ECMA

Pierrre Cortambert, Victor Spitzer

Décembre 2021

Table des matières

Introduction

Ce rapport présente un travail sur l'étude de cas du module ECMA, portée sur l'optimisation combinatoire robuste. Après avoir posé une modélisation PLNE pour le problème statique ainsi que pour des résolution par dualisation ou plans coupants, plusieur méthodes numériques sont proposées puis mises en application et comparées. Une résolution exacte par programmation dynamique ainsi qu'une heuristique obtenue par approxiamtion de cette résolution sont également étudiées. Des résultats sur quelques dizaines d'instances sont présentés pour évaluer la pertinence de chaque approche.

1 Modélisation papier

1.1 Question 1 : Modélisation du problème statique sous la forme d'un PLNE compact

On se place dans un graphe orienté $G = (V, A)$, et on cherche le chemin de durée la plus courte d'une source s vers une destination t dans ce graphe. On impose que la somme des poids de chaque sommet visité par ce chemin soit bornée par une constante fixée.

On propose une modélisation PLNE compacte du problème statique, c'est à dire en supposant fixées les durées de trajet d_{ij} et les poids de chaque sommet p_i. Pour cela, on s'inspire de la modélisation PLNE compacte d'un problème de flot, en posant x_{ij} le flot sur l'arc $(i, j) \in A$. Ainsi on a :

$$
\begin{aligned}
&\min_{x} \sum_{(i,j)\in A} x_{ij}.d_{ij} \\
&\quad \sum_{k;(k,i)\in A} x_{ki} - \sum_{q;(i,q)\in A} x_{iq} = 0, \quad \forall i \in V\backslash\{s,t\} \\
&\quad \sum_{q;(s,q)\in A} x_{sq} - \sum_{q;(q,s)\in A} x_{qs} = 1 \text{ et } \sum_{k;(k,t)\in A} x_{kt} - \sum_{q;(t,q)\in A} x_{tq} = 1 \\
&\quad \sum_{(i,j)\in A} x_{ij}.p_i + p_t \leq S \\
&\quad x_{ij} \in \{0,1\}, \quad \forall (i,j) \in A
\end{aligned}
$$

1.2 Question 2 : Modélisation du problème robuste sous la forme d'un PLNE

On cherche à présent à formuler une modélisation du problème robuste. Désormais les valeurs que peuvent prendre les durées et les poids sont représentées par les ensembles $\mathcal{U}_1, \mathcal{U}_2$, et on cherche une solution optimale dans le pire des cas. On reprends la formulation précédente pour intégrer les variations δ^1_{ij} et δ^2_i sur les durées et poids sous la forme de variables sous contraintes du problème tel que :

$$
\begin{aligned}
&\min_{x} \max_{\delta^1\in\mathcal{U}_1} \sum_{(i,j)\in A} x_{ij}.d_{ij} + x_{ij}.d_ij.\delta^1_{ij} \\
&\quad \sum_{k;(k,i)\in A} x_{ki} - \sum_{q;(i,q)\in A} x_{iq} = 0, \quad \forall i \in V\backslash\{s,t\} \\
&\quad \sum_{q;(s,q)\in A} x_{sq} - \sum_{q;(q,s)\in A} x_{qs} = 1 \text{ et } \sum_{k;(k,t)\in A} x_{kt} - \sum_{q;(t,q)\in A} x_{tq} = 1 \\
&\quad \sum_{(i,j)\in A} \left(x_{ij}.p_i + x_{ij}.\hat{p}_i.\delta^2_i\right) + p_t + \delta^2_t.\hat{p}_t \leq S, \quad \forall \delta^2 \in \mathcal{U}_2 \\
&\quad x_{ij} \in \{0,1\}, \quad \forall (i,j) \in A
\end{aligned}
$$

On en déduit la formulation PLNE du problème robuste en intégrant les contraintes sur les ensembles $\mathcal{U}_1$ et $\mathcal{U}_2$:

$$\begin{aligned}
&\min_{x} \max_{\delta^1} \sum_{(i,j)\in A} x_{ij}.d_{ij} + x_{ij}.d_{ij}.\delta^1_{ij} \\
&\quad \sum_{(k,i)\in A} x_{ki} - \sum_{(i,q)\in A} x_{iq} = 0, \quad \forall i \in V \\
&\quad \sum_{q;(s,q)\in A} x_{sq} - \sum_{q;(q,s)\in A} x_{qs} = 1 \text{ et } \sum_{k;(k,t)\in A} x_{kt} - \sum_{q;(t,q)\in A} x_{tq} = 1 \\
&\quad \max_{\delta^2} \sum_{(i,j)\in A} \left(x_{ij}.p_i + x_{ij}.\hat{p}_i.\delta^2_i\right) + p_t + \delta^2_t.\hat{p}_t \leq S \\
&\quad \sum \delta^1_{ij} \leq d_1 \text{ et } 0 \leq \delta^1_{ij} \leq D_{ij}, \quad \forall (i,j) \in A \\
&\quad \sum \delta^2_i \leq d_2 \text{ et } 0 \leq \delta^2_i \leq 2, \quad \forall i \in V \\
&\quad x_{ij} \in \{0,1\}, \quad \forall (i,j) \in A
\end{aligned}$$

