Rapport ECMA - MPRO 2022-2023

Camille Richer, Héloïse Gachet

Le langage utilisé est Julia, avec les modules `JuMP` et `CPLEX`.

Rappel du problème et des notations

On cherche à partitionner les sommets d'un graphe non orienté complet $G = (V, E)$ avec des longueurs d'arêtes l_{ij} et des poids de sommets w_i en K parties de poids maximum B, de façon à minimiser le poids des arêtes à l'intérieur des parties.

On définit les variables de décision suivantes :

$$\forall u, v \in \{1, \dots, n\} : x_{uv} = \begin{cases} 1 \text{ si } u \text{ et } v \text{ sont dans la même partie} \\ 0 \text{ sinon} \end{cases}$$

$$\forall v \in V, \forall k \in \{1, \dots, K\} : y_{vk} = \begin{cases} 1 \text{ si } v \text{ est dans la partie } k \\ 0 \text{ sinon} \end{cases}$$

Le problème robuste s'écrit (avec la robustesse dans les containtes) :

$$(\mathcal{P}) \begin{cases} \min\limits_{x,y,z} & z & & \\ s.c. & z \geq \sum\limits_{i=1}^{n} \sum\limits_{j=1}^{n} l_{ij}^1 x_{ij} & \forall l^1 \in \mathcal{U}^1 & (1) \\ & \sum\limits_{k=1}^{K} y_{ik} = 1 & \forall i \in \{1, \dots, n\} & (2) \\ & \sum\limits_{i=1}^{n} w_i^2 y_{ik} \leq B & \forall k \in \{1, \dots, K\}, \forall w^2 \in \mathcal{U}^2 & (3) \\ & y_{ik} + y_{jk} \leq x_{ij} + 1 & \forall i, j \in \{1, \dots, n\}, \forall k \in \{1, \dots, K\} & (4) \\ & x_{ij}, y_{ik} \in \{0, 1\} & \forall i, j \in \{1, \dots, n\}, \forall k \in \{1, \dots, K\} & (5) \end{cases}$$

Où les incertitudes sur les longueurs des arêtes et sur les poids des nœuds sont représentées par les ensembles :

$$\mathcal{U}^1 = \{\{l_{ij}^1 = l_{ij} + \delta_{ij}^{1*}(\hat{l}_i + \hat{l}_j)\}_{i,j \in E}, \delta_{ij}^1 \in [0, 3] \forall i, j \text{ et } \sum_{i,j \in E} \delta_{ij}^1 \leq L\}$$

$$\mathcal{U}^2 = \{\{w_i^2 = w_i(1 + \delta_j^{2*})\}_{i \in V}, \delta_i^2 \in [0, W_i] \forall i \in V \text{ et } \sum_{i \in V} \delta_i^2 \leq W\}$$

Méthodes exactes

Plans coupants et branch and cut

Pour les méthodes des plans coupants et du *branch and cut*, les ensembles $\mathcal{U}^1$ et $\mathcal{U}^2$ sont initialisés avec les valeurs du problème statique :

$$\mathcal{U}^{1*}_{\text{initial}} = \{\{l_{ij}^1 = l_{ij}\}_{i,j \in E}\}$$

$$\mathcal{U}^{2*}_{\text{initial}} = \{\{w_i^2 = w_i\}_{i \in V}\}$$

Pour trouver les nouvelles coupes à ajouter lorsqu'on a une solution (x^*, y^*, z^*), on résout les programmes linéaires des sous-problèmes $\mathcal{SP}_1$ et $\mathcal{SP}_2 k$ (cf modélisation papier). Ce sont les mêmes coupes, mais dans le cas des plans coupants on reprend la résolution de $\mathcal{P}$ depuis le début alors que dans le cas du *branch and cut*, en utilisant les

callbacks, on conserve le travail fait avant d'avoir ajouté une coupe pour la suite de la résolution. Le *branch and cut* devrait donc être plus rapide à l'exécution.

Les implémentations de ces méthodes sont classiques en utilisant CPLEX.

Dualisation

Le problème pour la résolution par dualisation s'écrit :

$$\begin{cases} \min_{x,y,\alpha,\beta} & \sum_{i,j\in E} l_{ij}x_{ij} + L\alpha + \sum_{i,j\in E} 3\beta_{ij} \\ s.c. & W\mu + \sum_{v\in V} W_v\lambda_v + \sum_{v\in V} w_v y_{v,k} \leq B \quad \forall k \in \{1...n\} \\ & \mu + \lambda_v \geq w_v y_{v,k} \quad \forall v \in V, \forall k \in \{1...n\} \\ & \mu \geq 0 \\ & \lambda_v \geq 0 \quad \forall v \in V \\ & \alpha + \beta_{ij} \geq (\hat{l}_i + \hat{l}_j)x_{ij} \quad \forall i,j \in E \\ & \alpha \geq 0 \\ & \beta_{ij} \geq 0 \quad \forall i,j \in E \\ & (2),(4),(5) \end{cases}$$

La résolution se fait de façon standard avec CPLEX.

Heuristiques

Heuristique type bin-packing

La première heuristique développée est de type bin-packing pour assurer la réalisabilité de la solution obtenue. On considère pour chaque sommet son pire poids indépendemment les uns des autres $w_i(1+W_i)$ et pour les longueurs des arêtes, leur pire longueur indépendemment les unes des autres $l_{ij}+3(\hat{l}_i+\hat{l}_j)$. On trie les sommets par pire poids décroissant et on les place dans la partition de façon gloutonne : pour chaque sommet i, on regarde, parmi les parties dans lesquelles on peut le placer, celle qui augmente le moins le coût courant de la solution et on place i dedans. Cette heuristique permet d'obtenir une solution réalisable pour toutes les instances, même si les poids et longueurs considérés ne respectent pas les contraintes sur $\mathcal{U}^1$ et $\mathcal{U}^2$ dans leur ensemble. En revanche, de ce fait, le coût de la solution est largement surestimé et on le recalcule au besoin à la fin avec un PL pour trouver le pire cas en utilisant la partition obtenue.

