le corrigé, question 4) g). Afin d'encoder les arcs qui existent, nous avons choisi de créer une matrice d'arcs pour le graphe complet mais de n'utiliser que les variables d'arcs correspondant aux arcs ayant une latence, i.e. à ceux ayant une valeur non-nulle dans la matrice de latence. Ainsi, nous créons trop de variables mais n'utilisons pas celles qui ne sont pas utile, ce qui ne dois pas affecter les performances du solveur. Ce choix a été fait pour rendre le code plus simple et plus lisible.

3 Résolution par Plans Coupants

La résolution par plans coupants s'effectue en résolvant tout d'abord le problème maître. Pui, tant que la condition suivant n'est pas vérifiée :

```
z2 >= S + 1e-3 || z1 > zstar + 1e-3 || z1 < zstar - 1e-3
```

Nous résolvons les deux problèmes esclaves et ajoutons des contraintes au problème maître et recommençons sa résolution. Le choix de la précision est nécessaire pour gérer l'inexactitude des calculs numériques. La borne de 10^{-3} est arbitraire.

4 Résolution par Branch-and-Cut

La résolution par plans coupants suit le même procédé que la résolution par plans coupants mais en utilisant des `LazyCallBack` à chaque fois qu'une solution entière est trouvée et que la condition (la même que pour les plans coupants) est vérifiée. Nous avons choisi la même borne de précision afin de comparer les deux algorithmes toute chose égales par ailleurs.

5 Résolution par heuristique

5.1 Borne

Nous obtenons une borne inf sur le problème à l'aide de la résolution du problème relaxé. Ce problème s'exprime de la même façon que le dual en relaxant certaines contraintes qui deviennent :

$x_{ij}, y_v \geq 0 \quad \forall ij \in A, \forall v \in V$
$x_{ij}, y_v \leq 1 \quad \forall ij \in A, \forall v \in V$

5.2 Heuristique

Notre heuristique se base sur un arbre de recherche. Pour que celui-ci puisse parcourir les chemins de manière efficace, on lui adjoint plusieurs méthodes.
La première est une simple utilisation de Dijkstra avec un poids sur chaque arc égal à 1. Ceci nous permet de savoir pour chaque sommet la longueur en arcs du plus court chemin de s à t.
Grâce à cette donnée, on met à jour la distance de chaque arc en lui ajoutant le minimum entre le pourcentage d'augmentation maximal divisé par la longueur du plus court chemin au sein duquel se trouve l'arc et le pourcentage maximal d'augmentation de cet arc. Ceci nous permet d'obtenir un graphe de distances qui reflète mieux la nature aléatoire du problème.
On calcule de plus pour chaque sommet, à l'aide du poids minimal des sommets et la distance la plus courte en arcs au sommet t, le poids maximal qu'un chemin de s à ce sommet peut avoir tout en respectant finalement la contrainte de poids. Ceci permet d'éliminer des branches de l'arbre sans avoir à l'explorer jusqu'aux feuilles.
Le poids d'un chemin non terminé est calculé initialement sans prendre en compte les augmentations possibles. Ensuite, dès lors que ce chemin atteint un poids qui viole potentiellement la contrainte de poids maximal, on calcule le poids robuste de ce chemin partiel. Par la suite, on se contente d'ajouter les poids

non robustes.
Enfin, la stratégie de branchement se fait d'abord par ordre de distance en arc à t, puis par ordre des distances heuristiques calculées précédemment. Pour éviter de trop s'éloigner de t, on limite globalement ainsi qu'à chaque étape la différence entre les plus courts chemins en terme d'arcs d'un sommet et du sommet suivant.

6 Résultats numériques

Nous présentons dans cette section les résultats obtenus durant la résolution des instances fournies en utilisant les différentes méthodes exposées ci-dessus.