1.3 Question 3 : Resolution par plans coupants

Question 3.a

On réécrit le problème pour que la robustesse n'apparaisse que dans les contraintes :

$$\begin{aligned}
&\min_{x,z} \sum_{(i,j)\in A} x_{ij}.d_{ij} + z \\
&\quad \sum_{(i,j)\in A} x_{ij}.d_{ij}.\delta^1_{ij} \leq z, \quad \forall \delta^1 \in \mathcal{U}_1 \\
&\quad \sum_{k;(k,i)\in A} x_{ki} - \sum_{q;(i,q)\in A} x_{iq} = 0, \quad \forall i \in V \backslash \{s,t\} \\
&\quad \sum_{q;(s,q)\in A} x_{sq} - \sum_{q;(q,s)\in A} x_{qs} = 1 \text{ et } \sum_{k;(k,t)\in A} x_{kt} - \sum_{q;(t,q)\in A} x_{tq} = 1 \\
&\quad \sum_{(i,j)\in A} \left(x_{ij}.p_i + x_{ij}.\hat{p}_i.\delta^2_i\right) + p_t + \delta^2_t.\hat{p}_t \leq S, \quad \forall \delta^2 \in \mathcal{U}_2 \\
&\quad x_{ij} \in \{0,1\}, \quad \forall (i,j) \in A
\end{aligned}$$

Question 3.b

Le principe de l'algorithme de coupe est d'ignorer initialement les contraintes de borne supérieure sur les coordonnées des variables $\delta^1_{i,j}$ et δ^2_i car les valeurs sur ces coordonnées n'ont d'importance que si on étudie un chemin réalisable passant respectivement par l'arc ou le sommet correspondant. On ajoute donc de nouvelles bornes sur ces variables au fur et à mesure que de nouveaux chemins sont explorés, sans considérer en permanence tous les chemins possibles, réalisables ou non.

Pour cette raison, on définit les ensembles $\mathcal{U}_1^*$ et $\mathcal{U}_2^*$ des coupes initiales pour obtenir une première solution réalisable pour le problème statique, et ajouter de nouvelles coupes en fonction de cette première solution. On fixe donc initialement tous les deltas à une valeur nulle :

$$\begin{cases} \mathcal{U}_0^* = \{0_{|A|}\} \\ \mathcal{U}_1^* = \{0_{|V|}\} \end{cases}$$

Question 3.c

Soit x^* la solution optimale de l'itération courante du problème maître, le sous-problème sur l'objectif est défini ainsi :

$$\begin{aligned} \max_{\delta^1} & \sum_{(i,j)\in A} x_{ij}^*.d_{ij}.(1+\delta_{ij}^1) \\ & \sum_{i,j} \delta_{ij}^1 \leq d_1 \\ & \delta_{ij}^1 \leq D_{ij}, \quad \forall (i,j) \in A \\ & \delta^1 \in \mathbb{R}_+^{|A|} \end{aligned}$$

De la même manière, le sous-problème sur les contraintes est défini par :

$$\begin{aligned} \max_{\delta^2} & \sum_{(i,j)\in A} x_{ij}^*.\hat{p}_i.\delta_i^2 + \delta_t^2.\hat{p}_t \\ & \sum_{i} \delta_i^2 \leq d_2 \\ & \delta_i^2 \leq 2, \quad \forall i \in V \\ & \delta^2 \in \mathbb{R}_+^{|V|} \end{aligned}$$

Question 3.d

Pour qu'une solution du problème maître soit optimale, il est nécessaire que les solutions des sous-problèmes objectif et contrainte appartiennent respectivement aux ensembles $\mathcal{U}_1^*$ et $\mathcal{U}_2^*$ à l'itération courante, et qu'on aura construites de manière itérative. Cela signifiera que cette solution est réalisable et optimale lorsqu'on considère le pire cas sur les valeurs prises par les paramètres δ^1 et δ^2.

Autrement dit, pour x^* la solution optimale de l'itération courante du problème maître et u_0^*, u_1^* les solutions des deux sous-problèmes correspondant, la solution x^* est optimale si :

$$\begin{cases} u_0^* \in \mathcal{U}_0^* \\ u_1^* \in \mathcal{U}_1^* \end{cases}$$

1.4 Question 4 : Résolution par dualisation

Question 4.a

On réécrit la formualtion donnée à la question 2 correspondant au problème robuste, mais on isole ici le terme faisant intervenir les variables δ_{ij}^1.

