Projet Optimisation Robuste
Problème du plus court chemin

Benoit Duval, Raphaël Taisant

9 février 2024

Partie 1: Modélisation papier

1 Notations

Définissons les termes que nous allons utiliser. Soient:

- un graphe orienté $G = (V, A)$
- un sommet origine $s \in V$ et un sommet destination $t \in V$
- une durée de trajet d_{ij} associée à chaque arc $ij \in A$
- un poids p_i associé à chaque sommet $i \in V$

On a $d_{ij} \geq 0$ et $p_i \geq 0$.

Dans ce projet, on cherche à trouver un plus court chemin (en terme de temps) de s à t dont le poids des sommets utilisés est inférieur ou égal à un entier S.

Sans perte de généralité, on considérera que les graphes que nous traitons n'admettent pas d'arc entrant en s ou sortant de t. On peut s'y ramener facilement en créant des sommets s' et t' avec des poids nuls et on crée des arcs $s's$ et tt' avec des coûts nuls également. Le noeud d'entrée devient s' et celui de sortie devient t'. En pratique, on peut également supprimer les arcs entrant en s/ sortants de t. On peut également forcer les arcs entrants en s/ sortant de t à ne pas être utilisés. Tout cela donne les mêmes solutions optimales.

2 Modélisation problème statique

Intéressons nous dans un premier temps au problème statique. Utilisons les variables binaires suivantes:

- $x_{ij} = 1$ si l'arc ij est utilisé, 0 sinon
- $y_i = 1$ si l'on passe par la ville i, 0 sinon

Pour simplifier les notations, on pourra considérer que les x_{ij} sont à la fois définis pour tout $i, j \in V^2$ (avec valeur 0 si $ij \notin A$) et définis uniquement pour $ij \in A$.

On peut modéliser ce problème à l'aide du PLNE 1:

$$
\begin{array}{ll}
min_{x,y} & \sum_{ij\in A} d_{ij}x_{ij} \\
st & \sum_{i\in V} y_i p_i \leq S \\
& \sum_{j\in V} x_{sj} = 1 \\
& \sum_{i\in V} x_{it} = 1 \\
& \sum_{j\in V} x_{ij} = y_i \quad \forall\, i \in V \setminus \{t\} \\
& \sum_{i\in V} x_{ij} = y_j \quad \forall\, j \in V \setminus \{s\} \\
& y_s = 1, y_t = 1 \\
& x_{ij} \in \{0,1\} \qquad \forall\, ij \in A \\
& y_i \in \{0,1\} \qquad \forall\, i \in V
\end{array}
\tag{1}
$$

3 Modélisation problème robuste

On considère maintenant des incertitudes sur les durées.

$$
\mathcal{U}^1 = \left\{ \left\{ d^1_{ij} = d_{ij}\left(1+\delta^1_{ij}\right) \right\}_{ij\in A} \text{ st } \sum_{ij\in A} \delta^1_{ij} \leq d_1, \quad \delta^1_{ij} \in [0, D_{ij}] \quad \forall\, ij \in A \right\}
$$

Avec D_{ij} et d_1 donnés. On considère également des incertitudes sur les poids.

$$
\mathcal{U}^2 = \left\{ \left\{ p^2_i = p_i + \delta^2_i \hat{p}_i \right\}_{i\in V} \text{ st } \sum_{i\in V} \delta^2_i \leq d_2, \quad \delta^2_i \in [0,2] \quad \forall\, i \in V \right\}
$$

Avec $\hat{p}_i$ et d_2 donnés. On veut maintenant résoudre le problème robuste 2:

$$
\begin{array}{ll}
\min_{x,y} \max_{d^1\in\mathcal{U}^1} & \sum_{ij\in A} d^1_{ij}x_{ij} \\
st & \sum_{i\in V} y_i p^2_i \leq S \qquad \forall p^2 \in \mathcal{U}^2 \\
& \sum_{j\in V} x_{sj} = 1 \\
& \sum_{i\in V} x_{it} = 1 \\
& \sum_{j\in V} x_{ij} = y_i \quad \forall\, i \in V \setminus \{t\} \\
& \sum_{i\in V} x_{ij} = y_j \quad \forall\, j \in V \setminus \{s\} \\
& y_s = 1, y_t = 1 \\
& x_{ij} \in \{0,1\} \qquad \forall\, ij \in A \\
& y_i \in \{0,1\} \qquad \forall\, i \in V
\end{array}
\tag{2}
$$

4 Résolution par plans coupants

On peut résoudre ce problème en utilisant des plans coupants. Pour cela, on commence par reformuler le problème robuste en faisant en sorte que la robustesse n'apparaisse plus dans l'objectif mais uniquement dans les contraintes, ce qui est fait dans le Pb. 3.

$$
\begin{array}{ll}
\min_{x,y,z} & z \\
st & \sum_{i\in V} y_i p^2_i \leq S \qquad \forall p^2 \in \mathcal{U}^2 \\
& \sum_{j\in V} x_{sj} = 1 \\
& \sum_{i\in V} x_{it} = 1 \\
& \sum_{j\in V} x_{ij} = y_i \quad \forall\, i \in V \setminus \{t\} \\
& \sum_{i\in V} x_{ij} = y_j \quad \forall\, j \in V \setminus \{s\} \\
& y_s = 1, y_t = 1 \\
& z \geq \sum_{ij\in A} d^1_{ij}x_{ij} \qquad \forall d^1 \in \mathcal{U}^1 \\
& x_{ij} \in \{0,1\} \qquad \forall\, ij \in A \\
& y_i \in \{0,1\} \qquad \forall\, i \in V
\end{array}
\tag{3}
$$

On définit le problème maître à partir du Pb. 3 en remplaçant $\mathcal{U}^1$ et $\mathcal{U}^2$ par respectivement $\mathcal{U}^{1*}$ et $\mathcal{U}^{2*}$. Initialement, pour résoudre le problème maître, on peut partir de $\mathcal{U}^{1*} = \varnothing$ et $\mathcal{U}^{2*} = \varnothing$.

Après avoir résolu le problème maître et obtenu des x^*, y^* et z^*, on résout les sous problèmes ci-dessous. Pour les contraintes sur les durées, on résout le Pb. 4:

$$\begin{array}{rlll} V^1 = & \max_{\delta^1} & \sum_{ij\in A} d_{ij}\left(1+\delta^1_{ij}\right)x^*_{ij} & \\ & st & \sum_{ij\in A}\delta^1_{ij} \leq d_1 & \\ & & \delta^1_{ij} \in [0, D_{ij}] & \forall\, ij \in A \end{array} \tag{4}$$

Pour les contraintes de poids, on résout le Pb. 5

$$\begin{array}{rlll} V^2 = & \max_{\delta^2} & \sum_{i\in V} y^*_i\left(p_i+\delta^2_i\hat{p}_i\right) & \\ & st & \sum_{i\in V}\delta^2_i \leq d_2 & \\ & & \delta^2_i \in [0,2] & \forall\, ij \in A \end{array} \tag{5}$$

Si on a:

$$\begin{cases} z^* \geq V^1 \\ S \geq V^2 \end{cases}$$

Alors, la solution du problème maître est optimale.

Si la contrainte de durée n'est pas respectée, on ajoute la coupe suivante au problème maître:

$$\sum_{ij\in A} d_{ij}\left(1+\delta^{1*}_{ij}\right)x_{ij} \leq z$$

Si la contrainte de poids n'est pas respectée, on ajoute la coupe suivante au problème maître:

$$\sum_{i\in V} x_i\left(p_i+\delta^{2*}_i\hat{p}_i\right) \leq S$$

On résout le problème maître à la racine, puis les sous-problèmes. Tant que les conditions d'optimalité ne sont pas respectées, on ajoute des coupes et on itère.

5 Résolution par dualisation

On peut également résoudre ce problème en utilisant de la dualisation. On peut réécrire le problème comme présenté dans le Pb. 6, en explicitant les deux sous problèmes avec des couleurs différentes:

$$\begin{array}{rlll} \min_{x,y}\max_{\delta^1} & \sum_{ij\in A} d_{ij}(1+\delta^1_{ij})x_{ij} & & \\ st & \sum_{ij\in A}\delta^1_{ij} \leq d_1 & & \\ & \delta^1_{ij} \in [0, D_{ij}] \quad \forall\, ij \in A & & \\ & \max_{\delta^2}\sum_{i\in V}\left(p_i+\delta^2_i\hat{p}_i\right)y_i \leq S & & \\ & \quad \sum_{i\in V}\delta^2_i \leq d_2 & & \\ & \quad \delta^2_i \in [0,2] \quad \forall\, i \in V & & \\ & \sum_{j\in V} x_{sj} = 1 & & \\ & \sum_{i\in V} x_{it} = 1 & & \\ & \sum_{j\in V} x_{ij} = y_i & \forall\, i \in V\setminus\{t\} & \\ & \sum_{i\in V} x_{ij} = y_j & \forall\, j \in V\setminus\{s\} & \\ & y_s = 1, y_t = 1 & & \\ & x_{ij} \in \{0,1\} & \forall\, ij \in A & \\ & y_i \in \{0,1\} & \forall\, i \in V & \end{array} \tag{6}$$

On isole le problème lié à la fonction objectif, Pb. 9:

$$\begin{array}{lll} \max_{\delta^1} & \sum_{ij\in A} d_{ij}(1+\delta^1_{ij})x_{ij} & \\ st & \sum_{ij\in A} \delta^1_{ij} \leq d_1 & (\eta) \\ & \delta^1_{ij} \leq D_{ij} \quad \forall\, ij \in A & (\lambda_{ij}) \\ & \delta^1_{ij} \geq 0 \quad \forall\, ij \in A & \end{array} \tag{7}$$