Instance	Dual	Branch and Cut	Plans coupants	Heuristique	Objectif
20_BAY	0.80, 0	0.71, 0	2.26, 0	0.9, 34	1529.16
40_BAY	0.86, 0	0.55, 0	1.16, 0	0.60, 59	932.36
60_BAY	1.06, 0	1.15, 0	1.17, 0	1.45, 333	244.98
80_BAY	1.51, 0	3.41, 0	2.63, 0	10, 81	598.68
100_BAY	2.54, 0	19.99, 0	5.43, 0	10, 87	598.68
200_BAY	27.2, 0	200.90, 0	179, 0	60, 116	706.72
20_COL	0.80, 0	0.65, 0	1.176, 0	0.9, 34	627.62
40_COL	1.04, 0	13.52, 0	180.69, 0	10, 23	1501.03
60_COL	1.24, 0	0.89, 0	0.42, 0	10, 337	625.0
80_COL	1.69, 0	3.07, 0	2.21, 0	10, 107	704.36
100_COL	2.77, 0	6.42, 0	4.22, 0	N/A	531.26
200_COL	31.8, 0	81.45, 0	42.85, 0	529, 216	1159.72
20_NY	0.90, 0	0.28, 0	0.19, 0	2.8, 132	406.16
40_NY	1.0, 0	0.37, 0	0.30, 0	10, 125	783.58
60_NY	1.2, 0	1.18, 0	0.85, 0	2.55, 186	1157.4
80_NY	1.75, 0	3.44, 0	3.01, 0	8, 198	1157.4
100_NY	2.84, 0	6.74, 0	4.06, 0	60, 378	753.24
200_NY	29.6, 0	97.45, 0	97.24, 0	60, 77	1998.84

Table 1 – Performances en seconde, % de la solution optimale

Pour l'instance 100_COL, notre heuristique ne permet pas de trouver de solution réalisable en moins de 10 minutes, la limite de temps qui nous avions imposé.
Pour des raisons de temps de calcul, nous n'avons pas réalisé de tests systématiques sur des instances plus grosses. Cependant, sur l'instance 400_NY par exemple, le dual met 510 secondes à converger, alors que notre heuristique donne une solution valable (qui reste néanmoins 3 fois supérieure à l'optimale) en une minute. Ceci illustre bien l'intérêt de cette technique approchée, pour obtenir des solutions réalisables en temps raisonnable.

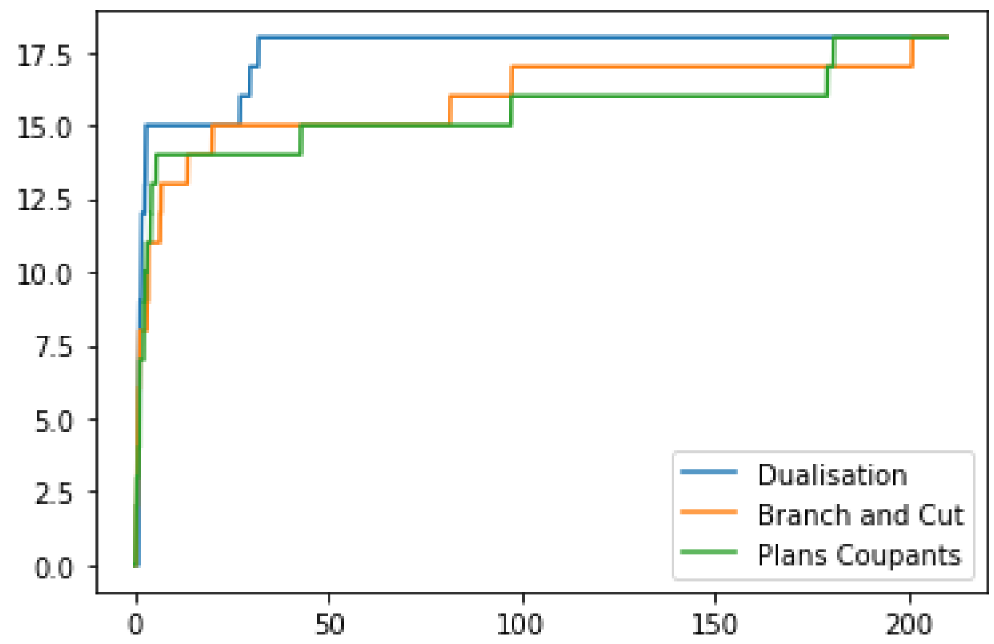

FIGURE 1 – Nombre d'instances résolues en fonction du temps (en secondes)