$$
\begin{aligned}
&\min_{x} \sum_{(i,j)\in A} x_{ij}.d_{ij} + \max_{\delta^1\in\mathcal{U}_1} \sum_{(i,j)\in A} x_{ij}.d_{ij}.\delta^1_{ij} \\
&\sum_{k;(k,i)\in A} x_{ki} - \sum_{q;(i,q)\in A} x_{iq} = 0, \quad \forall i \in V\backslash\{s,t\} \\
&\sum_{q;(s,q)\in A} x_{sq} - \sum_{q;(q,s)\in A} x_{qs} = 1 \text{ et } \sum_{k;(k,t)\in A} x_{kt} - \sum_{q;(t,q)\in A} x_{tq} = 1 \\
&\max_{\delta^2} \sum_{(i,j)\in A} \left(x_{ij}.p_i + x_{ij}.\hat{p}_i.\delta^2_i\right) + p_t + \delta^2_t.\hat{p}_t \leq S \\
&\sum \delta^1_{ij} \leq d_1 \text{ et } 0 \leq \delta^1_{ij} \leq D_{ij}, \quad \forall (i,j) \in A \\
&\sum \delta^2_i \leq d_2 \text{ et } 0 \leq \delta^2_i \leq 2, \quad \forall i \in V \\
&x_{ij} \in \{0,1\}, \quad \forall (i,j) \in A
\end{aligned}
$$

Question 4.b

Le problème interne lié aux variables δ^1_{ij} est :

$$
\begin{aligned}
&\max_{\delta^1\in\mathcal{U}_1} \sum_{(i,j)\in A} x_{ij}.d_{ij}.\delta^1_{ij} \\
&\sum_{(i,j)\in A} \delta^1_{ij} \leq d_1 \\
&0 \leq \delta^1_{ij} \leq D_{ij}, \quad \forall (i,j) \in A
\end{aligned}
$$

Question 4.c

Le dual s'écrit :

$$
\begin{aligned}
\min_{\alpha} \quad & \alpha_0.d_1 + \sum_{(i,j)\in A} \alpha_{ij}.D_{ij} \\
& \alpha_0 + \alpha_{i,j} \geq x_{ij}.d_{ij} \qquad \forall (i,j) \in A \\
& 0 \leq \alpha
\end{aligned}
$$

Question 4.d

Dans la contrainte robuste, on isole les termes liés à δ^2_{ij} :

$$\begin{aligned}
&\max_{\delta^2} \sum_{(i,j)\in A} \left(x_{ij}.p_i + x_{ij}.\hat{p}_i.\delta_i^2\right) + p_t + \delta_t^2.\hat{p}_t \leq S \\
&\sum_{i\in V} \delta_i^2 \leq d_2 \\
&0 \leq \delta_i^2 \leq 2, \quad \forall i \in V
\end{aligned}$$

Question 4.e

Le probème interne lié aux variables δ_i^2 ne prend pas en compte la contrainte inférieure ou égale à S. Par ailleurs le sous-problème a pour seule variable (δ_i^2) et on suppose x fixé. Ainsi le problème est le suivant :

$$\begin{aligned}
&\max_{\delta^2} \sum_{i\in V} \Big(\sum_{j;\,(i,j)\in A} \Big).\hat{p}_i.\delta_i^2 + \delta_t^2.\hat{p}_t + \sum_{(i,j)\in A} (x_{ij}.p_i) \\
&\sum_{i\in V} \delta_i^2 \leq d_2 \\
&0 \leq \delta_i^2 \leq 2, \quad \forall i \in V
\end{aligned}$$

Question 4.f

En dualisant la formulation précédente, on obtient :

$$\begin{aligned}
\min_{\alpha} \quad & \beta_0.d_2 + 2.\sum_{i\in V} \beta_i + \sum_{(i,j)\in A} (x_{ij}.p_i) \\
& \beta_0 + \beta_i \geq \hat{p}_i. \sum_{j;\,(i,j)\in A} x_{ij} \qquad \forall i \in V\backslash\{t\} \\
& \beta_0 + \beta_t \geq \hat{p}_t \\
& 0 \leq \beta
\end{aligned}$$

Question 4.g

On reprend le problème avec les questions précédentes, et on obtient :

$$
\begin{aligned}
&\min_{x} \sum_{(i,j)\in A} x_{ij}.d_{ij} + \max_{\delta^1\in\mathcal{U}_1} \sum_{(i,j)\in A} x_{ij}.\delta^1_{ij} \\
&\sum_{k;(k,i)\in A} x_{ki} - \sum_{q;(i,q)\in A} x_{iq} = 0, \quad \forall i \in V\backslash\{s,t\} \\
&\sum_{q;(s,q)\in A} x_{sq} - \sum_{q;(q,s)\in A} x_{qs} = 1 \text{ et } \sum_{k;(k,t)\in A} x_{kt} - \sum_{q;(t,q)\in A} x_{tq} = 1 \\
&\max_{\delta^2} \sum_{(i,j)\in A} \left(x_{ij}.p_i + x_{ij}.\hat{p}_i.\delta^2_i\right) + p_t + \delta^2_t.\hat{p}_t \leq S \\
&\sum \delta^1_{ij} \leq d_1 \text{ et } 0 \leq \delta^1_{ij} \leq D_{ij}, \quad \forall (i,j) \in A \\
&\sum \delta^2_i \leq d_2 \text{ et } 0 \leq \delta^2_i \leq 2, \quad \forall i \in V \\
&x_{ij} \in \{0,1\}, \quad \forall (i,j) \in A
\end{aligned}
$$