La complexité de cette méthode, de la façon dont elle est implémentée, est en $O(n+n^2+n\log(n)+n^2K)$ plus la complexité polynomiale éventuelle du PL à la fin pour le calcul du coût. Comme on peut majorer K par n (pour $K=n$ on a une solution triviale de coût nul), la complexité total de l'heuristique sans le PL est en $O(n^3)$.

Ajout d'une phase recherche locale

Pour améliorer l'heuristique précédente avec un faible coût en temps de calcul, on lui ajoute une phase de recherche locale limité à 30s : en partant de la solution obtenue par l'heuristique, on recalcule son "vrai" coût par le PL, puis pour chaque itération, on tire au hasard deux parties k_1 et k_2 ($k_2 \neq k_1$), un élément au hasard dans k_1 ou non et un élément au hasard dans k_2. On a donc tiré un ou deux éléments. On construit une nouvelle partition dans laquelle on met l'élément tiré de k_1 dans k_2 (s'il existe) et celui de k_2 dans k_1. On vérifie ensuite que la solution est réalisable en résolvant le sous-problème pour chacune des parties. Si elle est valide, on calcule son coût, et s'il est meilleure que la solution précédente, on la garde.

La complexité ajoutée par cette phase est polynomiale aussi : la construction de la nouvelle partition se fait en $O(n)$ et ensuite on résout des PL. Mais comme on donne une limite de temps de recherche, cela entraîne seulement une augmentation fixe et contrôlée du temps de calcul sur chaque instance.

Initialisation avec K-centers

L'objectif de cette heuristique est de choisir une initialisation de l'heuristique meilleure que celle par défaut : avec le comportement de l'heuristique par défaut, les K premiers sommets sont placés chacun dans une partie. Ici, on essaie de choisir K meilleurs sommets initiaux comme centre de chaque partie. Plutôt que d'initialiser le premier centre aléatoirement, c'est le sommet de plus grand pire poids qui est choisi (puisqu'il ira bien quelque part). Ensuite, on choisit les $K-1$ centres restants en prenant à chaque fois le sommet le plus éloigné des centres actuellement sélectionnés. Une fois que les K parties sont initialisées, on applique l'heuristique précédente sur les sommets restants. Toutefois, cette méthode ne semble pas meilleure que la première heuristique et ne garantie pas de trouver

une solution réalisable pour toutes les instances. Sa complexité est de l'ordre de $O(nK^2 + Kn^2)$ soit $O(n^3)$, comme la première.

K-means++ et réparation

Pour cette heuristique, on essaie de tirer parti du caractère euclidien des instances : le problème consiste à minimiser la somme des distances à l'intérieur de chaque partie, donc intuitivement, cela consiste à former des clusters dans le plan. Cependant comme on doit respecter les contraintes de poids sur les parties, il faudra réparer la solution trouvée. Empiriquement, on constate que les contraintes de poids sont telles que k-means++ ne trouve jamais une solution réalisable sur les instances. À partir de la solution obtenue par k-means++, on reconstruit donc une solution : on parcourt les sommets dans l'ordre des pires poids décroissants, et pour chaque sommet, si on peut le mettre dans la partie dans laquelle il est dans la solution k-means++, on l'y met, et sinon on le place au mieux à la façon de la première heuristique. Il faut impérativement placer le sommet dans une partie tout de suite : si on essaie de conserver au maximum la solution k-means++ et qu'on met les sommets qui ne rentrent pas dans leur partie en attente pour les recaser à la fin, on risque fortement de ne pas trouver de solution réalisable.

Pour l'implémentation, on utilise le module `Clustering` de Julia. Comme cet algorithme n'est pas déterministe, il peut arriver que pour certaines `random seed`, la réparation échoue à donner une solution réalisable. Cependant, l'heuristique étant rapide, on peut se permettre de relancer l'algorithme avec une autre graine si la première échoue. Empiriquement, tester jusqu'à 3 graines pour chaque instance permet de trouver une solution réalisable pour toutes les instances (peut-être que 2 suffisent même).

Conclusion sur les heuristiques

La grande difficulté est de s'assurer que l'heuristique renvoie une solution réalisable, et cela nous amène, dans tous les cas, à revenir à l'heuristique de base type bin-packing dans toutes les méthodes. Pour les résultats, on se contentera donc d'utiliser celle-ci, accompagnée de la phase de recherche locale dont le coût en temps peut être choisi à loisir et qui permet de conserver la réalisabilité. Le tableau 1 donne la valeur de la solution réalisable trouvée après l'étape de type bin-packing et la meilleure valeur trouvée à l'issu de 30 secondes de recherche locale.

Le graphique 1 reprend les valeurs du tableau et représente le gain relatif avant et après recherche locale, en pourcentage ((valeur avant RL - valeur après)/valeur avant)*100. Les instances sont ordonnées par nombre de nœuds croissants puis par K croissant. On constate que le gain est conséquent pour les instances de petite taille et de taille moyenne, jusqu'à 100 nœuds, et pour toutes les valeurs de K, mais pour les instances de 200 nœuds et plus, la recherche locale ne parvient plus à trouver d'amélioration suffisamment vite.