On enlève le terme en $d_{ij}x_{ij}$ de l'objectif puisqu'il est indépendant de δ^1, et on peut dualiser le problème pour obtenir un problème de minimisation, en associant la variable duale η a la contrainte sur la somme des δ^1_{ij} et les variables λ_{ij} aux bornes sup sur les δ^1_{ij}. On obtient le Pb. 8:

$$\begin{array}{lll} \min_{\lambda,\eta} & d_1\eta + \sum_{ij\in A} (d_{ij}x_{ij} + D_{ij}\lambda_{ij}) & \\ st & \eta + \lambda_{ij} \geq d_{ij}x_{ij} & \forall\, ij \in A \\ & \eta \geq 0, \lambda_{ij} \geq 0 \quad \forall\, ij \in A & \end{array} \tag{8}$$

De même, on isole le problème lié à la contrainte robuste, Pb. 9:

$$\begin{array}{lll} \max_{\delta^2} & \sum_{i\in V} (p_i + \delta^2_i \hat{p}_i)\, y_i & \\ st & \sum_{i\in V} \delta^2_i \leq d_2 & (\alpha) \\ & \delta^2_i \leq 2 \quad \forall\, i \in V & (\beta_i) \\ & \delta^2_i \geq 0 \quad \forall\, i \in V & \end{array} \tag{9}$$

On dualise le problème associé à la contrainte robuste, en introduisant la variable duale α à la contrainte sur la somme des δ^2_i et les variables β_i aux bornes sup sur les δ^2_i. On obtient alors le Pb. 10:

$$\begin{array}{lll} \min_{\alpha,\beta} & d_2\alpha + \sum_{i\in V} (p_i y_i + 2\beta_i) & \\ st & \alpha + \beta_i \geq \hat{p}_i y_i & \forall\, ij \in A \\ & \alpha \geq 0, \beta_i \geq 0 \quad \forall\, i \in V & \end{array} \tag{10}$$

On peut désormais intégrer les deux reformulations afin d'obtenir un seul problème linéaire en nombre entier, présenté dans le Pb. 11:

$$\begin{array}{lll} \min_{x,y,\eta,\lambda,\alpha,\beta} & d_1\eta + \sum_{ij\in A} (d_{ij}x_{ij} + D_{ij}\lambda_{ij}) & \\ st & \eta + \lambda_{ij} \geq d_{ij}x_{ij} & \forall\, ij \in A \\ & d_2\alpha + \sum_{i\in V} (p_i y_i + 2\beta_i) \leq S & \\ & \alpha + \beta_i \geq \hat{p}_i y_i & \forall\, i \in V \\ & \sum_{j\in V} x_{sj} = 1 & \\ & \sum_{i\in V} x_{it} = 1 & \\ & \sum_{j\in V} x_{ij} = y_i & \forall\, i \in V \setminus \{t\} \\ & \sum_{i\in V} x_{ij} = y_j & \forall\, j \in V \setminus \{s\} \\ & y_s = 1, y_t = 1 & \\ & x_{ij} \in \{0,1\}, \lambda_{ij} \geq 0 & \forall\, ij \in A \\ & y_i \in \{0,1\}, \beta_i \geq 0 & \forall\, i \in V \\ & \alpha, \eta \geq 0 & \end{array} \tag{11}$$

Partie 2: Implémentation

Lors de l'implémentation, nous avons légèrement modifié la modélisation de notre partie théorique en définissant la variable x sur les arcs a des instances. On a ainsi $x[a] = 1$ si l'arc a est utilisé, 0 sinon. Cela permet en effet d'utiliser moins de variables et économise de la mémoire sur machine.

Nous avons également ajouté la contrainte que la somme des arcs entrants en s doit valoir 0 ainsi que la somme des arcs sortants de t. En effet, y compris dans le cas statique sans contrainte de poids, si l'on n'ajoute pas ces contraintes, on peut obtenir une solution qui n'est pas un chemin, comme sur la Fig. 1.

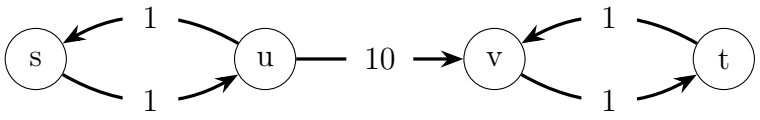

Figure 1: Cas justifiant l'interdiction d'arcs entrant en s ou sortant de t

L'exemple ci-dessus est trivial. Il n'y a qu'une seule solution au problème, à savoir le chemin (s, u, v, t) avec un score de 12. Cependant, si l'on ne force pas à ce qu'aucun arc n'entre dans s et qu'aucun arc ne sorte de t, la solution optimale est l'ensemble d'arcs $\{(s, u), (u, s), (v, t), (t, v)\}$ de score 4 mais qui n'est pas un chemin.

6 Plans coupants

La résolution par plans coupants est présentée dans le chapitre 1. Nous listons ci-dessous différentes astuces que nous avons mises en place afin d'accélérer sa résolution.

Résolution des sous-problèmes via knapsack continu Après chaque résolution du problème maître, les sous problèmes sont résolus à l'aide d'un *knapsack continu*. En effet, le problème maître renvoie un ensemble d'arcs sélectionnés (ainsi qu'un ensemble de ville traversées et une valeur objectif). Le sous-problème lié à l'objectif robuste (Pb. 4) consiste à sélectionner des $\delta_a^1 \in [0, D_a] \, \forall a \in$ solution de manière à maximiser $\sum_{a \in \text{solution}} d_a \delta_a^1$ avec $\sum_{a \in \text{solution}} \delta_a^1 \leq d_1$. On peut résoudre ce problème en rédigeant un PLNE. Cependant, on remarque que l'algorithme simple suivant renvoie la solution optimale.

On trie les arcs sélectionnés selon leur poids d_a puis on les parcourt dans l'ordre décroissant. Tant que l'on n'a pas dépassé le budget, on fixe les δ_a^1 à D_a si possible, ou bien à $d_1 - D_a$ pour le dernier arc sélectionné. On résout le sous-problème de contrainte de poids robuste (Pb. 5) de la même façon.

Coupes anti-symétries En plus de la résolution de ces sous-problèmes, nous avons ajouté des coupes qui permettent de réduire certaines symétries. Nous définissons de manière détaillée ces coupes dans la section 9. L'ajout de ces coupes nous a permis de clore certaines instances de taille moyennes de manière bien plus rapide, une dizaine de secondes alors que 120s sans enlever les symétries ne fermait pas.

Coupes pour accélerer borne inférieure Il est courant que CPLEX trouve rapidement la solution optimale et prenne ensuite du temps à prouver l'optimalité. Pour accélérer cette deuxième étape, on peut garder en mémoire la meilleure borne inférieure trouvée à l'itération précédente et ajouter la contrainte $z \geq$ meilleure borne inférieure. Cette inégalité est vraie puisqu'on ne fait qu'ajouter des contraintes.

Sauvegarde de la meilleure solution Un problème s'est présenté lors de la résolution avec les plans coupants. Si la résolution du problème maître renvoie une solution admissible vis à vis de la contrainte robuste de poids mais que la valeur de la solution était inférieure à son score robuste, alors la solution était rejetée car non admissible. Si l'on coupe la résolution trop vite ensuite, on peut se retrouver sans solution admissible alors qu'on avait un chemin admissible, ce qui n'est pas souhaitable. Ainsi, de manière manuelle, après chaque résolution du problème maître, on met à jour la meilleure solution admissible trouvée jusqu'à présent.

De plus, même si l'on ne trouve pas de solution qui respecte la contrainte robuste de poids des villes, il est intéressant de garder en mémoire la solution qui viole le moins possible cette contrainte plutôt que de renvoyer un inutile "Pas de solution admissible trouvée". Nous avons ajouté cette *feature* manuellement. Cela nous permet en particulier de conserver une borne inférieure, même si l'on n'a pas trouvé de solution admissible au problème.

WarmStart Pour accélérer la résolution du problème maitre, nous utilisons des WarmStarts qui permettent d'orienter la recherche. Il y a plusieurs manières de faire cela. La première est d'ajouter à chaque itération la meilleure solution trouvée jusqu'à présent et uniquement elle. La deuxième est d'ajouter à chaque itération toutes les meilleures solutions trouvées. La troisième est de créer l'objet $IloCplex$ en dehors des itérations et simplement d'ajouter progressivement les coupes au modèle. Ainsi, CPLEX reconnait qu'il a déjà résolu une variante très semblable au modèle actuel et gère lui même son WarmStart avec les solutions entières qu'il a trouvées précédemment. Empiriquement, c'est la dernière solution qui marche le mieux et que nous avons choisie.

Analyse des performances D'un point de vue code, la première version de la résolution des sous-problèmes prenait 40s en vérifiant la sélection des arcs via $cplex.getValue(x[a])$ pour chaque arc. Pour accélérer considérablement cette résolution, il suffit de stocker toutes les valeurs de x en une fois, via $cplex.getValues(xValues, x)$ pour obtenir le même résultat en un temps négligeable. En ordre de grandeur, l'écriture puis la résolution d'un PLNE pour une solution avec une vingtaine de sommets prend 1ms et la résolution du sous problème prend 1μs.