Tel que, par dualité, la valeur objectif du sous-problème relatif à δ_1 est donnée par :

$$
\begin{aligned}
\min_{\alpha} \quad & \alpha_0.d_1 + \sum_{(i,j)\in A} \alpha_{ij}.D_{ij} \\
& \alpha_0 + \alpha_{i,j} \geq x_{ij}.d_{ij} \qquad \forall (i,j) \in A \\
& 0 \leq \alpha
\end{aligned}
$$

On peut alors intégrer cette valeur-ci de l'objectif du sous-problème à l'objectif du problème maître, et reformuler celui-ci ainsi :

$$
\begin{aligned}
&\min_{x,\,\alpha} \sum_{(i,j)\in A} x_{ij}.d_{ij} + \alpha_0.d_1 + \sum_{(i,j)\in A} \alpha_{ij}.D_{ij} \\
&\sum_{(k,i)\in A} x_{ki} - \sum_{(i,q)\in A} x_{iq} = 0, \quad \forall i \in V\backslash\{s,t\} \\
&\sum_{q;(s,q)\in A} x_{sq} - \sum_{q;(q,s)\in A} x_{qs} = 1 \text{ et } \sum_{k;(k,t)\in A} x_{kt} - \sum_{q;(t,q)\in A} x_{tq} = 1 \\
&\alpha_0 + \alpha_{i,j} \geq x_{ij}.d_{ij} \qquad \forall (i,j) \in A \\
&0 \leq \alpha \\
&\max_{\delta^2} \sum_{(i,j)\in A} \left(x_{ij}.p_i + x_{ij}.\hat{p}_i.\delta^2_i\right) + p_t + \delta^2_t.\hat{p}_t \leq S \\
&\sum \delta^2_i \leq d_2 \text{ et } 0 \leq \delta^2_i \leq 2, \quad \forall i \in V \\
&x_{ij} \in \{0,1\}, \quad \forall (i,j) \in A
\end{aligned}
$$

De même, par dualité, la valeur objectif du sous-problème relatif à δ_2 est donnée par :

$$
\begin{aligned}
\min_{\alpha} \quad & \beta_0.d_2 + 2.\sum_{i\in V}\beta_i + \sum_{(i,j)\in A}(x_{ij}.p_i) \\
& \beta_0 + \beta_i \geq \hat{p}_i.\sum_{j;\,(i,j)\in A} x_{ij} \qquad \forall i \in V\backslash\{t\} \\
& \beta_0 + \beta_t \geq \hat{p}_t \\
& 0 \geq \beta
\end{aligned}
$$

Et la contrainte sur le sous-problème est que la valeur optimale soit inférieure à S. Un argument suffisant est qu'il existe une solution $(\tilde{\beta}_i)$ réalisable du dual pour ce sous-problème, et qui respecte l'inégalité : il vient immédiatement que la solution réalisable d'objectif minimale du sous-problème dual soit de valeur également inférieure à S :

$$
\min_{\beta} \beta_0.d_2 + 2.\sum_{i\in V}\beta_i + \sum_{(i,j)\in A}(x_{ij}.p_i) \leq \tilde{\beta}_0.d_2 + 2.\sum_{i\in V}\tilde{\beta}_i + \sum_{(i,j)\in A}(x_{ij}.p_i) \leq S - p_t
$$

On propose donc une dernière refomulation du problème maître en prenant en considération cet argument de dualité, et on obtient la formulation MILP standard suivante :

$$
\begin{aligned}
\min_{x,\,\alpha,\,\beta} \quad & \sum_{(i,j)\in A} x_{ij}.d_{ij} + \alpha_0.d_1 + \sum_{(i,j)\in A}\alpha_{ij}.D_{ij} \\
& \sum_{(k,i)\in A} x_{ki} - \sum_{(i,q)\in A} x_{iq} = 0, \quad \forall i \in V\backslash\{s,t\} \\
& \sum_{q;(s,q)\in A} x_{sq} - \sum_{q;(q,s)\in A} x_{qs} = 1 \text{ et } \sum_{k;(k,t)\in A} x_{kt} - \sum_{q;(t,q)\in A} x_{tq} = 1 \\
& \alpha_0 + \alpha_{i,j} \geq x_{ij}.d_{ij} \qquad \forall (i,j) \in A \\
& 0 \leq \alpha \\
& \beta_0.d_2 + 2.\sum_{i\in V}\beta_i + \sum_{(i,j)\in A}(x_{ij}.p_i) + p_t \leq S \\
& \beta_0 + \beta_i \geq \hat{p}_i.\sum_{j;\,(i,j)\in A} x_{ij} \qquad \forall i \in V\backslash\{t\} \\
& \beta_0 + \beta_t \geq \hat{p}_t \\
& 0 \leq \beta \\
& x_{ij} \in \{0,1\}, \quad \forall (i,j) \in A
\end{aligned}
$$

2 Résolution numérique

2.1 Plans coupants, Branch&Cut et dual

On implémente l'algorithme des plans coupants, du Branch&Cut et du dual comme détaillés ci-dessus.