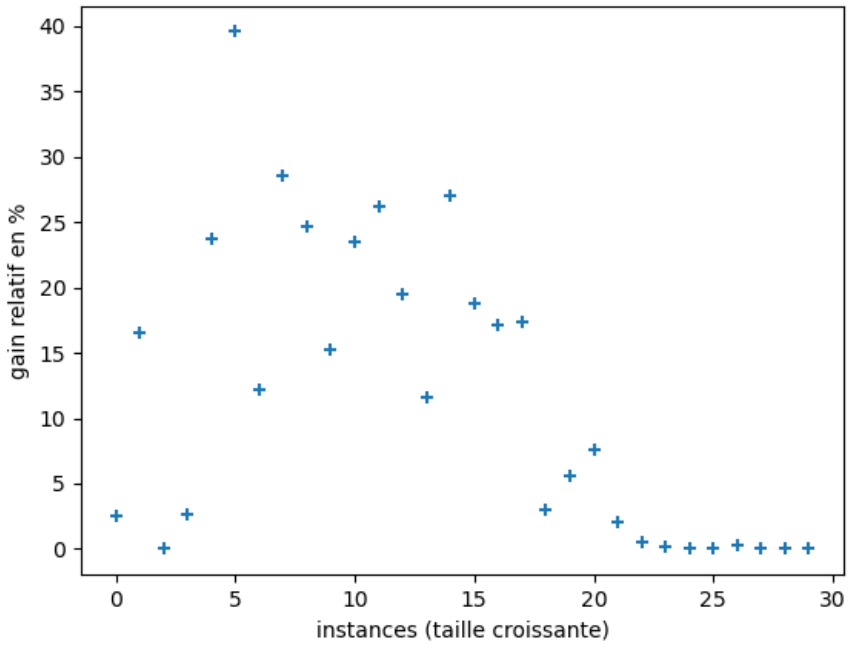


Figure 1: Utilité de la recherche locale

Instance	Valeur d'une solution réalisable	Valeur après recherche locale
100_kroA_3.tsp	2.4865754962e6	2.0191641228e6
100_kroA_6.tsp	1.1491332684e6	952716.5199
100_kroA_9.tsp	683073.0031	564329.9075
10_ulysses_3.tsp	140.5448	136.9953
10_ulysses_6.tsp	66.0375	55.1194
10_ulysses_9.tsp	33.2919	33.2919
14_burma_3.tsp	120.0661	116.9021
14_burma_6.tsp	56.5284	43.1295
14_burma_9.tsp	34.3653	20.7624
202_gr_3.tsp	88428.673	85823.3338
202_gr_6.tsp	42755.1602	40370.8459
202_gr_9.tsp	29285.586	27057.6787
22_ulysses_3.tsp	611.1139	536.6424
22_ulysses_6.tsp	185.7196	132.6357
22_ulysses_9.tsp	121.8414	91.8208
26_eil_3.tsp	3300.0705	2798.0291
26_eil_6.tsp	1504.4338	1151.5283
26_eil_9.tsp	1100.5689	812.3249
30_eil_3.tsp	4168.7894	3358.7788
30_eil_6.tsp	1641.5449	1451.5468
30_eil_9.tsp	1108.4111	808.6163
318_lin_3.tsp	3.0096253313e7	2.9468034057e7
318_lin_6.tsp	1.4078426246e7	1.39988549e7
318_lin_9.tsp	8.365114105e6	8.350554267e6
400_rd_3.tsp	1.036858307e7	1.036858307e7
400_rd_6.tsp	4.215208819e6	4.213273659e6
400_rd_9.tsp	3.007306539e6	2.998634845e6
532_att_3.tsp	1.2817733074e8	1.2806217564e8
532_att_6.tsp	5.12207162e7	5.120306943e7
532_att_9.tsp	2.396545131e7	2.396545131e7

Table 1: Importance de la recherche locale

Les garanties de performances a posteriori de cette heuristique sont présentées dans la section Résultats, tableaux 2a, 3, 4.

Résultats

Les valeurs sont arrondies au centième dans les tableaux 2a, 2b et 2c. Les figures 2, 3 et 4 montrent les meilleures valeurs entières obtenues avec les différentes méthodes, ainsi que la meilleure relaxation et la meilleure valeur du problème statique.

Les figures 5, 6 et 7 montrent le nombre d'instances résolues, en fonction de la méthode utilisée, avec une limite de temps de 30 secondes parmi les instances du tableau 1. On remarque que très peu d'instances sont résolues jusqu'à l'optimalité et les gaps sont élevés (on passe de 6 à 9 instances résolues avec un gap maximal de 70 %). On note que les temps pour l'heuristique correspondent tous au temps maximal imparti : la recherche locale tourne jusqu'à interruption. L'heuristique permet de trouver une solution réalisable pour la quasi-totalité des instances, la dualisation est la méthode la plus fiable sinon : les instances sont résolues plus rapidement et une solution réalisable est trouvée pour un plus grand nombre d'instances.

Pistes d'améliorations

Plusieurs idées d'améliorations ont été explorées.

- Pour rompre les symétries du modèle, nous avons tenté d'ajouter la contrainte $y_{11} = 1$ signifiant que le premier point est dans le premier cluster, sans perte de généralité. La table 3 donne les résultats obtenus pour cette modification.
- Nous avons essayé de réduire le temps de calcul pour les méthodes de *branch and cut* et de plans coupants en donnant au modèle CPLEX une solution initiale (hot start). La solution choisie pour cette amélioration