Au final, la résolution des sous problèmes et l'ajout des coupes de symétries se fait en quelques centièmes de seconde. C'est négligeable devant le temps mis par CPLEX a résoudre le problème maitre, qui peut prendre plusieurs dizaines de secondes pour les grosses instances.

7 Branch and cut

L'implémentation du Branch and Cut se fait de manière très similaire à celle des Plans coupants. Les astuces utilisées sont les mêmes, à savoir la résolution des sous-problèmes via des knapsack continus, l'ajout de coupes anti-symétries ainsi que la sauvegarde de la meilleure solution.

On implémente cela à l'intérieur d'un CallBack. Pour une grosse instance que l'on fait tourner plusieurs minutes, on atteint facilement une centaine d'appels au CallBack. Un CallBack prend quelques centièmes de seconde, ce qui est négligeable devant le temps total de résolution.

8 Heuristique

Nous avons choisi d'implémenter une heuristique. L'heuristique a été imaginée en ayant pour objectif d'être capable de retourner une solution admissible avec un score correct pour les grosses instances que le PLNE dualisé ne pouvait pas résoudre en temps raisonnable.

Pour cela, nous sommes repartis du problème de base qui est un problème de plus court chemin. Sans contrainte de poids des villes, l'algorithme de Dijkstra permet de trouver une solution optimale en temps polynomial quand les poids des arcs sont positifs. Avec un algorithme de Dijkstra "basique" ne prenant en compte que les distances d_{ij}, la solution renvoyée ne sera sûrement pas celle avec le coût robuste optimal, et ne sera sûrement pas admissible. Pour garder les performances des algorithmes polynomiaux tout en obtenant une solution admissible, nous avons passé la contrainte robuste en objectif, avec une pondération K. Le coût d'un sous-chemin devient donc:

$$\text{cout du chemin } s-u = \text{longueur robuste de } s-u + K \times \text{ poids robuste de } s-u$$

La particularité des sous problèmes est qu'il s'agit de problèmes de sac à dos. Pour calculer les longueur robustes lors de l'exploration du graphe, il suffit de garder en mémoire pour chaque nœud les poids dans le sac à dos associé au sous chemin menant au nœud courant. A chaque nœud, on associe les données suivantes, stocké dans un objet *node_ info*:

- le parent du nœud
- la distance $\sum d_{ij}$
- la distance robuste $\sum \delta^1_{ij} d_{ij}$
- le poids $\sum p_i$
- le poids robuste $\sum \delta^2_i \hat{p}_i$
- le contenu du sac à dos sous la forme $[\{d_{ij}, D_{ij}\}$ tq. $\delta^1_{ij} \neq 0$ ie. est dans le sac à dos$]$
- le contenu du sac à dos de poids $[\hat{p}_i$ tq. $\delta^2_i \neq 0$ ie. est dans le sac à dos$]$

Lorsqu'on explore un nœud v depuis un nœud u, on crée un nouvel objet *node_ info* à partir du *node_ info* de u. On met à jour la distance et le poids en ajoutant d_{uv} et p_u. La mise à jour des coûts robustes se fait facilement grâce à la structure de donnée. Pour la distance, si le sac à dos est plein (la somme des D_{ij} dans le sac est supérieure à d_1) et que d_{uv} est plus petit que le plus petit des d_{ij} dans le sac, alors on ne met rien à jour. Sinon, on ajoute le couple $\{d_{ij}, D_{ij}\}$ dans le sac à dos, on le trie par d_{ij} décroissant. Il suffit alors de *pop* les derniers éléments du vecteur jusqu'à retrouver une capacité adaptée au sac à dos (seul le dernier élément n'est pas un δ^1 égal à D_{ij}). A partir de là, il est aisé de recalculer la distance robuste. On procède de façon identique avec le poids, en sachant que les capacités des poids valent 2. Il suffit ensuite de comparer le coût actuel du nœud et le coût associé au nouveau *node_ info* créé, et d'attribuer le nouvel objet si le coût est meilleur.

Pour améliorer les performances de la méthode, on peut utiliser l'algorithme A^* à la place de l'algorithme de Dijkstra. On commence par réaliser une exploration du graphe à partir de t avec les distances statiques d_{ij} pour obtenir la distance statique minimale. On réalise une seconde exploration sur les poids statique p_i depuis t pour avoir le poids statique minimale. On obtient ainsi une borne inférieur pour A^*:

$$\text{borne inf de } u-t = \text{distance statique min } u-t + K \times \text{ poids statique min de } u-t$$

Pour chaque valeur positive de K, on peut résoudre efficacement le problème approché grâce à A^*. L'idée de notre heuristique est de jouer sur cette valeur de K pour essayer de trouver une solution optimale. En effet, si on choisit $K = 0$, l'algorithme ne va pas prendre en compte le poids du chemin. Au contraire, si on prends K très grand l'algorithme va chercher un chemin de poids minimal sans prendre en compte la longueur du chemin. L'intuition derrière l'heuristique est qu'il existe une valeur de K noté K^* telle que, si on note $val(K)$ la distance robuste de la solution $sol(K)$ renvoyée par l'heuristique avec la valeur K on a:

- $\forall K < K^*$ on a $sol(K)$ non admissible, et $val(K) < val(K^*)$
- $\forall K > K^*$ on a $sol(K)$ admissible, et $val(K) > val(K^*)$

Notre heuristique va donc instantier une solution pour $K_{inf} = 0$, puis trouver un K_{sup} tel que la solution est admissible (en cherchant par puissance de deux). On réalise ensuite une dichotomie sur K pour essayer de converger vers K^*, en mettant à jour K_{inf} et K_{sup} selon si $sol((K_{inf} + K_{sup})/2)$ est admissible ou non. Une illustration du processus est présenté dans la Fig. 2.

L'algorithme renvoie donc une solution admissible avec $sol(K_{sup})$ et une solution non admissible qui est parfois une borne inférieure avec $sol(K_{inf})$.

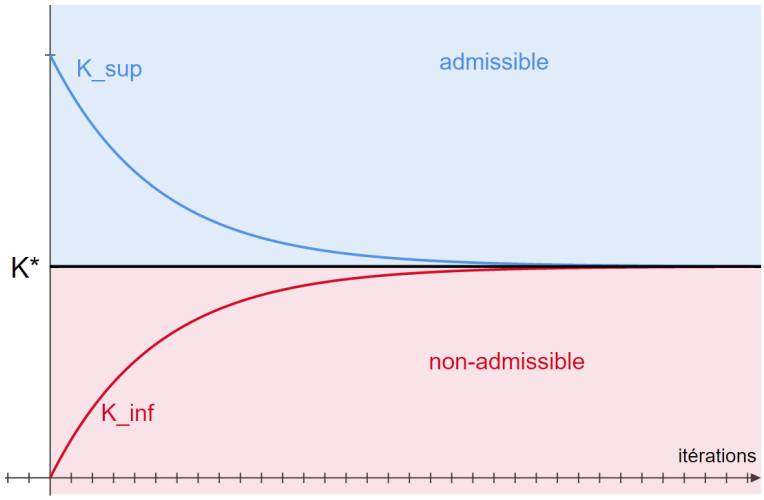

Figure 2: Convergence de K_{inf} et K_{sup} dans l'heuristique

Limites Notre heuristique possède un certain nombre de limites. La première limite théorique est qu'on calcule à chaque fois les problèmes de sac à dos sur le sous chemin entre s et le sommet courant, mais sans connaître le chemin entre le sommet courant et t. Ainsi le coût du chemin entre s et un sommet u change de valeur entre le moment où on explore u et lorsqu'on renvoie le chemin optimal. On perd donc la propriété que les sous-chemin du chemin optimal sont optimaux.

D'un point de vue pratique, nous avons observé plusieurs défauts. Premièrement, l'heuristique n'est pas adaptée aux instances pour lesquels on peut facilement trouver un chemin robustement admissible. En effet, si l'heuristique trouve un chemin admissible pour $K = 0$, la suite de l'heuristique perd son sens. Dans ces cas là, il semble que la limite théorique énoncée précédemment ait un fort impact.

Deuxièmement, l'heuristique renvoie une solution admissible, mais on ne possède aucune garantie sur la borne inférieure. En comparant les résultats de l'heuristique avec les résultats

de la méthode par dualisation, on observe que $sol(K_{inf})$ est potentiellement une borne inférieure pour les grosses instances.

Troisièmement, il existe des cas où l'algorithme n'arrive pas à instantier un chemin admissible, quelque soit la valeur de K. Dans ce cas, on se contente de renvoyer une solution admissible. On commence par résoudre un problème de plus court chemin sur les poids $p + \hat{p}$. Si cela n'est toujours pas suffisant, on résout un PLNE avec le coût statique et la contrainte robuste dualisée. Ce dernier cas n'est apparu que sur l'instance $1300 - BAY$.

Remarques Malgré ces quelques défauts, notre heuristique nous a surpris par son efficacité sur les instances pour lesquelles elle a été construite. Sur les grosses instances (plus de 2000 nœuds), l'heuristique renvoie son résultat en moins de 5 secondes. Sur ces instances, la solution admissible renvoyée est souvent identique voir meilleure que celle renvoyée par CPLEX au bout de plusieurs minutes quel que soit la méthode utilisée.

9 Réduction des symétries

Les sorties de CPLEX nous ont indiqué un grand nombre de symétries dans nos problèmes. Par ailleurs, en nous penchant sur les solutions successives de la méthode de plans coupants, nous nous sommes aperçus que lorsqu'une solution était coupée, la solution suivante était souvent très proche, sans qu'aucune amélioration ne soit faite sur la valeur du score ou de la contrainte robuste. Cela rendait particulièrement inefficace la résolution par plans coupants car le problème maître est résolu à la racine à chaque nouvelle itération. Pour ne pas itérer sur des solutions similaires, nous avons choisi d'étudier le cas présenté dans la Fig. 3.