Nous avons implémenté l'algorithme du dual avec CPLEX. L'implémentation a été directe et les temps de résolutions très bons.

Dans le cas des plans coupants, on distingue deux algorithmes : celui avec une résolution PLNE des sous-problèmes, et celui avec une résolution polynomiale :

Résoudre le premier sous-problème revient à résoudre un problème de sac-à-dos à variables réelles car pour une solution du problème maître x^*, on peut le reformuler ainsi :

$$\begin{aligned} \max_y & \sum_{(i,j)\in x^*} d_{ij}.y_{ij} \\ & \sum_{(i,j)\in x^*} y_{ij} \leq d_1 \\ & y_{ij} \in [0, D_{ij}] \end{aligned}$$

D'où on déduit une solution optimale définie pour (i,j) arc de x^* par $\delta^1_{ij} = y_{ij}$ et $\delta^1_{ij} = 0$ sinon. Une solution optimale s'obtient en triant les variables concernées (sur le chemin x^*) dans l'ordre décroissant des d_{ij} et en fixant dans cet ordre les variables à leur borne supérieure D_{ij} tant que la contrainte de capacité est respectée, puis en imposant à la variable suivante la valeur permettant d'atteindre l'égalité sur la contrainte de capacité. On fixe à 0 les variables restantes.

On résout de la même manière le second sous-problème, en le reformulant comme un problème de sac-à-dos en variable réelle :

$$\begin{aligned} \max_y & \sum_{i\in x^*} y_i.\hat{p}_i \\ & \sum_{i\in x^*} y_i \leq d_2 \\ & y_i \in [0, 2] \end{aligned}$$

D'où on déduit une solution optimale définie pour i sommet de x^* par $\delta^2_i = y_i$ et $\delta^2_i = 0$ sinon. Une solution optimale s'obtient en triant les variables concernées (sur le chemin x^*) dans l'ordre décroissant des $\hat{p}_i$ et en fixant dans cet ordre les variables à leur borne supérieure de valeur 2 tant que la contrainte de capacité est respectée, puis en imposant à la variable suivante la valeur permettant d'atteindre l'égalité sur la contrainte de capacité. On fixe à 0 les variables restantes.

Pour l'algorithme du Branch&Cut, nous avons considéré le même problème maître et les mêmes sous problèmes que pour l'algorithme des plans coupants. La différence est que nous résolvons les sous-problèmes à chaque noeud de l'arbre de branchement du problème maître qui retournent une solution entière. Nous l'avons implémenté via la fonction $callback_{bc}$ et des fonctions intégrées à CPLEX. On a également appliqué les résolutions polynomiales des sous-problèmes au branch-and-cut pour améliorer la génération de coupes.

2.2 Heuristique

On propose un algorithme exacte par programmation dynamique pour trouver le plus court chemin sous contrainte de poids sur les sommets, si les poids sur chaque sommet sont supposés entiers. Sans perte de généralités, on considère également que le poids sur le sommet destination t est nul. Enfin, on distingue le poids d'un sommet donné en entrée du problème, et le poids d'un chemin qui corresponds à la somme des poids de ses sommets. On présente l'algorithme dans le cas statique, pour ensuite le développer dans le cas robuste et présenter une heuristique pour améliorer les temps de calcul.

Cas statique

Sur chaque sommet x_i, on cherche les valeurs d'un vecteur $v(x_i) \in \mathbb{R}^S$ tel que $v_k(x_i)$ soit le couple coût-chemin de x_i à t de poids k et de coût minimal :

$$v_{k+p_i}(i) = \begin{cases} \min_{j;(i,j)\in A} \left(v_k(j) + d_{ij}\right) & \text{si } k + p_i \leq S \\ +\infty & \text{sinon} \end{cases}$$

On peut alors procéder par programmation dynamique en fixant initialement $v(t) = 0$ et pour tout autre sommet x_i, $v(x_i) = +\infty$. On minimise l'objectif en retrouvant dans $v(s)$ la coordonnée de plus faible coût, et son chemin associé : cette solution respectera la contrainte sur les poids car le poids de la solution correspond à son indice dans le vecteur $v(s)$, donc est inférieur à S. L'algorithme est similaire à celui de Bellman, excepté qu'ici on retient S valeurs sur chaque sommet, et le chemin optimal associé à chaque valeur correspondante.