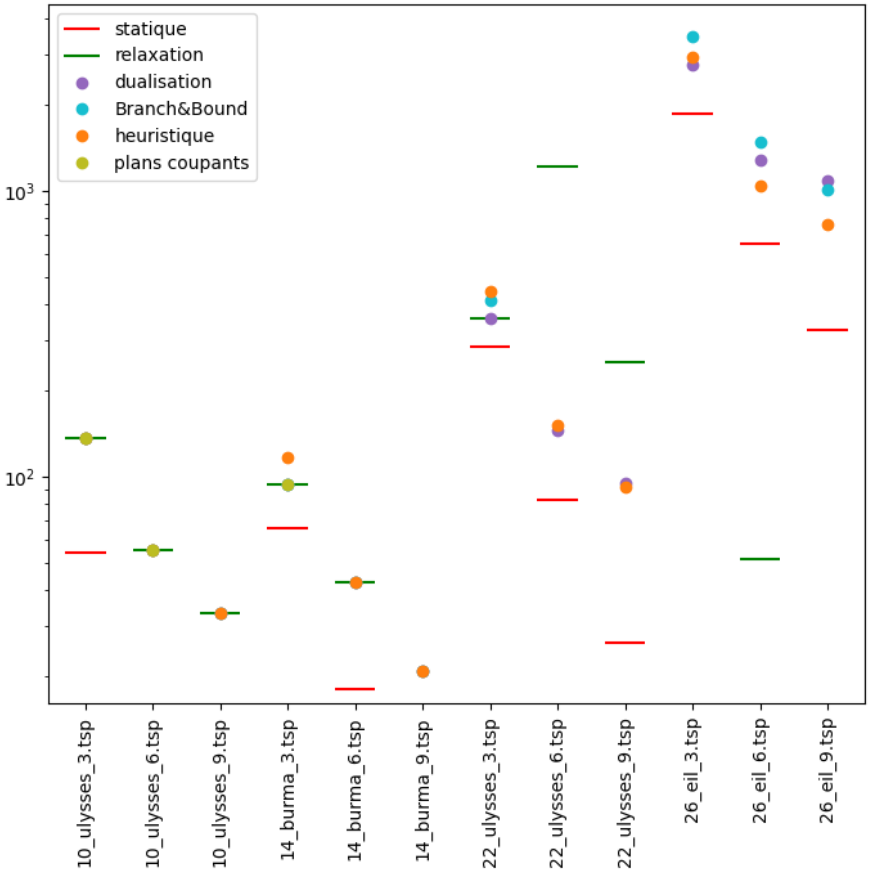


Figure 2: Valeur de la meilleure solution obtenue en 30 secondes

potentielle est la meilleure solution renvoyée par l'heuristique. La table 4 donne les résultats obtenus pour cette modification.

On constate que les résultats pour les plans coupants n'ont pas changé : très peu d'instances ont une solution réalisable en 30 secondes.

Pour le *branch and cut*, les résultats s'en trouvent légèrement améliorés : deux instances supplémentaires ont une solution réalisable (en fait celle fournie par l'heuristique...) et les gaps sont généralement plus serrés. Certaines instances donnent de meilleurs résultats (`26_eil_6.tsp`) avec le *branch and cut + hot start* qu'avec la dualisation simple, mais cela est vraisemblablement uniquement liés aux bonnes performances de l'heuristique car aucune amélioration n'est faite dans le *branch and cut*.

Pour aller plus loin, il serait intéressant d'initialiser de même les variables de la dualisation avec les valeurs données par l'heuristique pour les variables primales et calculer les valeurs initiales à donner aux variables duales grâce aux écarts complémentaires. Il est toutefois à noter que cette méthode suppose que l'on connaisse les solutions données par l'heuristique. Pour un calcul en partant de zéro, il faut donc compter dans le temps de calcul, à la fois le temps passé dans l'heuristique et le temps passé dans la partie de programmation linéaire.

- Finalement, après avoir constaté que certaines instances ont une valeur pour la meilleure relaxation nulle (ce qui rend le gap très grand et la résolution inefficace), nous avons tenté d'ajouter une nouvelle contrainte dans les trois modèles : on coupe les solutions de valeur inférieure à celle de la relaxation statique. La table 5 expose les résultats obtenus pour cette modification. On constate que sur certaines instances la valeur de la meilleure relaxation du problème robuste est effectivement supérieure à celle qu'on obtenait précédemment (`14_burma_9` par exemple). Toutefois, pour certaines instances, la valeur de la relaxation se retrouve dégradée (`14_burma_6` par exemple).

  D'autre part, certaines instances résolues par le *branch and cut* avec le modèle de base ne le sont plus suite à l'ajout de cette contrainte, et les meilleures solutions robustes réalisables obtenues avec cette modification sont globalement de moins bonne qualité que les précédentes : les gaps sont plus serrés mais les meilleures solutions entières sont moins bonnes.