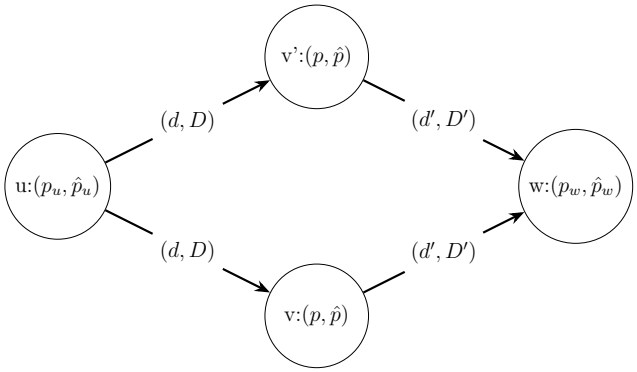

Figure 3: Sous-chemins entre u et w

Sur la figure ci-dessus, on remarque que les sous chemins $(u - v - w)$ et $(u - v' - w)$ sont équivalents, puisque les valeurs de tous les paramètres sont identiques. Ainsi, tout chemin contenant le sous chemin (u, v', w) ne sera pas meilleur que les chemin contenant $(u - v - w)$ (tout les autres noeuds étant égaux).

Pour éviter que les algorithmes de Branch-and-cut et Plans coupants ne proposent successivement deux chemin similaires différents uniquement sur le sommet v/v', on interdit la pair d'arc $uv' + v'w$. En pratique on laisse seulement le sous-chemin avec le sommet intermédiaire d'indice minimum. On réalise cette interdiction en ajoutant une contrainte dans le modèle:

$$x_{uv'} + x_{v'w} \leq 1$$

Ainsi, on n'interdit pas les arcs concernés, mais seulement le sous-chemin. On remarque qu'on peut également supprimer le sous-chemin si les couples (d, D) et (d', D') sont inversés entre les deux sous-chemins.

Nous avons essayé de réaliser ces interdictions en pré-traitement en explorant l'ensemble des couples de sommets (u, w), mais cela représente trop de contraintes ajoutées et cela ralentit très fortement le calcul. Cette méthode n'a donc pas été implémenté dans la résolution du problème dualisé.

Pour éviter cela, on va réaliser les interdictions dans le callback: pour tout triplet de sommet (u, v, w) dans la solution courante, on va ajouter les contraintes relatives au sous-chemins entre u et w. Sur une petite instance tel que $200 - COL$, avec la méthode de branch and cut, on ne résolvait pas l'instance en $500s$ avant d'implémenter cette réduction. Avec la réduction, l'instance est résolue en $45s$.

On pourrait élaborer davantage en généralisant la démarche aux sous-chemins de taille 3 et plus. On pourrait également interdire les sous-chemins qui dégradent strictement la solution. Nous avons choisi de ne pas implémenter ces options pour ne pas surcharger les modèles donnés à CPLEX.

10 Résultats

Les différentes méthodes ont été implémentées en C++ 11.4.0 et avec CPLEX 22.1.1. Le benchmark a été réalisé à partir de 123 instances issues du 9^e challenge DIMACS avec entre 20 et 2500 sommets. Le temps de résolution maximum du Branch-and-Cut et du Plans coupants est fixé à $500s$ et celui de la méthode par dualisation à $1200s$ car c'était empiriquement la méthode qui était la plus efficace pour obtenir une borne inférieure. Nous avons ainsi pu fermer 93 des 123 instances proposées. Les gaps sont calculés en fonction de la meilleure borne inférieure trouvée, souvent avec la résolution par dualisation, de la manière suivante:

$$\text{gap} = 100 \times \frac{\text{objectif robuste} - \text{meilleure borne inf}}{\text{objectif robuste}}$$

On observe sur la Fig 4 que la résolution par dualisation est de loin la méthode la plus performante pour fermer les instances, suivie du branch and cut et enfin de plans coupants. Les pics des méthodes Plans coupants et Branch-and-cut après 500s sont dûs au fait que leur résolution s'est arrêtée car leur limite de temps a été atteinte mais que le gap n'était pas encore à 0. Cependant, comme dans ce graphique, nous avons calculé le gap en fonction de la meilleure borne inférieure, il se trouve qu'ils avaient déjà trouvé la solution optimale sans prouver l'optimalité. Il serait intéressant d'étudier le temps où les différentes méthodes trouvent leur meilleure solution.

Sur la Fig. 5, on représente le pourcentage d'instances résolues avec un gap inférieur à l'abscisse. Le saut à droite pour le gap de 100% correspond aux instances non-admissibles, dont on a fixé le score à 10^9.

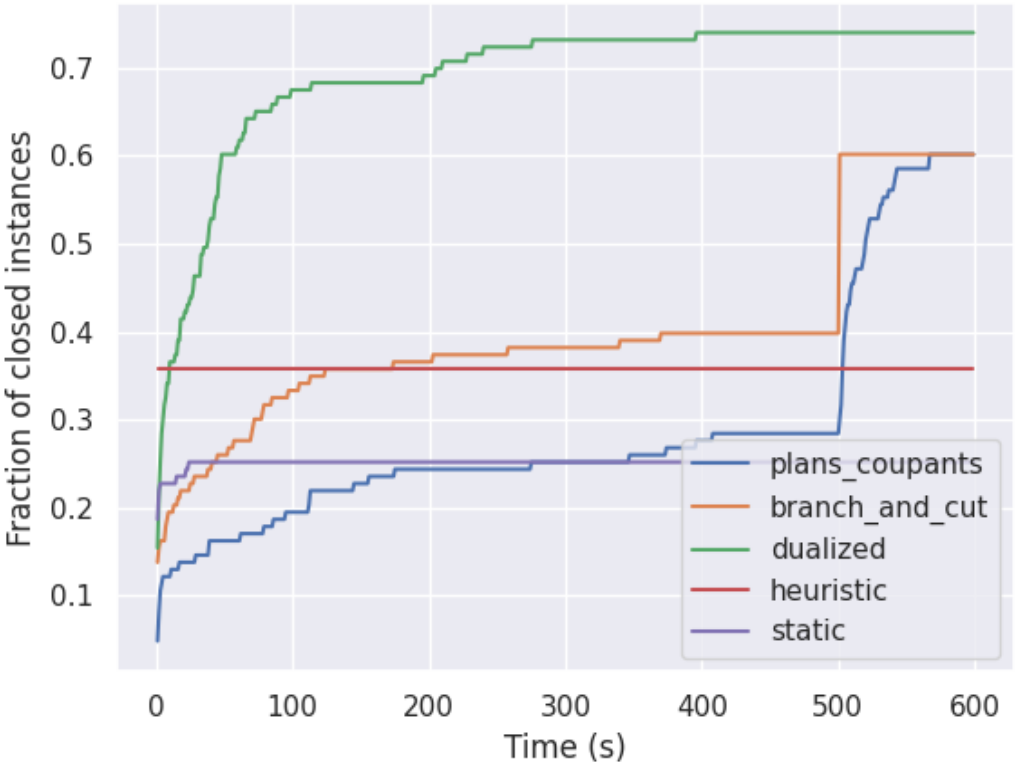

Figure 4: Fraction d'instances fermées par les différentes méthodes

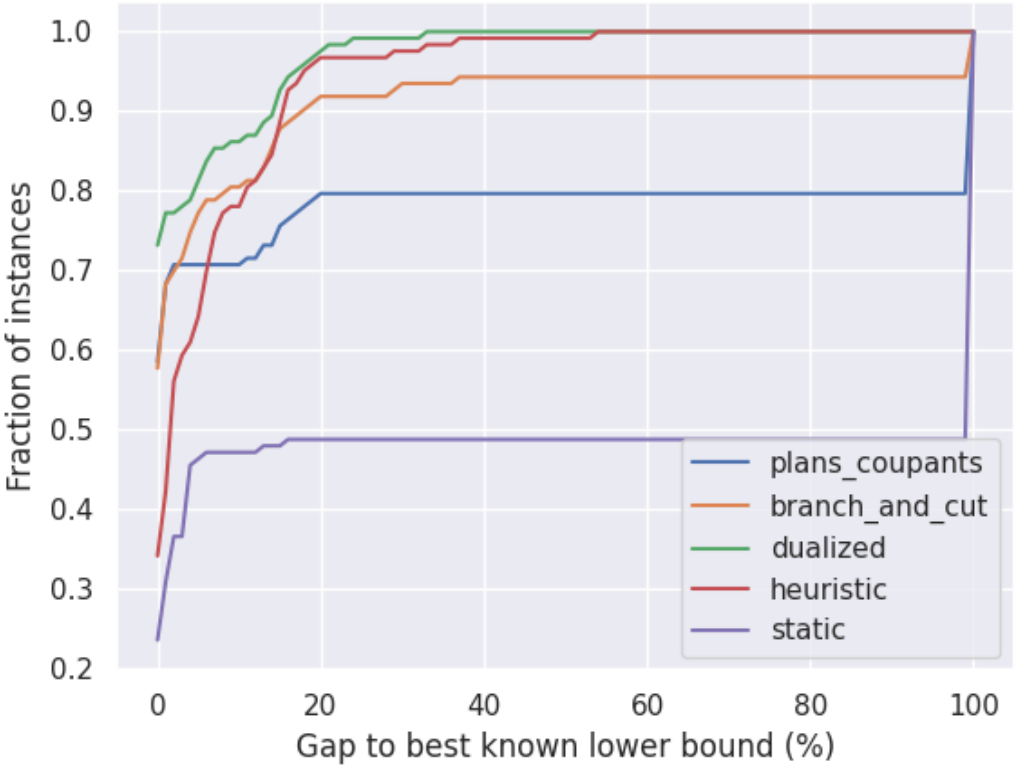

Figure 5: Fraction d'instances résolues avec un gap inférieur à l'abscisse par les différentes méthodes

Sur la Fig. 5, on observe que bien que l'heuristique ne ferme pas beaucoup d'instances (ordonnée à l'origine), son gap est généralement faible et qu'elle est compétitive contre les autres méthodes, d'autant plus que son exécution ne prend au plus que quelques secondes. Son principal inconvénient est qu'elle ne renvoie pas de garantie.