Cas robuste : variation des distances

Considérons pour l'instant uniquement la variation sur les distances d_{ij} pour des poids fixés. On note **spo** la fonction par laquelle on obtient polynomialement la résolution du sous-problème objectif de la décomposition en plan coupant. Une telle fonction indique le pire surcout possible sur les arcs pour un chemin donnée. Avec celle-ci, on met à jour la fonction valeur de Bellman et ses sur-coûts d'objectif :

$$v_{k+p_i}(i) = \begin{cases} \min_{j;(i,j)\in A} \left(v_k(j) + d_{ij} + \text{spo}\left(\{i\} \cup x_k^*(j)\right)\right) & \text{si } k + p_i \leq S \\ +\infty & \text{sinon} \end{cases}$$

On revient alors à la résolution d'un algorithme de programmation dynamique comme dans le cas statique.

Cas robuste complet

Considérons désormais les variations à la fois sur les distances et les poids. On note $x_k^*(j)$ la solution optimale de somme de poids k à partir du sommet j. On introduit une fonction $p_k(j)$ qui indique la pire somme des coûts sur un chemin x :

$$p(x) = \min(d_2, \sum_{i\in x} 2\hat{p}_i) + \sum_{i\in x} p_i$$

On met alors à jour une dernière fois la fonction valeur de Bellman :

$$v_{p(\{i\}\cup x_k^*(j))}(i) = \begin{cases} v_k(j) + d_{ij} + \text{spo}\left(\{i\} \cup x_k^*(j)\right) \text{ si } \{i\} \cup x_k^*(j) \leq S \\ +\infty \text{ sinon} \end{cases}$$

Soit **spc** la fonction par laquelle on obtient polynomialement la résolution du sous-problème contrainte de la décomposition en plan coupant. Dans les faits, on calcule $p(x)$ ainsi :

$$p(x) = \begin{cases} \sum_{i\in x} 2\hat{p}_i + \sum_{i\in x} p_i \text{ si } 2.|x| \leq d_2 \\ \text{spc}(x) \text{ sinon} \end{cases}$$

Ainsi on limite les appels à la fonction **spc**. On revient encore une fois à la résolution d'un algorithme de programmation dynamique comme dans le cas statique.

Complexité et approche heuristique

Contrairement à un algorithme de Bellman, on a sur chaque sommet du graphe autant de valeurs et chemins associés que chaque valeur possible du cumul de poids sur un chemin optimal, soit S entrée sur chaque sommet. Le problème est donc pseudo-polynomial numérique.

Pour limiter le nombre d'itération des valeurs de Bellman, on construit un arbre couvrant avec le puit t comme racine et avec les arcs inversés, par une recherche en largeur. Ainsi, on identifie pour chaque sommet le nombre d'arc minimum nécessaire pour former un chemin vers le puit, et on ajoute à chaque itération de la valeur de bellman les sommets pouvant joindre le puit avec un arc supplémentaire. De cette manière, il suffit de $|V|$ itérations pour observer une convergence des valeurs. On en déduit que la complexité de l'algorithme est en $\mathcal{O}(S.|V|^3)$.

Pour diminuer la complexité de l'algorithme tout en proposant une approche heuristique plutôt qu'exacte, on cherche à diminuer le nombre d'entrée par sommet du graphe. Pour un nombre d'entrée fixée S_0, on stocke à l'entrée k la meilleure valeur de Bellman pour un poids le plus faible possible dans $[\llcorner k.\frac{S}{S_0}\lrcorner, \llcorner(k+1).\frac{S}{S_0}\lrcorner]$.

En sélectionnant toujours celle de poids minimum parmi les meilleures valeur de Bellman dans un intervalle de poids donné, on garantit l'existence d'une solution au problème même avec un nombre d'entrée en chaque sommet réduit. Cependant on ignore certaines solutions à meilleures valeur de Bellman mais à poids plus important, et il est donc possible de ne pas atteindre la meilleure solution. On doit donc chercher un compromis lorsqu'on pose $S_0 < S$ tel qu'il y ait un gain de temps sans une trop grande perte de précision : cela se fait au cas par cas.

