ECMA : Rapport Final

Antonio Tavares, Julien Dallot

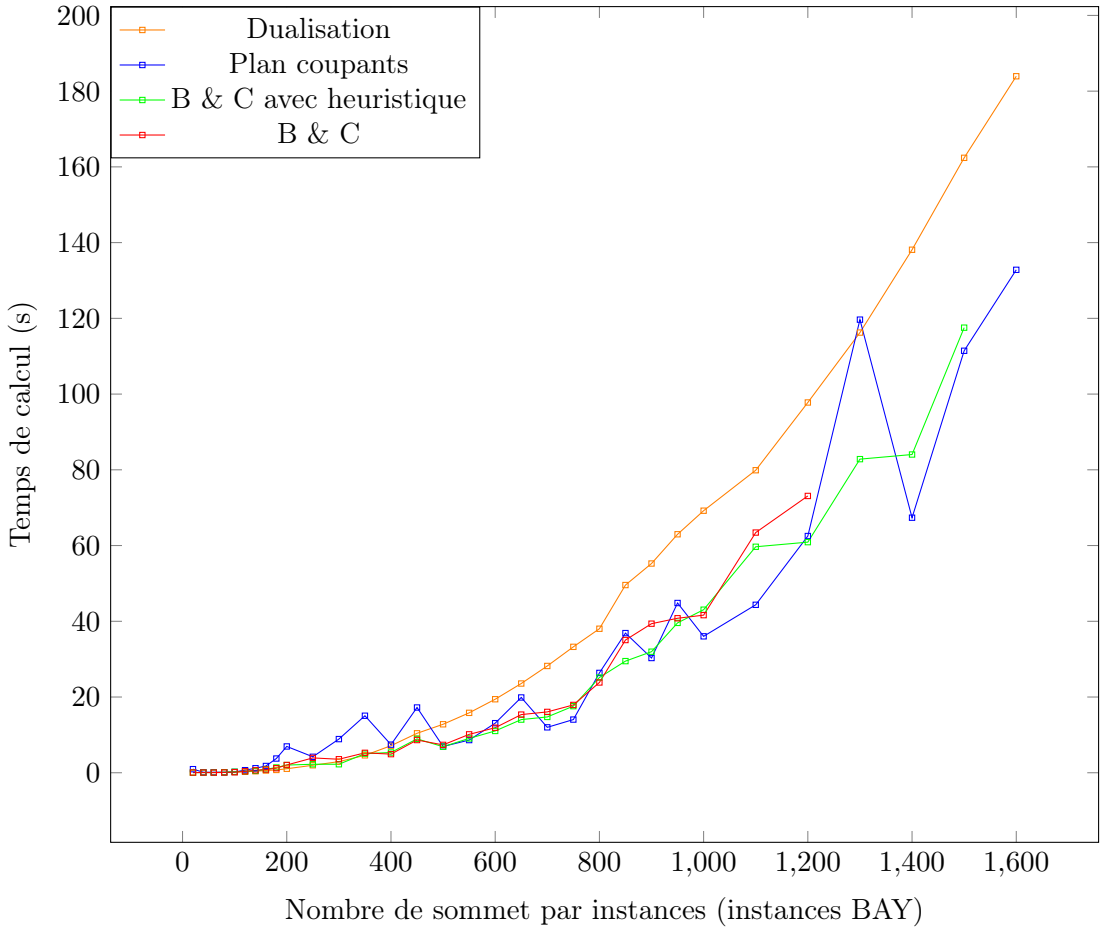
6 février 2022

Nous avons réalisé l'entièreté de ce projet avec le langage julia couplé au solveur CPLEX, version académique. Les calculs ont été effectués sur un processeur AMD Ryzen 9 5900HX avec 16 GB de mémoire vive.