Les résultats pour chaque instance dans le détail sont présentés en Annexe dans le Tab. 1. Si une méthode n'a pas trouvé de solution admissible dans le temps imparti, nous avons fixé son objectif à 10^9. Tous les instances statiques se résolvent en moins de 2 minutes. Etant donné que les instances (robustes) ne sont pas toutes fermées, nous donnons un encadrement du prix de la robustesse, qui est défini de la manière suivante:

$$\mathrm{PR}_{\mathrm{inf}} = 100 \times \frac{\text{meilleure borne inférieure robuste} - \text{score optimal statique}}{\text{score optimal statique}}$$

$$\mathrm{PR}_{\mathrm{sup}} = 100 \times \frac{\text{meilleure solution robuste} - \text{score optimal statique}}{\text{score optimal statique}}$$

La meilleure solution que nous avons pour chaque instance est présentée en Annexe Tab. 2.

11 Ouvertures

Pour continuer ce projet et chercher à améliorer les résultats, voici les pistes de réflexion que nous avons analysées:

- Étendre la cassure des symétries aux triplets/quadruplets de sommets et étudier son impact sur la résolution en Branch-and-cut et Plans coupants
- Étudier de manière précise pourquoi le modèle par dualisation ressort un nombre très important de symétries (1e350).
- Utiliser un WarmStart à l'aide de l'heuristique afin d'accélérer la résolution de la méthode par dualisation et essayer de fermer les instances restantes

Par ailleurs, bien que ce ne soit pas une amélioration à proprement parler, il serait intéressant d'analyser le temps où les modèles ont trouvé leur meilleure solution. En effet, on observe souvent que la meilleure solution est trouvée assez rapidement mais que le modèle continue de tourner pour augmenter la borne inférieure et fermer l'instance.

Pour finir, lors de l'implémentation du branch-and-cut, nous avons repéré une incohérence de CPLEX que nous détaillons en Annexe dans la section C.

Annexes

A Présentation des résultats des différentes méthodes

instance	PR		plans coupants		branch-and-cut		dualized		heuristic	
	inf	sup	obj.	gap	obj.	gap	obj.	gap	obj.	gap
20 NY	27.57	27.57	9454.47	0.00	9454.47	0.00	9454.47	0.00	9454.47	0.00
20 BAY	38.92	38.92	15332.60	0.00	15332.60	0.00	15332.60	0.00	15656.50	2.07
20 COL	24.30	24.30	7076.52	0.00	7076.52	0.00	7076.52	0.00	7076.52	0.00
40 NY	28.39	28.39	17330.10	0.00	17330.10	0.00	17330.10	0.00	17330.10	0.00
40 COL	33.004	33.004	15059.00	0.00	15059.00	0.00	15059.00	0.00	15059.00	0.00
40 BAY	37.97	37.97	12664.30	0.00	12664.30	0.00	12664.30	0.00	12664.30	0.00
60 COL	29.31	29.31	23914.20	0.00	23914.20	0.00	23914.20	0.00	23914.20	0.00
60 BAY	34.50	34.50	10633.30	0.00	10633.30	0.00	10633.30	0.00	10633.30	0.00
60 NY	61.47	61.47	31775.80	0.00	31775.80	0.00	31775.80	0.00	31775.80	0.00
80 NY	61.47	61.47	31775.80	0.00	31775.80	0.00	31775.80	0.00	31775.80	0.00
80 COL	23.71	23.71	14277.50	0.00	14277.50	0.00	14277.50	0.00	22378.50	36.20
80 BAY	18.78	18.78	10857.10	0.00	10857.10	0.00	10857.10	0.00	12880.30	15.71
100 NY	54.99	54.99	33931.00	0.00	33931.00	0.00	33931.00	0.00	33931.00	0.00
100 COL	25.24	25.24	25320.20	0.00	25320.20	0.00	25320.10	0.00	25320.20	0.00
100 BAY	18.78	18.78	10857.10	0.00	10857.10	0.00	10857.10	0.00	10857.10	0.00
120 BAY	17.88	17.88	12219.00	0.00	12219.00	0.00	12219.00	0.00	12219.00	0.00
120 COL	26.00	26.00	25582.60	0.00	25582.60	0.00	25582.60	0.00	31065.50	17.65
120 NY	38.21	38.21	30613.10	0.00	30613.10	0.00	30613.10	0.00	31592.00	3.10
140 NY	22.92	22.92	33079.20	0.00	33079.20	0.00	33079.20	0.00	39267.80	15.76
140 COL	25.76	25.76	24651.00	0.00	24651.00	0.00	24651.00	0.00	27650.90	10.85
140 BAY	22.21	22.21	15643.70	0.00	15643.70	0.00	15643.70	0.00	15643.70	0.00
160 BAY	18.01	18.01	13395.70	0.00	13395.70	0.00	13395.70	0.00	13395.70	0.00
160 COL	25.76	25.76	24651.00	0.00	24651.00	0.00	24651.00	0.00	27650.90	10.85
160 NY	36.94	36.94	31661.10	0.00	31661.10	0.00	31661.10	0.00	32640.00	3.00
180 NY	35.08	35.08	30750.70	0.00	30750.70	0.00	30750.70	0.00	32640.00	5.79
180 COL	32.23	32.23	33704.60	0.00	33704.60	0.00	33704.60	0.00	36557.50	7.80
180 BAY	18.01	18.01	13395.70	0.00	13395.70	0.00	13395.70	0.00	13395.70	0.00
200 COL	24.99	24.99	31845.70	0.00	31845.70	0.00	31845.70	0.00	34466.70	7.60
200 BAY	18.01	18.01	13395.70	0.00	13395.70	0.00	13395.70	0.00	14081.60	4.87
200 NY	19.50	19.50	30595.10	0.00	30595.10	0.00	30595.10	0.00	30595.10	0.00
250 BAY	25.98	25.98	19248.00	0.00	19248.00	0.00	19248.00	0.00	19248.00	0.00
250 NY	21.38	21.38	37057.40	0.00	37057.40	0.00	37057.40	0.00	37057.40	0.00
250 COL	24.13	24.13	32794.30	0.00	32794.30	0.00	32794.30	0.00	34677.40	5.43
300 BAY	24.58	24.58	21954.20	0.00	21954.20	0.00	21954.20	0.00	21954.20	0.00
300 COL	22.87	22.87	34605.30	0.00	34605.30	0.00	34605.30	0.00	36488.40	5.16
300 NY	20.94	20.94	35802.80	0.00	35802.80	0.00	35802.80	0.00	35802.80	0.00
350 BAY	24.58	24.58	21954.20	0.00	21954.20	0.00	21954.20	0.00	21954.20	0.00
350 COL	22.01	22.01	30611.00	0.00	30611.00	0.00	30611.00	0.00	30611.00	0.00
350 NY	18.29	18.29	41712.40	0.00	41712.40	0.00	41712.40	0.00	45262.90	7.84
400 BAY	19.04	19.04	32288.40	0.00	32288.40	0.00	32288.40	0.00	32288.40	0.00
400 NY	26.96	26.96	44441.90	0.00	44441.90	0.00	44441.90	0.00	45262.90	1.81
400 COL	21.62	21.62	39224.00	0.00	40704.50	3.64	39224.00	0.00	39911.20	1.72
450 NY	16.22	16.22	47080.70	0.00	47080.70	0.00	47080.70	0.00	47080.70	0.00
450 COL	21.62	21.62	39224.00	0.00	40704.50	3.64	39224.00	0.00	39911.20	1.72
450 BAY	19.04	19.04	32288.40	0.00	32288.40	0.00	32288.40	0.00	32288.40	0.00