Les figures 10 et 11 donnent les meilleures solutions robustes trouvées,

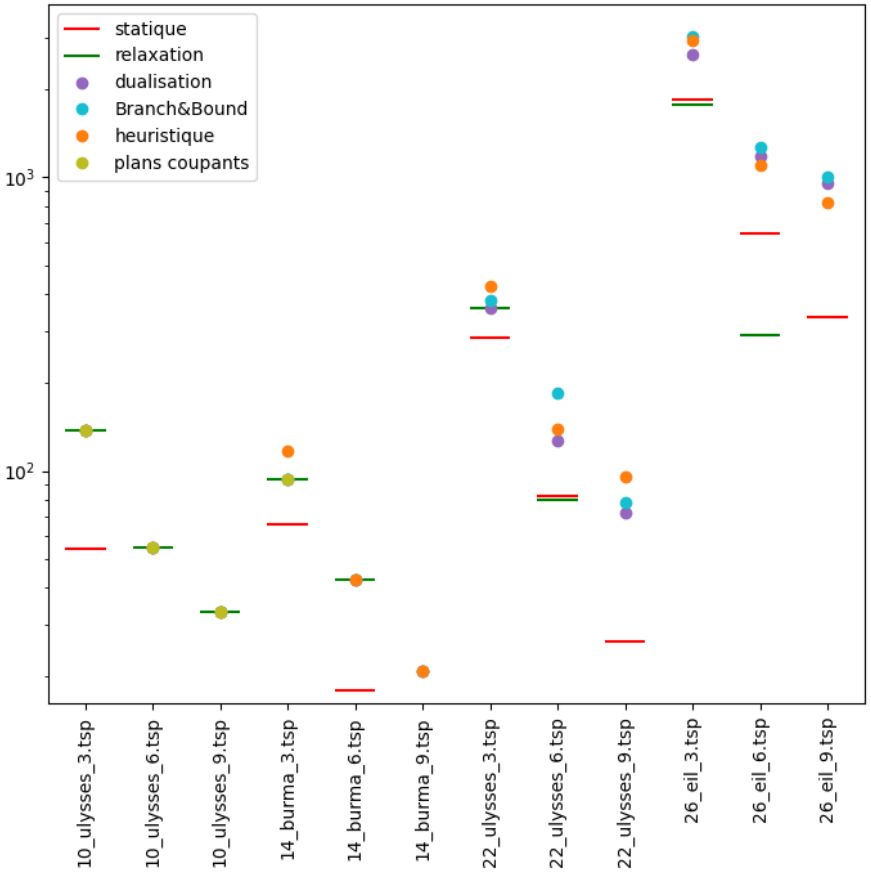


Figure 3: Valeur de la meilleure solution obtenue en 2 minutes

Conclusion

Les heuristiques déterministes sont très rapides, mais du fait des contraintes de robustesse, la difficulté à construire des solutions réalisables rend secondaire la recherche d'une solution de moindre coût. La recherche locale permet de remédier en partie à ce problème, et on pourrait peut-être la rendre meilleure avec un peu plus de sophistication, mais elle atteint probablement ses limites sur les grandes instances. Cette méthode approchée a toutefois l'avantage de nous permettre de contrôler le temps de calcul, qui explose très rapidement avec les méthodes exactes, y compris pour le problème statique.

Parmi les méthodes exactes, la dualisation semble la plus performante sur l'ensemble des instances. Sans surprise, les plans coupants sont dominés par le *branch and cut*. Les améliorations essayées pour accélérer les résolutions exactes n'ont pas d'effet significatif, sur la période de temps de 30s testée du moins : l'optimum est atteint un peu plus vite quand il est atteint, mais on ne résout pas plus d'instances.

Pour les améliorations consistant à ajouter des contraintes supplémentaires, que ce soit sur $y_{1,1}$ ou la valeur du problème, on pourrait utiliser les UserCut avec les Callbacks de CPLEX au lieu de les ajouter comme des contraintes du problème.

Le prix de la robustesse est défini ici comme $\frac{\text{valeur robuste} - \text{valeur statique}}{\text{valeur robuste}}$. Il représente la proportion de la valeur robuste (*100 pour l'avoir en pourcentage) qui est due à la prise en compte de la robustesse. Les instances étant légèrement différentes à sommets égaux, on ne peut pas vraiment bien les comparer, mais il semble que PR augmente avec K à instance (presque) égale, ce qui semble contre-intuitif au premier abord, puisque le problème devrait être plus facile avec un K plus grand. Mais cela peut vouloir dire que le bénéfice d'avoir plus de clusters possibles est plus faible en version robuste qu'en version statique. De façon générale, le PR varie beaucoup selon les instances, entre et 20 et 98%, et il est souvent conséquent.