1 Performances des différentes méthodes

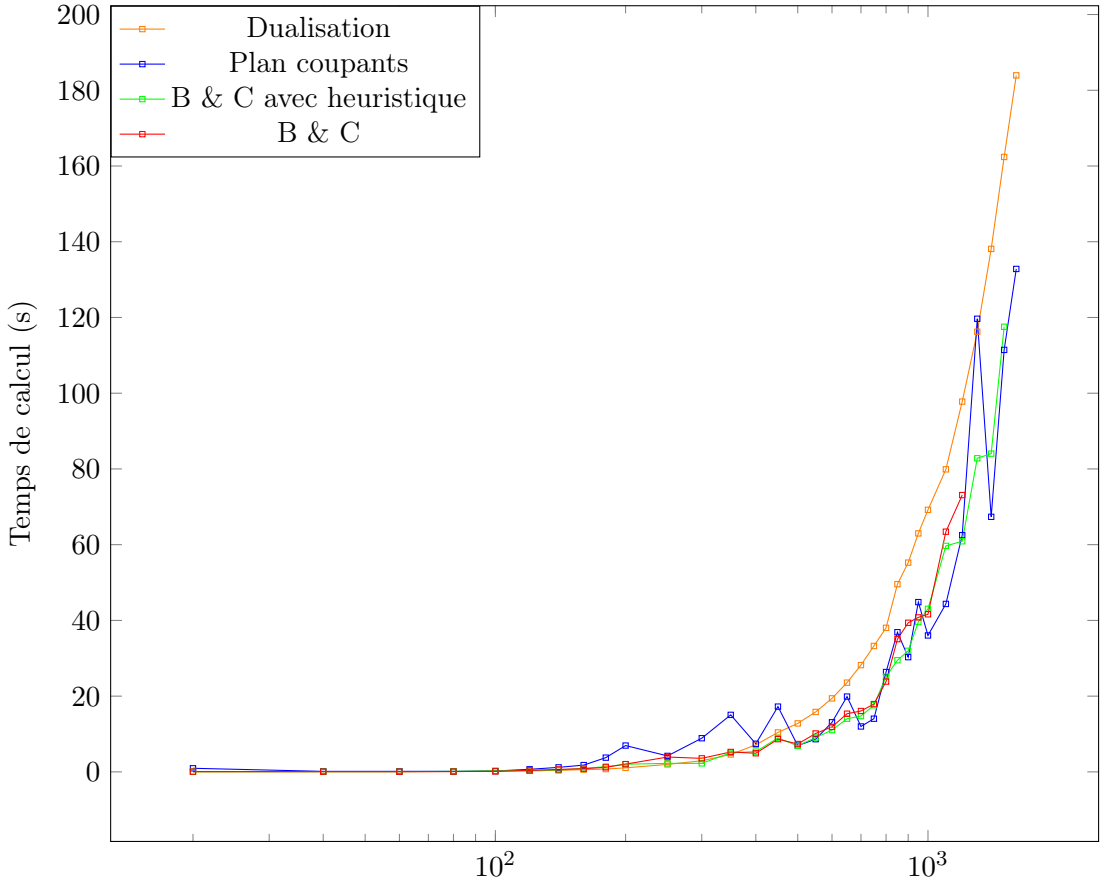
Nous avons implémenté les méthodes de dualisation et des plans coupants telles quelles, sans technique supplémentaire. Nous avons néanmoins codé une version améliorée du branch & cut en utilisant des heuristiques gloutonnes pour résoudre les sous-problèmes. Voici le graphique regroupement nos temps de calculs pour les quatre méthodes, par dualisation, plans coupants, branch & cut et branch & cut avec heuristique.

Temps de calculs en fonction de la taille de l'instance, pour les quatre méthodes



Le même graphique en échelle logarithmique pour mieux distinguer les meilleures méthodes sur les petites instances.

Temps de calculs en fonction de la taille de l'instance, pour les quatre méthodes



Nombre de sommet par instances (instances BAY, échelle logarithmique)

Nous avons obtenu nos temps de calculs en utilisant le paquet BenchmarkTools de julia, qui permet de calculer le temps moyen et la place mémoire utilisée entre autres. Parmi les quatre méthodes, le branch & cut avec heuristique se démarque en fournissant des temps plus régulier et globalement moins élevés que les autres, le branch & cut sans heuristique (les sous problèmes sont alors résolus par un PL avec CPLEX) présente des résultats de la même qualité, légèrement mois bons néanmoins. Les plans coupants fournissent des temps dans le même ordre de grandeur mais plus irréguliers, dont le temps de calcul peut parfois augmenter subitement. La dualisation est nettement moins bonne, ce qui est plutôt intuitif dans la mesure où cette méthode consiste à attaquer le problème frontalement sous la forme d'un PLNE unique : CPLEX prend naturellement plus de temps sans plus de spécificité sur le problème.

Nous avons rassemblé les performances pour les quatre méthodes dans le tableau ci-dessous. La colonne PR indique le pourcentage d'éloignement de la solution robuste par rapport à la solution du problème statique, c'est le "prix de robustesse".