instance	PR		plans coupants		branch-and-cut		dualized		heuristic	
	inf	sup	obj.	gap	obj.	gap	obj.	gap	obj.	gap
500 BAY	26.10	26.10	38097.90	0.00	38097.90	0.00	38097.90	0.00	38097.90	0.00
500 NY	17.41	17.41	48093.40	0.00	48093.40	0.00	48093.40	0.00	48093.40	0.00
500 COL	21.62	21.62	39224.00	0.00	39653.80	1.08	39224.00	0.00	39911.20	1.72
550 NY	16.93	16.93	47814.40	0.00	47814.40	0.00	47814.40	0.00	47814.40	0.00
550 BAY	21.01	21.01	22317.20	0.00	22317.20	0.00	22317.20	0.00	22317.20	0.00
550 COL	21.62	21.62	39605.00	0.96	39224.00	0.00	39224.00	0.00	39911.20	1.72
600 BAY	17.37	17.37	34634.20	0.00	54388.10	36.32	34634.20	0.00	34849.90	0.62
600 COL	21.62	21.62	39605.00	0.96	39224.00	0.00	39224.00	0.00	39911.20	1.72
600 NY	16.93	16.93	47814.40	0.00	47814.40	0.00	47814.40	0.00	47814.40	0.00
650 BAY	17.37	17.37	34634.20	0.00	34849.90	0.62	34634.20	0.00	34634.20	0.00
650 COL	21.62	21.62	39605.00	0.96	39911.20	1.72	39224.00	0.00	39911.20	1.72
650 NY	16.34	16.34	43329.00	0.00	43329.00	0.00	43329.10	0.00	43841.30	1.17
700 NY	16.34	16.34	43329.00	0.00	43329.00	0.00	43329.10	0.00	43841.30	1.17
700 BAY	17.19	17.19	46578.90	0.76	48015.30	3.72	46226.90	0.00	49188.90	6.02
700 COL	21.62	21.62	39653.80	1.08	39224.00	0.00	39224.00	0.00	39911.20	1.72
750 NY	17.58	17.58	43624.90	0.00	43624.90	0.00	43624.90	0.00	43624.90	0.00
750 BAY	17.19	17.19	46226.90	0.00	48015.30	3.72	46226.90	0.00	49188.90	6.02
750 COL	21.62	21.62	39653.80	1.08	40083.80	2.15	39224.00	0.00	39911.20	1.72
800 NY	17.58	17.58	43624.90	0.00	43624.90	0.00	43624.90	0.00	43624.90	0.00
800 COL	21.62	21.62	39653.80	1.08	41102.30	4.57	39224.00	0.00	40845.50	3.97
800 BAY	17.79	17.79	40198.40	0.00	40198.40	0.00	40198.40	0.00	48081.10	16.39
850 COL	18.89	18.89	29904.80	0.78	29904.80	0.78	29672.80	0.00	29904.80	0.78
850 BAY	16.71	16.71	42201.10	0.00	42201.10	0.00	42201.10	0.00	42707.00	1.18
850 NY	16.34	16.34	43329.00	0.00	43329.00	0.00	43329.10	0.00	43841.30	1.17
900 BAY	16.33	16.33	54274.20	0.006	54274.20	0.006	54240.20	0.00	61420.20	11.69
900 COL	18.89	18.89	29904.80	0.78	29904.80	0.78	29672.80	0.00	29904.80	0.78
900 NY	16.34	16.34	43329.00	0.00	43329.00	0.00	43329.10	0.00	43841.30	1.17
950 COL	18.89	18.89	29904.80	0.78	29904.80	0.78	29672.80	0.00	29904.80	0.78
950 BAY	16.33	16.33	54274.20	0.006	54274.20	0.006	54240.20	0.00	57297.60	5.34
950 NY	16.34	16.34	43329.00	0.00	43329.00	0.00	43329.10	0.00	43841.30	1.17
1000 BAY	16.86	16.86	48250.00	0.00	48250.00	0.00	48250.00	0.00	48738.70	1.00
1000 COL	18.89	18.89	29904.80	0.78	29904.80	0.78	29672.80	0.00	29904.80	0.78
1000 NY	16.34	16.34	43329.00	0.00	43329.00	0.00	43329.10	0.00	43841.30	1.17
1100 BAY	22.88	22.88	1e9	100.00	1e9	100.00	70226.30	0.00	83140.00	15.53
1100 COL	9.05	20.32	42766.50	12.40	42766.50	12.40	42766.50	12.40	42766.50	12.40
1100 NY	29.58	29.59	1e9	100.00	70089.20	0.001	70089.20	0.001	71656.20	2.20
1200 NY	17.88	17.88	72566.00	0.00	72566.00	0.00	72566.00	0.00	72566.00	0.00
1200 BAY	18.66	18.66	1e9	100.00	76338.20	0.00	76338.20	0.00	76338.20	0.00
1200 COL	9.00	20.32	42766.50	12.44	42766.50	12.44	42766.50	12.44	42766.50	12.44
1300 COL	10.30	23.15	42505.70	14.33	42505.70	14.33	42505.70	14.33	42505.70	14.33
1300 NY	18.17	18.17	67883.90	0.00	67883.90	0.00	67883.90	0.00	67883.90	0.00
1300 BAY	21.28	21.28	1e9	100.00	1e9	100.00	76715.80	0.00	77217.70	0.65
1400 COL	13.98	23.15	42505.70	10.67	42505.70	10.67	42505.70	10.67	42505.70	10.67
1400 BAY	15.36	15.36	1e9	100.00	79995.30	0.00	79995.30	0.00	79995.30	0.00
1400 NY	18.26	18.26	67883.90	0.00	67883.90	0.00	67883.90	0.00	67883.90	0.00
1500 BAY	11.31	15.36	1e9	100.00	79379.70	4.57	79379.70	4.57	79379.70	4.57
1500 COL	9.79	23.15	42505.70	14.81	42505.70	14.81	42505.70	14.81	42505.70	14.81
1500 NY	18.90	18.90	66475.50	0.00	66475.50	0.00	66475.50	0.00	66475.50	0.00
1600 BAY	10.52	14.69	1e9	100.00	81914.40	4.79	81809.70	4.66	81914.40	4.79
1600 COL	6.75	19.83	42505.70	14.03	42505.70	14.03	42505.70	14.03	42505.70	14.03
1600 NY	18.99	18.99	66475.50	0.00	66475.50	0.00	66475.50	0.00	66475.50	0.00
1700 BAY	12.09	17.22	1e9	100.00	70557.80	5.84	70557.80	5.84	70557.80	5.84
1700 NY	18.99	18.99	66475.50	0.00	66475.50	0.00	66475.50	0.00	66475.50	0.00
1700 COL	4.92	23.15	42505.70	19.18	42505.70	19.18	42505.70	19.18	42505.70	19.18

instance	PR		plans coupants		branch-and-cut		dualized		heuristic	
	inf	sup	obj.	gap	obj.	gap	obj.	gap	obj.	gap
1800 COL	4.96	19.83	1e9	100.00	1e9	100.00	42505.70	15.65	42505.7	15.65
1800 NY	18.26	18.26	69098.3	0.00	69098.30	0.00	69098.30	0.00	69098.3	0.00
1800 BAY	9.11	16.71	1e9	100.00	72553.70	8.95	72086.60	8.36	72511.8	8.9
1900 COL	5.50	19.83	42505.7	15.17	42505.70	15.17	42505.70	15.17	42505.7	15.17
1900 NY	18.26	18.26	69098.3	0.00	69098.30	0.00	69098.20	0.00	69098.3	0.00
1900 BAY	13.59	15.68	1e9	100.00	77036.70	2.81	76729.60	2.42	77036.7	2.81
2000 BAY	16.04	16.37	1e9	100.00	67604.0	0.40	67604.0	0.40	67604.0	0.4
2000 COL	5.35	19.54	50172.5	16.95	50172.50	16.95	49019.10	14.99	49019.1	14.99
2000 NY	21.43	26.21	1e9	100.00	72476.80	13.37	66853.30	6.09	66853.3	6.09
2100 BAY	10.73	15.86	1e9	100.00	72250.20	5.76	72250.20	5.76	72250.2	5.76
2100 COL	0.36	19.16	43998.7	18.87	43998.70	18.87	43998.70	18.87	43998.7	18.87
2100 NY	49.6	49.6	123257.0	0.00	123257.0	0.00	123257.0	0.00	123989.0	0.59
2200 BAY	10.43	15.07	1e9	100.00	1e9	100.00	80585.30	5.17	80585.3	5.17
2200 COL	7.18	19.71	1e9	100.00	48962.90	13.5	48962.90	13.5	48962.9	13.50
2200 NY	7.23	25.98	1e9	100.00	1e9	100.00	72476.80	20.2	123989.0	53.36
2300 COL	1.66	18.67	50172.5	17.30	50172.50	17.3	50172.50	17.3	50172.5	17.30
2300 BAY	11.97	14.72	1e9	100.00	104070.00	28.14	77192.10	3.12	104070.00	28.14
2300 NY	35.68	56.47	1e9	100.00	1e9	100.00	123257.0	32.33	123257.0	32.33
2400 BAY	11.23	15.38	1e9	100.00	104070.00	29.29	77192.10	4.67	77192.1	4.67
2400 COL	6.38	19.54	1e9	100.00	1e9	100.00	50172.50	16.03	49019.1	14.05
2400 NY	15.5	15.5	1e9	100.00	71974.30	0.00	71974.30	0.00	76784.1	6.26
2500 BAY	10.005	16.07	1e9	100.00	74364.20	7.28	73897.10	6.70	73974.2	6.79
2500 NY	15.31	15.5	1e9	100.00	71974.30	0.22	71974.30	0.22	76784.1	6.47
2500 COL	5.15	18.03	1e9	100.00	48427.80	13.58	54468.60	23.17	48427.8	13.58

Table 1: Présentation pour chaque instance du prix de la robustesse, du gap et de la valeur de l'objectif obtenue pour chaque méthode