7 Annexe

Nous fournissons ici la solution duale optimale sous la forme de la valeur des variables pertinentes. Eut égard à la longueur de cette forme, nous ne donnons ici que la valeur pour l'instance "20_USA-road-d.BAY.gr".

objectif 1529.1599999999999

x_star [0.0 -0.0 -0.0 -0.0 0.0 -0.0 -0.0 -0.0 0.0 0.0 -0.0 0.0 0.0 -0.0 0.0 0.0 0.0 -0.0 -0.0 0.0 ; -0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 -0.0 0.0 0.0 0.0 -0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 ; -0.0 0.0 0.0 -0.0 0.0 0.0 0.0 0.0 -0.0 0.0 -0.0 0.0 -0.0 0.0 0.0 0.0 0.0 0.0 -0.0 0.0 ; 0.0 0.0 -0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 -0.0 0.0 -0.0 0.0 0.0 0.0 0.0 ; -0.0 0.0 0.0 0.0 0.0 -0.0 0.0 0.0 -0.0 -0.0 0.0 0.0 -0.0 0.0 0.0 -0.0 1.0 -0.0 0.0 0.0 ; -0.0 0.0 0.0 0.0 0.0 0.0 -0.0 0.0 -0.0 0.0 0.0 -0.0 -0.0 0.0 0.0 0.0 0.0 0.0 0.0 -0.0 ; -0.0 0.0 0.0 0.0 0.0 -0.0 0.0 0.0 -0.0 0.0 0.0 0.0 -0.0 0.0 0.0 0.0 0.0 -0.0 0.0 0.0 ; -0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 -0.0 0.0 0.0 0.0 -0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 ; 0.0 -0.0 -0.0 -0.0 0.0 -0.0 -0.0 -0.0 0.0 0.0 -0.0 -0.0 0.0 -0.0 0.0 0.0 0.0 -0.0 -0.0 -0.0 ; 0.0 0.0 0.0 -0.0 -0.0 0.0 0.0 0.0 0.0 0.0 -0.0 -0.0 0.0 0.0 0.0 0.0 0.0 0.0 -0.0 -0.0 ; 0.0 0.0 -0.0 0.0 0.0 0.0 0.0 0.0 -0.0 -0.0 0.0 0.0 -0.0 -0.0 -0.0 -0.0 0.0 0.0 0.0 0.0 ; -0.0 0.0 0.0 0.0 0.0 -0.0 0.0 0.0 -0.0 -0.0 0.0 0.0 -0.0 0.0 0.0 -0.0 -0.0 -0.0 0.0 0.0 ; 0.0 -0.0 -0.0 -0.0 1.0 -0.0 -0.0 -0.0 0.0 0.0 -0.0 -0.0 0.0 -0.0 0.0 0.0 0.0 -0.0 -0.0 -0.0 ; -0.0 0.0 0.0 -0.0 0.0 0.0 0.0 0.0 -0.0 0.0 -0.0 0.0 -0.0 0.0 0.0 0.0 0.0 0.0 -0.0 0.0 ; 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 -0.0 0.0 ; 0.0 0.0 0.0 -0.0 0.0 0.0 0.0 0.0 0.0 0.0 -0.0 -0.0 0.0 0.0 0.0 0.0 0.0 0.0 -0.0 -0.0 ; 0.0 0.0 0.0 0.0 -0.0 0.0 0.0 0.0 0.0 0.0 0.0 -0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 -0.0 ; -0.0 0.0 0.0 0.0 0.0 0.0 -0.0 0.0 -0.0 0.0 0.0 -0.0 -0.0 0.0 0.0 0.0 0.0 0.0 0.0 -0.0 ; 0.0 0.0 -0.0 0.0 0.0 0.0 0.0 0.0 -0.0 -0.0 0.0 0.0 -0.0 -0.0 -0.0 -0.0 0.0 0.0 0.0 0.0 ; -0.0 0.0 0.0 0.0 0.0 -0.0 0.0 0.0 -0.0 -0.0 0.0 0.0 -0.0 0.0 0.0 -0.0 0.0 -0.0 0.0 0.0] y_star [0.