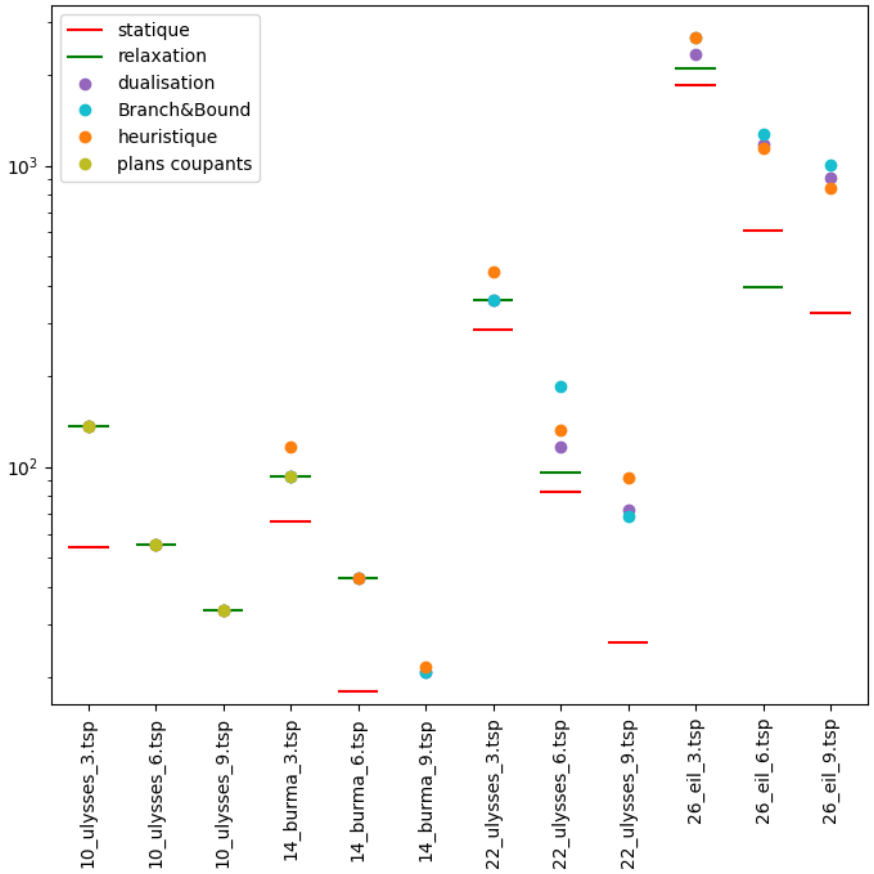


Figure 4: Valeur de la meilleure solution obtenue en 5 minutes

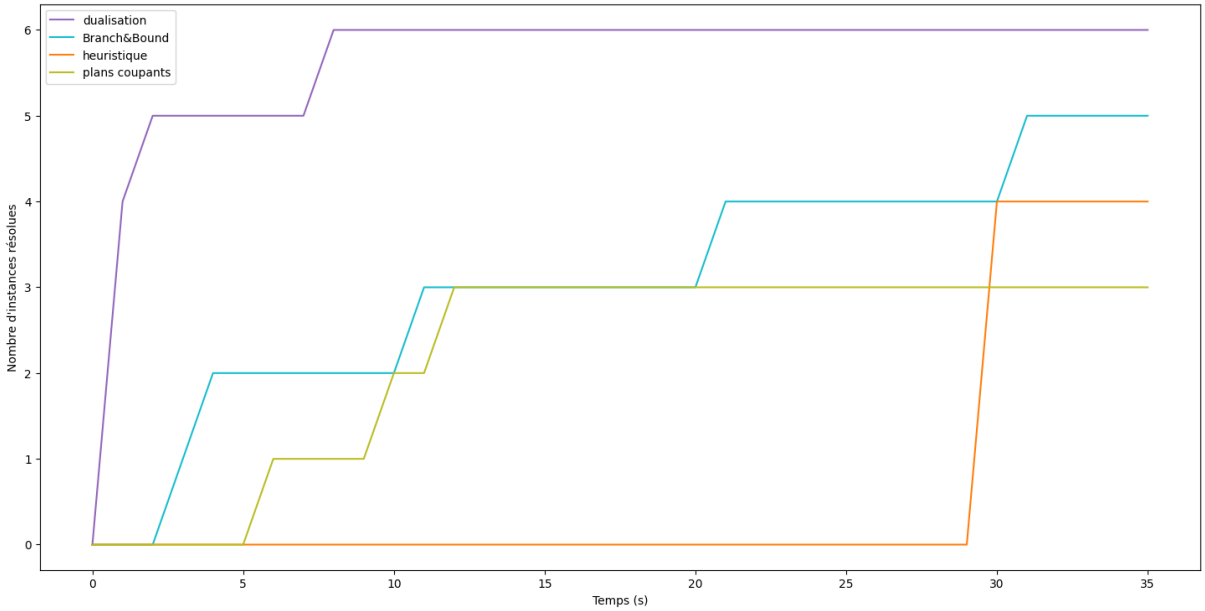


Figure 5: Nombre d'instances résolues jusqu'à l'optimalité en 30 secondes

Instance	PR	Statique			Dualisation			Plans coupants			Branch & Cut			Heuristique		
		Time	Best Sol	Gap	Time	Best Sol	Gap	Time	Best Sol	Gap	Time	Best Sol	Gap	Best relaxation	Best Sol	Gap
10_ulysse_3.tsp	60.3%	0.04	54.35	0	0.46	137.0	0	9.16	137.0	0	2.51	137.0	0	137.0	137.0	0
10_ulysses_6.tsp	86.9%	0.089	7.22	0	0.19	55.12	0	5.44	55.12	0	20.18	55.12	0	55.12	55.12	0
10_ulysses_9.tsp	97.8%	0.082	0.72	0	1.30	33.29	0	/	/	/	10.39	33.29	0	33.29	33.29	0
14_burma_3.tsp	29.1%	0.28	66.21	0	0.57	93.39	0	11.93	93.39	0	3.11	93.39	0	93.39	116.90	20.1%
14_burma_6.tsp	58.0%	0.15	17.96	0	0.94	42.74	0	/	/	/	30	42.74	100%	42.74	42.74	0
14_burma_9.tsp	<77.3%	0.33	4.72	0	30	20.76	100%	/	/	/	30	20.76	100%	0	20.76	0
22_ulysses_3.tsp	20.8%	1.05	284.21	0	8.99	358.64	0	/	/	/	30	414.59	58.1%	358.61	445.93	19.6%
22_ulysses_6.tsp	<43.2%	6.42	82.63	0	30	145.44	64.5%	/	/	/	/	/	/	51.62	150.89	65.8%
22_ulysses_9.tsp	<71.4%	1.93	26.22	0	30	94.68	100%	/	/	/	/	/	/	0	91.82	100%
26_eil_3.tsp	<32.6%	14.55	1850.42	0	30	2743.86	55.5%	/	/	/	30	3475.64	94.9%	1221.36	2937.27	58.4%
26_eil_6.tsp	?	30	648.83	38.6%	30	1275.03	78.4%	/	/	/	30	1485.79	100%	250.72	1037.62	75.8%
26_eil_9.tsp	?	30	336.72	43.4%	30	1085.72	100%	/	/	/	30	1006.51	100%	0	762.82	100%

(a) Résultats pour une limite de 30 secondes

Instance	PR	Statique			Dualisation			Plans coupants			Branch & Cut			Heuristique		
		Time	Best Sol	Gap	Time	Best Sol	Gap	Time	Best Sol	Gap	Time	Best Sol	Gap	Best relaxation	Best Sol	Gap
10_ulysses_9.tsp	97.8%	0.082	0.72	0	1.30	33.29	0	64.7	33.29	0	10.39	33.29	0	33.29	33.29	0
14_burma_6.tsp	58.0%	0.15	17.96	0	0.94	42.74	0	/	/	/	120	42.74	87.3%	42.74	42.74	0
14_burma_9.tsp	<77.3%	0.33	4.72	0	120	20.76	100%	/	/	/	120	20.76	100%	0	20.76	0
22_ulysses_3.tsp	20.8%	1.05	284.21	0	8.99	358.64	0	/	/	/	120	381.61	24.3%	358.61	445.93	19.6%
22_ulysses_6.tsp	< 35.1	6.42	82.63	0	120	127.26	37.0%	/	/	/	120	184.86	100%	80.16	150.89	65.8%
22_ulysses_9.tsp	<63.6	1.93	26.22	0	120	72.05	100%	/	/	/	120	78.49	100%	0	91.82	100%
26_eil_3.tsp	< 29.6	14.55	1850.42	0	120	2628.78	32.7%	/	/	/	120	3028.19	74.1%	1768.90	2937.27	58.4%
26_eil_6.tsp	?	120	641.96	26.2%	120	1181.48	75.4%	/	/	/	120	1268.24	100%	290.52	1037.62	75.8%
26_eil_9.tsp	?	120	334.63	33.9%	120	952.1	100%	/	/	/	120	1006.51	100%	0	762.82	100%