B Présentation de la meilleure solution pour chaque instance

instance	method	objective	lower_bound	path
20 NY	dualized	9454.47	9454.47	[2;8;7;9]
20 BAY	dualized	15332.6	15332.6	[15;11;1;20;17]
20 COL	dualized	7076.52	7076.52	[12;2;14;6;15]
40 NY	dualized	17330.1	17330.1	[24;33;8;38;15;39]
40 COL	dualized	15059.0	15059.0	[24;36;13;33;28;27]
40 BAY	dualized	12664.3	12664.3	[29;21;25;30;34;40]
60 COL	dualized	23914.2	23914.2	[53;52;56;58;51;44;43;28;27;60]
60 BAY	dualized	10633.3	10633.3	[21;39;42;34;44;59]
60 NY	dualized	31775.8	31775.8	[39;60;7;46;47;59;56]
80 NY	dualized	31775.8	31775.8	[39;60;63;66;47;44;56]
80 COL	dualized	14277.5	14277.5	[46;55;10;14;69;71;78;76]
80 BAY	dualized	10857.1	10857.1	[59;44;10;30;65;66;74]
100 NY	dualized	33931.0	33931.0	[51;61;39;60;17;66;47;78;96]
100 COL	dualized	25320.1	25320.1	[81;63;90;5;74;79;58;93;76;49;71;83]
100 BAY	dualized	10857.1	10857.1	[59;44;48;4;38;21;74]
120 BAY	dualized	12219.0	12219.0	[74;21;25;30;10;79;100;105]
120 COL	dualized	25582.6	25582.6	[81;59;62;21;30;29;58;93;97;78;71;83]
120 NY	dualized	30613.1	30613.1	[110;91;27;71;83;76;29;54;120]
140 NY	dualized	33079.2	33079.2	[134;98;114;97;44;100;87;63;60;54;120;140]
140 COL	dualized	24651.0	24651.0	[60;83;117;78;97;138;106;127;126;82;129]
140 BAY	dualized	15643.7	15643.7	[63;58;71;75;40;116;105;84;136]
160 BAY	dualized	13395.7	13395.7	[74;89;38;104;90;124;59;105;159]
160 COL	dualized	24651.0	24651.0	[60;83;71;49;76;93;58;101;126;143;129]
160 NY	dualized	31661.1	31661.1	[120;54;29;76;28;2;143;91;142;154]
180 NY	dualized	30750.7	30750.7	[120;122;127;51;157;161;156;26;3;2;129;133;142;154]
180 COL	dualized	33704.6	33704.6	[83;117;49;76;93;137;56;52;158;156;59;159;161]
180 BAY	dualized	13395.7	13395.7	[74;21;178;163;48;44;100;138;159]
200 COL	dualized	31845.7	31845.7	[161;159;59;26;5;52;56;137;184;148;49;71;83;196]
200 BAY	dualized	13395.7	13395.7	[74;89;103;30;48;44;167;164;159]
200 NY	dualized	30595.1	30595.1	[137;125;154;194;103;135;148;24;117;120;149]
250 BAY	dualized	19248.0	19248.0	[159;105;209;191;186;185;77;199;222;224]
250 NY	dualized	37057.4	37057.4	[147;82;172;184;197;198;191;187;108;240;199;228;204;239]

250 COL	dualized	32794.3	32794.3	[83;167;45;65;46;12;2;47;239;25;113;192;161]
300 BAY	dualized	21954.2	21954.2	[224;222;199;172;185;133;276;280;253;260;269]
300 COL	dualized	34605.3	34605.3	[83;60;110;217;46;86;219;121;145;25;222;261;161;260]
300 NY	dualized	35802.8	35802.8	[239;267;293;264;240;262;263;19;174;85;54;212;245;279]
350 BAY	dualized	21954.2	21954.2	[224;330;199;316;185;281;276;348;253;160;269]
350 COL	dualized	30611.0	30611.0	[215;308;273;71;27;9;33;34;67;24;270;302;347]
350 NY	dualized	41712.4	41712.4	[242;251;250;117;62;334;3;26;262;336;199;152;294;205;137;248;270;349]
400 BAY	dualized	32288.4	32288.4	[269;160;289;130;355;287;393;358;383;371;230;223;374;382]
400 NY	dualized	44441.9	44441.9	[242;227;366;112;50;183;26;262;336;353;361;351;151;347;305;349]
400 COL	dualized	39224.0	39224.0	[83;60;45;65;267;375;397;23;239;134;222;261;281;260;389]
450 NY	dualized	47080.7	47080.7	[279;443;381;226;301;69;289;187;363;240;353;361;351;151;450;373;354;431]
450 COL	dualized	39224.0	39224.0	[83;60;45;312;46;444;397;313;239;336;113;301;281;260;389]
450 BAY	dualized	32288.4	32288.4	[269;440;268;325;418;287;393;358;224;215;119;267;422;382]
500 BAY	dualized	38097.9	38097.9	[466;488;160;164;456;476;287;393;433;224;109;498;481]
500 NY	dualized	48093.4	48093.4	[463;461;434;313;351;451;353;336;108;364;453;50;62;402;381;490;393;280;488]
500 COL	dualized	39224.0	39224.0	[83;310;45;217;46;12;219;457;114;25;222;192;281;352;389]
550 NY	dualized	47814.4	47814.4	[463;461;434;360;406;335;439;407;363;109;453;174;197;226;481;490;516;458;488]
550 BAY	dualized	22317.2	22317.2	[413;426;471;541;65;538;549;32;256;475;476;493;373;537]
550 COL	dualized	39224.0	39224.0	[83;259;64;65;360;538;10;8;37;25;222;301;346;260;389]
600 BAY	dualized	34634.2	34634.2	[269;284;105;209;290;586;531;433;535;478;212;240;587;567;588]
600 COL	dualized	39224.0	39224.0	[83;567;45;552;46;86;41;1;453;527;317;261;211;352;389]
600 NY	dualized	47814.4	47814.4	[463;461;549;313;568;361;353;240;363;109;243;69;197;402;586;281;393;458;488]
650 BAY	dualized	34634.2	34634.2	[269;608;591;623;191;493;393;540;497;215;625;240;587;642;588]
650 COL	dualized	39224.0	39224.0	[83;84;311;65;85;582;16;34;616;621;317;301;281;601;389]
650 NY	dualized	43329.1	43329.1	[576;385;382;539;456;320;444;360;396;392;415;128;106;105;313;120;348;620;478;621]
700 NY	dualized	43329.1	43329.1	[576;595;382;513;151;496;324;581;396;641;26;3;50;62;402;435;348;349;646;621]
700 BAY	dualized	46226.9	46226.9	[664;669;630;615;625;109;570;540;531;586;611;184;428;608;488;674;524;677]
700 COL	dualized	39224.0	39224.0	[83;167;45;312;581;699;10;619;453;527;222;192;346;352;389]
750 NY	dualized	43624.9	43624.9	[621;646;349;348;740;313;709;522;84;281;641;240;715;324;695;713;578;691;707]
750 BAY	dualized	46226.9	46227.0	[664;534;663;267;661;109;445;433;393;493;418;698;741;160;733;674;734;677]
750 COL	dualized	39224.0	39224.0	[83;480;64;65;581;86;87;121;68;134;584;301;161;601;389]
800 NY	dualized	43624.9	43624.9	[621;478;349;348;268;313;555;198;587;26;108;396;581;324;695;443;578;758;707]
800 COL	dualized	39224.0	39224.0	[83;167;45;650;85;283;2;766;199;728;113;301;161;260;389]

800 BAY	dualized	40198.4	40198.4	[588;567;394;432;492;510;782;433;531;381;355;508;479;748;766]
850 COL	dualized	29672.8	29672.8	[213;188;526;499;22;791;39;1;20;745;117;724;576;666]
850 BAY	dualized	42201.1	42201.1	[659;656;642;628;717;785;625;456;595;424;516;165;783;817;698;786]
850 NY	dualized	43329.1	43329.1	[576;595;596;539;592;651;640;581;696;838;766;19;2;85;465;268;256;620;478;621]
900 BAY	dualized	54240.2	54240.2	[820;659;878;755;734;896;216;873;670;559;875;731;165;109;177;666;773;844;854]
900 COL	dualized	29672.8	29672.8	[213;188;596;499;839;897;735;890;158;577;117;724;710;666]
900 NY	dualized	43329.1	43329.1	[576;595;382;678;151;359;444;581;582;447;109;787;552;555;845;268;348;349;759;621]
950 COL	dualized	29672.8	29672.8	[213;188;162;465;726;139;433;816;20;512;280;724;710;666]
950 BAY	dualized	54240.2	54240.2	[820;919;878;642;628;348;452;112;600;438;875;943;335;109;177;371;432;746;854]
950 NY	dualized	43329.1	43329.1	[576;595;382;708;364;320;444;581;582;641;415;800;182;901;687;268;619;349;646;621]
1000 BAY	dualized	48250.0	48250.0	[659;878;578;233;206;313;205;996;331;424;330;310;395;994;966;965;993]
1000 COL	dualized	29672.8	29672.8	[213;188;230;70;22;559;873;1;20;964;864;319;576;666]
1000 NY	dualized	43329.1	43329.1	[576;707;689;678;677;651;923;916;325;838;415;102;18;105;117;995;256;620;759;621]
1100 BAY	dualized	70226.3	70226.3	[820;910;541;1053;316;758;658;457;559;424;926;1045;859;454;975;786;990;890;1038;1082;1099]
1100 COL	dualized	42766.5	37464.3	[88;189;1033;467;904;568;484;628;787;185;383;1030;696;529;1053]
1100 NY	dualized	70089.2	70082.2	[621;759;620;256;829;149;589;584;1093;855;866;697;326;447;696;581;324;695;655;959;1095;1066]
1200 NY	dualized	72566.0	72566.0	[1066;1095;578;938;239;1153;360;696;447;856;697;866;855;1044;1070;1117;617;864;517;646; 1118;1144]
1200 BAY	dualized	76338.2	76338.2	[1099;1082;1115;890;1127;1046;698;928;859;989;516;888;559;936;861;902;1159;860;900;970;792;826; 1177]
1200 COL	dualized	42766.5	37445.0	[88;189;213;518;687;967;406;359;210;1004;150;945;554;529;1053]
1300 COL	dualized	42505.7	36412.7	[88;473;331;1237;172;1288;861;181;210;171;1232;1178;554;529;1053]
1300 NY	dualized	67883.9	67883.9	[1144;621;1054;875;477;1254;1117;1070;1093;1068;866;697;856;108;240;581;640;579;1235;1223;959; 691;1271;1296]
1300 BAY	dualized	76715.8	76715.8	[1099;1150;1115;1018;1127;1116;1084;1288;1273;1158;520;989;414;424;1266;936;1160;1237;671;938; 994;1093;939;826;1177]
1400 COL	dualized	42505.7	37971.9	[88;189;213;518;401;1386;406;1059;955;171;1232;528;479;1001;1053]
1400 BAY	dualized	79995.3	79995.3	[1177;1161;792;1370;502;860;503;1237;861;581;1148;963;1016;1267;1236;993;1332;1288;1361;1153; 1243;1152;1381;1363;1323;1338]
1400 NY	dualized	67883.9	67883.9	[1144;722;1390;721;1243;1082;1117;1070;1093;1068;866;697;326;108;582;1220;1153;579;1235;1223; 1206;1270;1271;1296]
1500 BAY	dualized	79379.7	75750.8	[1177;1161;1123;1093;994;582;671;1091;1235;850;1286;330;1045;859;1158;1332;1288;1469;1427;1443; 1152;1381;1363;1323;1338]