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, -0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0] z_star [0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 0.0 0.0 -0.0 -0.0 0.0 -0.0 0.0 0.0 0.0 -0.0 -0.0 -0.0 ; -0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 -0.0 0.0 0.0 0.0 -0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 ; -0.0 0.0 0.0 -0.0 0.0 0.0 0.0 0.0 -0.0 0.0 -0.0 0.0 -0.0 0.0 0.0 0.0 0.0 0.0 -0.0 0.0 ; -0.0 0.0 -0.0 0.0 0.0 0.0 0.0 0.0 -0.0 -0.0 0.0 0.0 -0.0 -0.0 -0.0 -0.0 0.0 0.0 0.0 0.0 ; -0.0 0.0 0.0 0.0 0.0 -0.0 0.0 0.0 -0.0 -0.0 0.0 0.0 -0.0 0.0 0.0 -0.0 70.0 -0.0 0.0 0.0 ; -0.0 0.0 0.0 0.0 -0.0 0.0 -0.0 0.0 -0.0 0.0 0.0 -0.0 -0.0 0.0 0.0 0.0 0.0 0.0 0.0 -0.0 ; -0.0 0.0 0.0 0.0 0.0 -0.0 0.0 0.0 -0.0 0.0 0.0 0.0 -0.0 0.0 0.0 0.0 0.0 -0.0 0.0 0.0 ; -0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 -0.0 0.0 0.0 0.0 -0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 ; 0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 0.0 0.0 -0.0 -0.0 0.0 -0.0 0.0 0.0 0.0 -0.0 -0.0 -0.0 ; 0.0 0.0 0.0 -0.0 -0.0 0.0 0.0 0.0 0.0 0.0 -0.0 -0.0 0.0 0.0 0.0 0.0 0.0 0.0 -0.0 -0.0 ; -0.0 0.0 -0.0 0.0 0.0 0.0 0.0 0.0 -0.0 -0.0 0.0 0.0 -0.0 -0.0 -0.0 -0.0 0.0 0.0 0.0 0.0 ; -0.0 0.0 0.0 0.0 0.0 -0.0 0.0 0.0 -0.0 -0.0 0.0 0.0 -0.0 0.0 0.0 -0.0 -0.0 -0.0 0.0 0.0 ; 0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 0.0 0.0 -0.0 -0.0 0.0 -0.0 0.0 0.0 0.0 -0.0 -0.0 -0.0 ; -0.0 0.0 0.0 -0.0 0.0 0.0 0.0 0.0 -0.0 0.0 -0.0 0.0 -0.0 0.0 0.0 0.0 0.0 0.0 -0.0 0.0 ; 0.0 0.0 0.0 37.0 0.0 0.0 0.0 0.0 0.0 0.0 -0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 -0.0 0.0 ; 0.0 0.0 0.0 -0.0 -0.0 0.0 0.0 0.0

0.0 0.0 -0.0 -0.0 0.0 0.0 0.0 0.0 0.0 0.0 -0.0 -0.0 ; 0.0 0.0 0.0 0.0 -0.0 0.0 0.0 0.0 0.0 0.0 0.0 -0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 -0.0 ; -0.0 0.0 0.0 0.0 -0.0 0.0 -0.0 0.0 -0.0 0.0 0.0 -0.0 -0.0 0.0 0.0 0.0 0.0 0.0 0.0 -0.0 ; -0.0 0.0 -0.0 0.0 0.0 0.0 0.0 0.0 -0.0 -0.0 0.0 0.0 -0.0 -0.0 -0.0 -0.0 0.0 0.0 0.0 0.0 ; -0.0 0.0 0.0 0.0 0.0 -0.0 0.0 0.0 -0.0 -0.0 0.0 0.0 -0.0 0.0 0.0 -0.0 -0.0 -0.0 0.0 0.0]
t1_star 247.0
t2_star 4.0
zprim_star [-0.0, -0.0, -0.0, -0.0, -0.0, -0.0, -0.0, -0.0, -0.0, -0.0, -0.0, -0.0, -0.0, -0.0, 0.0, -0.0, 0.0, -0.0, -0.0, -0.0]