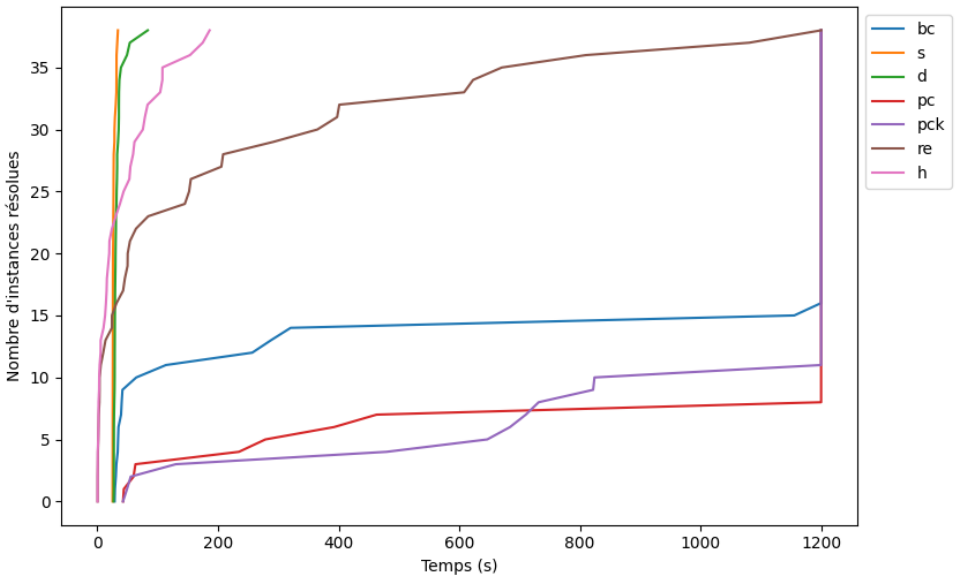
3 Présentation des résultats

3.1 Diagramme des performances

Nosu nous sommes limités à un temps maximal de 20 minutes afin de pouvoir obtenir les résultats des 39 instances étudiées avec 7 algorithmes différents pour un temps total maximum de 84 heures. Les instance de plus de 350 sommets n'ont pas été traitées car une majorité des méthodes ne sont plus efficaces pour celles-ci.

Les labels du graphe ci-dessous décrivent les méthodes suivantes :

— bc : Branch and Cut
— s : Resolution du probleme statique
— d : Resolution de la dualisation
— pc : Resolution du plan coupant
— pck : Resolution du plan coupant avec sous-problèmes polynomiaux
— re : Resolution exacte en programmation dynamique
— h : Resolution heuristique en programmation dynamique



Il apparaît de manière évidente que la méthode de dualisation est beaucoup plus efficace que celles par plans coupants. Le branch&cut semble apporter une amélioration par rapport aux méthodes plus simples de plans coupants, mais demeure lent pour de grandes instances. Enfin l'emploi d'heuristiques pour les sous-problèmes offre un gain marginal de temps, comme observé dans la comparaison entre les méthodes *pc* et *pck*.

La résolution exacte par programmation dynamique obtient de bien meilleures performances que celles par plans coupants mais reste inefficace pour de grandes instances, comme identifié dans le paragraphe dédié plus tôt. On emploie l'heuristique associée à cette résolution en fixant $S_0 = 100$. Non seulement les performances sont extrêmement satisfaisantes, mais les résultats sont très proche voire identiques à ceux d'une résolution exacte. Il est possible d'améliorer encore ces performances en diminuant S_0, au potentiel détriment de l'optimalité de la solution qui en découlera.

3.2 Tableau récapitulatif

			Plans coupants	PC	Branch&Cut	B&C	PC Knackpack	PC K
Taille	Zone	PR	Time	Gap	Time	Gap	Time	Gap
20	BAY	38.92	41.141	0.0	28.169	0.0	273.953	0.0
	COL	24.29	36.69	0.0	28.18	0.0	40.726	0.0
	NY	27.56	40.04	0.0	28.203	0.0	44.965	0.0
40	BAY	37.96	402.772	0.0	29.642	0.0	1201.233	2.12
	COL	33.03	207.91	0.0	33.664	0.0	1213.778	3.11
	NY	28.39	142.379	0.0	30.435	0.0	1208.891	5.21
60	BAY	34.49	464.521	0.0	34.235	0.0	1211.432	1.83
	COL	29.31	63.046	0.0	29.887	0.0	41.358	0.0
	NY	61.46	1212.784	4.86	29.887	0.0	1201.698	1.30
80	BAY	18.78	1216.128	3.44	50.332	0.0	1205.333	1.72
	COL	23.71	1205.057	3.99	37.944	0.0	117.554	0.0
	NY	61.46	1213.115	10.44	152.186	0.0	1223.228	4.09
100	BAY	18.78	1204.732	8.62	225.837	0.0	1206.373	3.44
	COL	25.23	1276.002	5.09	1066.371	0.0	1248.673	5.01
	NY	54.98	1251.46	9.22	218.204	0.0	1262.172	4.67
120	BAY	17.88	1227.524	7.95	1238.691	0.0	1210.904	7.95
	COL	26.00	1217.626	7.59	1238.888	0.0	1236.135	7.59
	NY	38.20	1224.472	12.71	1239.198	0.0	1221.529	7.02
140	BAY	22.21	1235.207	7.08	1239.015	0.0	1248.594	7.00
	COL	25.75	1212.315	0.53	314.815	0.0	1235.051	2.20
	NY	22.91	1228.089	6.47	1232.195	0.0	1202.122	4.67
160	BAY	18.01	1203.629	8.34	1237.25	0.0	1206.47	8.34
	COL	25.75	1225.139	3.15	1237.25	0.0	1229.889	3.15
	NY	36.94	1215.406	24.69	1235.722	0.0	1213.948	12.29
180	BAY	18.01	1205.713	10.04	1236.407	0.0	1204.047	10.04
	COL	32.23	1242.649	8.96	1215.731	0.0	1221.964	8.96
	NY	35.07	1274.429	27.62	1247.608	0.0	1224.14	18.03
200	BAY	18.01	1206.626	10.04	1239.913	0.0	1202.923	10.04
	COL	24.99	1303.792	5.06	1240.064	0.0	1217.325	5.06
	NY	19.50	1320.611	8.59	1235.793	0.0	1202.222	8.59
250	BAY	25.98	1203.025	3.82	1240.387	0.0	1217.159	2.00
	COL	24.13	1229.912	10.24	1432.37	0.0	1207.953	10.24
	NY	21.37	1202.575	3.90	2242.631	0.0	1202.659	3.90
300	BAY	24.58	1202.787	5.41	1263.677	0.0	1223.716	4.68
	COL	22.87	1226.945	18.85	3410.264	5.11	1221.022	14.57
	NY	20.94	1226.49	4.38	2024.365	0.0	1206.541	3.89
350	BAY	24.58	1201.2	5.41	1763.754	0.0	1214.859	5.41
	COL	22.00	1202.494	9.11	6174.686	0.69	1205.948	6.79
	NY	18.29	1208.041	11.596	1241.766	0.0	1244.276	9.61