1500	COL	dualized	42505.7	36209.5	[88;538;1238;947;1464;151;535;1278;1079;1179;1056;1263;1000;1001;1053]
1500	NY	dualized	66475.5	66475.5	[1144;1218;1267;517;1165;1082;1117;1070;1093;507;866;1445;856;108;910;360;1207;579;1235;1223;1469;1470;1296]
1600	BAY	dualized	81809.7	77994.7	[1329;1404;1330;1203;1152;1572;1451;1249;1487;1424;1457;859;732;731;577;559;936;1299;1577;1024;1258;1338;792;826;1545;1580]
1600	COL	dualized	42505.7	36541.9	[88;362;1535;1511;275;274;1200;359;1199;110;292;1178;309;1001;1053]
1600	NY	dualized	66475.5	66475.5	[1133;1390;1054;721;1551;1248;1108;1070;442;1068;866;1463;585;447;910;1223;324;1437;1240;1547;1554;1555;1307]
1700	BAY	dualized	70557.8	66436.8	[1456;1407;1640;1459;1579;1097;1269;1159;902;1639;457;850;577;414;989;1213;1158;1424;1265;1402;1366;1572;1627]
1700	NY	dualized	66475.5	66475.5	[1133;722;1190;721;477;1248;1108;1070;694;1068;866;1463;585;1193;396;1223;1208;1176;1240;1547;1554;1555;1307]
1700	COL	dualized	42505.7	34354.9	[88;189;331;1560;1032;1114;104;1156;184;357;383;774;1430;1398;1053]
1800	COL	dualized	42505.7	35852.0	[88;931;331;188;1708;1352;360;1031;358;1253;1249;774;696;1398;1053]
1800	NY	dualized	69098.3	69098.2	[1133;621;1190;1171;1281;1082;1772;1740;1720;1706;866;1463;1451;1484;1450;581;1788;1176;1240;1226;1554;1555;1756;1785]
1800	BAY	dualized	72086.6	66058.8	[1627;1379;1366;1249;1487;1301;1409;859;889;1776;888;438;553;1688;1740;770;502;1338;1215;1259;1177;1456;1788]
1900	COL	dualized	42505.7	36057.7	[88;362;1717;1285;1032;967;1005;1031;358;1822;1434;528;1000;1001;1053]
1900	NY	dualized	69098.2	69098.2	[1133;1390;1054;1171;618;617;1772;1070;1044;507;866;697;326;838;325;916;1683;1176;1240;1547;1850;1288;1756;1785]
1900	BAY	dualized	76729.6	74871.4	[1580;1641;1793;1303;1093;1372;1269;1159;562;457;559;888;1016;1401;1340;1457;1898;1363;1809;1526;1379;1727;1679;1834]
2000	BAY	dualized	67604.0	67334.7	[1712;1862;1915;1861;1869;1822;1533;1933;732;516;1707;438;1974;1160;902;671;490;1280;1370;1987;1979;1637;1864;1918]
2000	COL	dualized	49019.1	41670.4	[88;1499;1650;518;952;1619;1511;1686;1685;1557;209;1679;1301;1299;1573;1753;1883;1888;1917;1887]
2000	NY	dualized	66853.3	62783.6	[1792;1604;1746;576;1982;1204;1481;729;1434;686;769;1490;1694;1783;1661;1858;1394;776;1787;1319;739;1741;1950;1973]
2100	BAY	dualized	72250.2	68091.9	[1788;1456;1177;1259;939;862;760;860;503;1692;861;936;595;424;516;732;520;1962;1965;1913;2018;1948;1835;1627;2038]
2100	COL	dualized	43998.7	35697.2	[88;473;1987;835;172;274;775;1477;1786;1163;658;1984;479;2012;1985;2013]
2100	NY	dualized	123257.0	123257.0	[1785;1756;1288;1850;1226;1240;1437;1905;1931;325;392;1828;1463;866;855;1044;1070;1108;617;

					618;517;646;621;1612;1599;1582;1569;2091;1553;1327;764;1434;1974]
2200	BAY	dualized	80585.3	76415.1	[1580;2029;1162;939;1900;760;1411;1577;902;1639;1212;2122;1827;731;1621;1340;1845;1898;2171; 1915;1890;1572;2127;2073;1834;2179]
2200	COL	dualized	48962.9	42350.8	[88;1297;1353;188;1964;1635;2018;628;1487;591;338;532;370;2006;1997;2033;2153]
2200	NY	dualized	72476.8	57833.7	[1785;1617;1823;1259;1538;692;1230;1738;1441;1834;966;1078;1366;731;1946;1799;1585;317;1323; 1432;1326;1677;1954;1974]
2300	COL	dualized	50172.5	41491.6	[88;1395;331;1045;687;248;973;1797;1558;957;1863;2231;2294;1423;1410;2033;2299;2272]
2300	BAY	dualized	77192.1	74781.4	[1788;1456;1177;1259;1578;771;825;938;995;2089;2027;2147;595;424;1937;889;2242;993;2137;2087; 2233;2178;2040;2070;2276;2300]
2300	NY	dualized	123257.0	83408.0	[1785;1756;1288;1554;1806;1240;1807;640;360;1450;392;326;697;866;507;1044;1070;1108;2209;1237; 875;1190;1631;2262;1599;728;1798;1092;2212;2224;2131;2141;1974]
2400	BAY	dualized	77192.1	73583.9	[1788;1956;1177;2090;792;970;760;582;671;2089;2027;1737;2217;1272;1937;1401;1422;993;1898; 1913;2233;1948;2129;2390;2276;2300]
2400	COL	heuristic	49019.1	42130.9	[88;473;1353;1915;205;151;104;87;247;61;1594;1984;2186;2202;2257;2031;2280;2273;2278;2272]
2400	NY	dualized	71974.3	71973.6	[1785;1307;1823;576;1883;1309;1230;1592;2350;2368;2380;2389;2318;1394;2061;2369;1952;2035; 2319;2336;2353;1954;1974]
2500	BAY	dualized	73897.1	68949.1	[2222;1712;1862;1915;2193;1869;2299;1367;1586;2328;516;424;331;1973;791;2365;1721;1823;771; 1334;1161;2486;2225;1918;2405]
2500	NY	dualized	71974.3	71816.8	[1792;1296;1066;1648;2405;1471;1959;2020;2373;2029;1570;2463;2112;1420;2278;2374;1969;2335; 2316;2336;2358;1950;1973]
2500	COL	heuristic	48427.8	41849.6	[88;1060;249;2121;260;2472;104;751;210;61;1463;1178;2297;1522;1700;1634;2161;1888;2050;2088]

Table 2: Meilleure solution trouvée pour chaque instance

C Incohérence CPLEX

Lors du débug dans le callBack du branch and cut, nous avons détecté une erreur de CPLEX. Afin de résoudre les sous-problèmes, on extrait les valeurs dont on a besoin dans un vecteur. Afin de ne pas perdre de temps à faire des $vect.push\_back(d_a)$, nous voulions connaître le nombre d'arcs utilisés par la solution actuelle. L'appel au Callback se fait en LazyCallBack, c'est à dire quand le solveur trouve une solution entière, ainsi, les valeurs de x_a sont binaires (à la précision numérique près). Cependant, nous nous sommes aperçus que $IloInt\ n_1 = getValue(IloSum(x))$ ainsi que $unsigned\ int\ n_2 = getValue(IloSum(x))$ valent 3 dans une solution où 4 noeuds sont sélectionnés. Cependant, $double\ n_3 = getValue(IloSum(x))$ vaut 4, qui est la valeur désirée, mais cela n'a pas de sens de donner un double comme taille d'un vecteur. Pour être sûr de ne pas avoir d'erreur, nous avons ainsi favorisé l'écriture avec un $push\_back$. C'est probablement une erreur d'arrondi, cependant il faudrait faire un rapport et l'envoyer à CPLEX.