(b) Résultats pour une limite de 2 minutes

Instance	PR	Statique			Dualisation			Plans coupants			Branch & Cut			Heuristique		
		Time	Best Sol	Gap	Time	Best Sol	Gap	Time	Best Sol	Gap	Time	Best Sol	Gap	Best relaxation	Best Sol	Gap
14_burma_6.tsp	58.0%	0.15	17.96	0	0.94	42.74	0	/	/	/	300	42.74	22.4%	42.74	42.74	0
14_burma_9.tsp	77.3%	0.33	4.72	0	300	20.76	100%	/	/	/	300	20.76	100%	0	20.76	0
22_ulysses_3.tsp	20.8%	1.05	284.21	0	8.99	358.64	0	/	/	/	263.8	356.64	0	358.61	445.93	19.6%
22_ulysses_6.tsp	<29.1%	6.42	82.63	0	300	116.53	17.9%	/	/	/	300	184.86	100%	95.69	132.64	27.9%
22_ulysses_9.tsp	<61.8%	1.93	26.22	0	300	72.05	100%	/	/	/	300	68.64	100%	0	91.82	100%
26_eil_3.tsp	<20.9%	14.55	1850.42	0	300	2340.29	9.8%	/	/	/	300	2669.78	60.2%	2109.83	2656.06	20.6%
26_eil_6.tsp	?	300	608.72	13.0%	300	1181.48	66.6%	/	/	/	300	1268.24	99.9%	395.09	1147.21	65.6%
26_eil_9.tsp	?	300	324.94	22.2%	300	910.5	100%	/	/	/	300	1006.51	100%	0	762.82	100%