Instance	PR	Plans coupants		Dualisation		Branch & Cut		Heuristique	
		Time (s)	Gap	Time (s)	Gap	Time (s)	Gap	Time (s)	Gap
20.BAY	38,9%	0,95	0%	0,022	0%	0,06	0%	0,06	0%
40.BAY	33,1%	0,14	0%	0,036	0%	0,06	0%	0,07	0%
60.BAY	6,9%	0,14	0%	0,045	0%	0,05	0%	0,07	0%
80.BAY	18,5%	0,15	0%	0,083	0%	0,08	0%	0,14	0%
100.BAY	18,5%	0,23	0%	0,18	0%	0,15	0%	0,26	0%
120.BAY	17,8%	0,68	0%	0,26	0%	0,39	0%	0,41	0%
140.BAY	18,1%	1,19	0%	0,40	0%	0,65	0%	0,51	0%
160.BAY	21,0%	1,79	0%	0,59	0%	0,85	0%	0,90	0%
180.BAY	21,0%	3,76	0%	0,74	0%	1,20	0%	1,39	0%
200.BAY	21,0%	6,95	0%	1,06	0%	2,05	0%	2,03	0%
250.BAY	19,4%	4,22	0%	1,98	0%	3,92	0%	2,25	0%
300.BAY	19,7%	8,90	0%	2,94	0%	3,57	0%	2,25	0%
350.BAY	19,7%	15,06	0%	4,60	0%	5,26	0%	5,04	0%
400.BAY	19,4%	7,43	0%	7,14	0%	4,92	0%	5,35	0%
450.BAY	19,4%	17,23	0%	10,42	0%	8,64	0%	8,95	0%
500.BAY	16,7%	6,94	0%	12,80	0%	7,36	0%	6,88	0%
550.BAY	16,6%	8,63	0%	15,82	0%	10,14	0%	9,07	0%
600.BAY	17,8%	13,10	0%	19,42	0%	11,89	0%	11,04	0%
650.BAY	17,8%	19,88	0%	23,57	0%	15,36	0%	14,04	0%
700.BAY	17,7%	12,00	0%	28,20	0%	16,07	0%	14,74	0%
750.BAY	17,7%	14,05	0%	33,25	0%	17,91	0%	17,59	0%
800.BAY	21,3%	26,37	0%	38,04	0%	23,84	0%	25,15	0%
850.BAY	20,5%	36,88	0%	49,57	0%	35,05	0%	29,48	0%
900.BAY	20,4%	30,30	0%	55,26	0%	39,37	0%	31,97	0%
950.BAY	20,4%	44,83	0%	62,98	0%	40.80	0%	39,58	0%
1000.BAY	20,1%	36.03	0%	69.20	0%	41.62	0%	43.06	0%
1100.BAY	19,4%	44.36	0%	79.89	0%	63.43	0%	59.68	0%
1200.BAY	20,0%	62.51	0%	97.78	0%	73.10	0%	60.90	0%
1300.BAY	20,0%	119.66	0%	116.24	0%	-	-	82.82	0%
1400.BAY	18,9%	67.36	0%	138.12	0%	-	-	84.04	0%
1500.BAY	18,9%	111.44	0%	162.39	0%	-	-	117.54	0%
1600.BAY	18,5%	132.82	0%	183.93	0%	-	-	123.02	0%
1700.BAY	19,7%	-	-	-	-	-	-	154.08	0%

TABLE 1 – Tableau de performances

2 Remarques sur l'implémentation

Nous avons réalisé des heuristiques pour résoudre les sous problèmes (SP1) et (SP2) dans le branch & cut en les réduisant à des problèmes de sac à dos continues (SaDC), sur lesquels une heuristique gloutonne rapide et simple donne une solution exacte [1]. L'heuristique s'applique sur des instances du type

$$(\text{SaDC}) \begin{cases} \max\limits_{x} & \sum\limits_{v \in V} a_v \, z_v & & \\ \text{s.c.} & & & \\ & \sum\limits_{v \in V} b_v z_v & \leq & S \\ & x_v \in [0,1] & & \forall v \in V \end{cases}$$

Avec $a, b \in \mathbb{R}^n_+$ respectivement les profits et les poids des objets. L'heuristique consiste à classer chaque objets $v \in V$ par ordre décroissant des ratios $\frac{a_v}{b_v}$, puis de parcourir cette liste en fixant $x_v = 1$ tant que le budget S n'est pas dépassé. Soit v_{lim} l'objet dont la sélection dépasse le budget S, soit S_{rest} le budget restant dans le sac à ce moment, on fixe alors $x_{v_{\text{lim}}} = S_{\text{rest}}/b_{v_{\text{lim}}}$. On fixe $x_v = 0$ pour tous les objets v restants.

Le problème (SP2) par exemple se présente sous une forme légèrement différente car les variables continues δ_v^2, $\forall v$ sommet, sont dans $[0, \hat{p}_v]$. Un simple changement de variable $\sigma_v = \frac{\delta_v^2}{\hat{p}_v}$. transforme le sous-problème en un problème équivalent (SP2-σ) suivant :

$$(\text{SP2-}\sigma) \begin{cases} \max\limits_{y} & \sum\limits_{v \in V} p_v \, \hat{p}_v \, \sigma_v & & \\ \text{s.c.} & & & \\ & \sum\limits_{v \in V} \hat{p}_v \sigma_v & \leq & d_2 \\ & \sigma_v \in [0,1] & & \forall v \in V \end{cases}$$

L'heuristique décrite ci-dessus s'applique directement à ce problème, et on remarque que les ratios à classer $\frac{p_v \, \hat{p}_v}{\hat{p}_v}$ se simplifient en le poids du sommet p_v. Il suffit donc de classer les objets par poids, puis de leur appliquer l'heuristique pour avoir une solution maximale de (SP2-σ).

Dans nos essaies, cette heuristique marche très bien pour le sous-problème (SP2) (la résolution en programmation linéaire et notre heuristique donnent bien les mêmes résultats pour toutes les instances testées), mais elle ne fonctionne pas systématiquement sur le sous-problème (SP1), sans explication de notre part. Les essais réalisés dans la section précédente sur le branch & cut avec heuristique n'utilisent donc l'heuristique que pour le problème (SP2), (SP1) est résolu avec CPLEX.

Références

[1] Michael T. Goodrich et Roberto Tamassia. "Algorithm design : Foundations, Analysis and Internet Examples". In : (1999). Sous la dir. de John Wiley & Sons, p. 259-260.