			Dual	Dual	Heuristique	H	Resolution exacte	R exacte
Taille	Zone	PR	Time	Gap	Time	Gap	Time	Gap
20	BAY	38.92	27.333	0.0	0.21	0.0	0.27	0.0
	COL	24.29	26.671	0.0	0.21	0.0	0.32	0.0
	NY	27.56	27.339	0.0	0.52	0.0	0.86	0.0
40	BAY	37.96	27.624	0.0	0.34	0.0	0.35	0.0
	COL	33.03	27.186	0.0	1.69	0.0	3.57	0.0
	NY	28.39	27.3	0.0	1.85	0.0	1.79	0.0
60	BAY	34.49	27.146	0.0	0.38	0.0	0.67	0.0
	COL	29.31	27.261	0.0	5.43	17.61	13.38	0.0
	NY	61.46	28.668	0.0	1.3	0.0	1.41	0.0
80	BAY	18.78	29.624	0.0	2.77	0.0	3.59	0.0
	COL	23.71	27.981	0.0	2.95	0.0	3.26	0.0
	NY	61.46	28.413	0.0	1.78	0.0	2.22	0.0
100	BAY	18.78	28.703	0.0	3.71	0.0	5.58	0.0
	COL	25.23	28.822	0.0	12.7	0.0	23.97	0.0
	NY	54.98	28.599	0.0	5.31	0.0	9.67	0.0
120	BAY	17.88	30.637	0.0	15.73	0.0	50.21	0.0
	COL	26.00	29.33	0.0	14.15	0.0	42.58	0.0
	NY	38.20	28.997	0.0	10.14	0.0	24.05	0.0
140	BAY	22.21	30.058	0.0	17.7	0.0	53.94	0.0
	COL	25.75	28.905	0.0	15.21	0.0	31.68	0.0
	NY	22.91	29.204	0.0	19.59	0.0	49.95	0.0
160	BAY	18.01	34.188	0.0	43.4	0.0	152.03	0.0
	COL	25.75	29.907	0.0	24.03	0.0	45.58	0.0
	NY	36.94	30.732	0.0	19.97	0.0	63.9	0.0
180	BAY	18.01	35.782	0.0	52.98	0.0	208.42	0.0
	COL	32.23	30.323	0.0	37.33	0.0	144.99	0.0
	NY	35.07	31.582	0.0	30.38	6.14	84.18	0.0
200	BAY	18.01	35.616	0.0	59.14	0.0	154.89	0.0
	COL	24.99	32.708	0.0	75.36	6.32	291.64	0.0
	NY	19.50	31.831	0.0	54.62	0.0	205.53	0.0
250	BAY	25.98	35.644	0.0	107.83	0.0	622.93	0.0
	COL	24.13	32.374	0.0	78.74	2.32	364.75	0.0
	NY	21.37	32.716	0.0	61.43	0.0	397.49	0.0
300	BAY	24.58	53.701	0.0	174.6	0.0	809.98	0.0
	COL	22.87	35.222	0.0	107.72	5.11	608.02	0.0
	NY	20.94	48.993	0.0	83.19	0.0	401.03	0.0
350	BAY	24.58	83.434	0.0	186.05	0.0	1200.0	0.0
	COL	22.00	36.448	0.0	104.06	6.13	670.59	0.0
	NY	18.29	39.011	0.0	153.47	8.51	1080.82	0.0

Conclusion

Cette étude de cas nous a donné l'opportunité de nous familiariser à l'outil de résolution CPLEX, ainsi qu'à la programmation mathématiques pour les problèmes d'optimisation robuste. La variété des approches et des implémentations ont permis une étude rigoureuse de chaque approche, ainsi que la compréhension de leurs avantages et inconvénients.