(c) Résultats pour une limite de 5 minutes

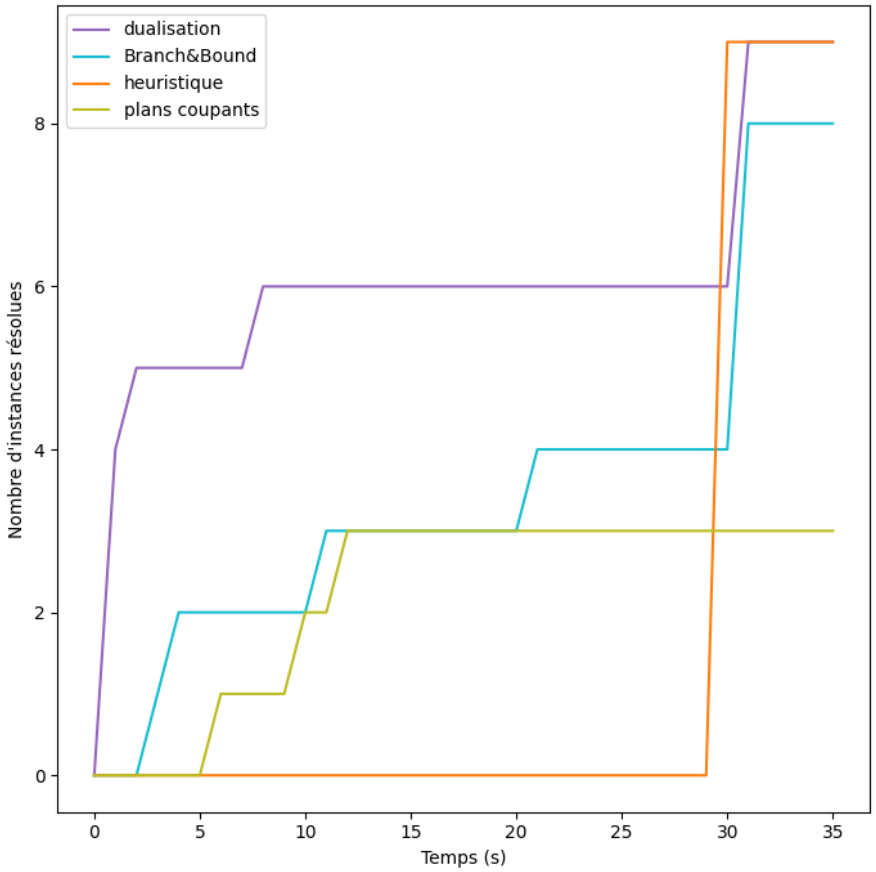


Figure 6: Nombre d'instances résolues jusqu'à un gap de 70% en 30 secondes

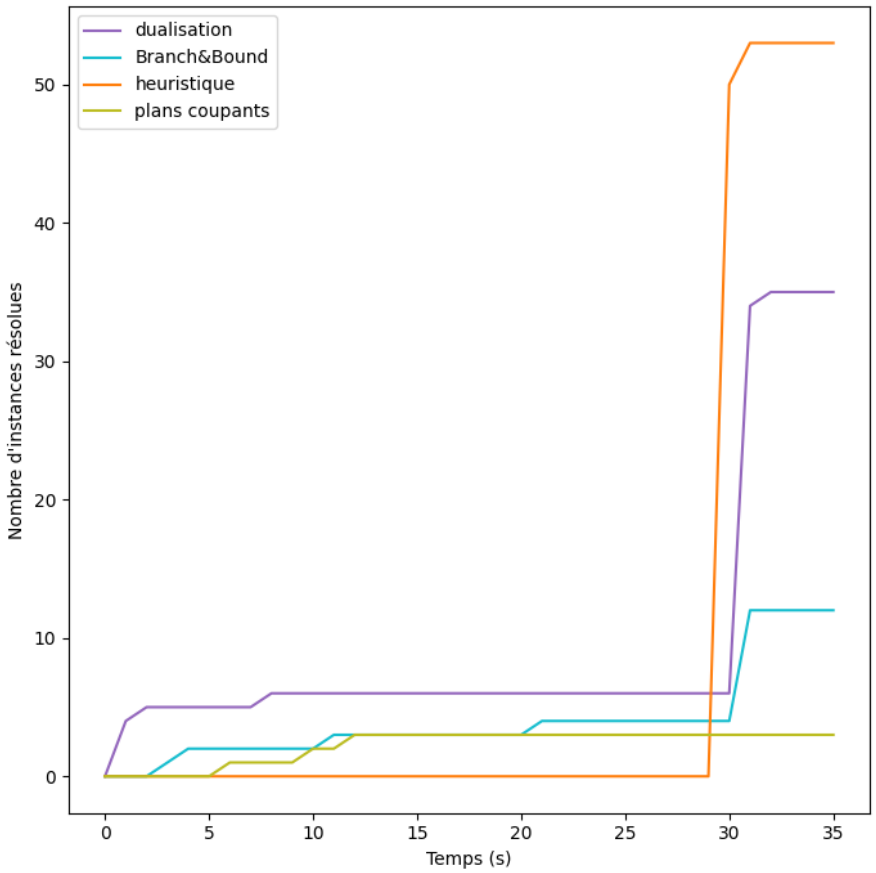


Figure 7: Nombre d'instances avec une solution réalisable en 30 secondes

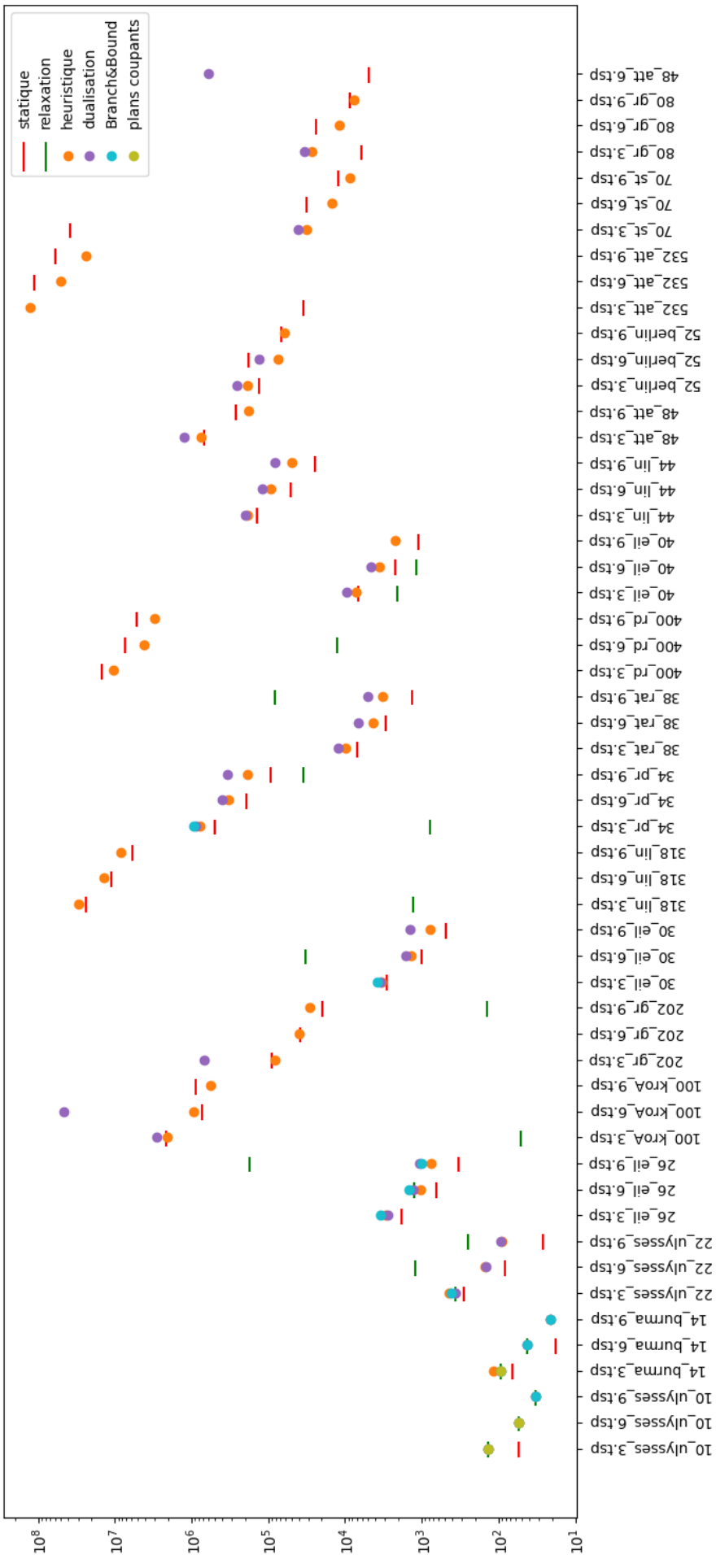


Figure 8: Valeur de la meilleure solution obtenue en 30 secondes

Instance	Statique			Dualisation			Plans coupants				Branch & Cut			
	Time	Best Sol	Gap	Time	Best Sol	Best relax	Gap	Time	Best Sol	Best relax	Gap	Time	Best Sol	Best relax	Gap
10_ulysse_3.tsp	0.13	54.35	0	0.15	137.0	137.0	0	4.29	137.0	137.0	0	1.55	137.0	137.0	0
10_ulysses_6.tsp	0.089	7.22	0	0.37	55.12	55.12	0	3.0	55.12	55.12	0	4.5	55.12	55.12	0
10_ulysses_9.tsp	0.07	0.72	0	1.40	33.29	33.29	0	14.75	33.29	33.29	0	1.2	33.29	33.29	0
14_burma_3.tsp	0.16	66.21	0	0.25	93.39	93.39	0	2.79	93.39	93.39	0	3.7	93.39	93.39	0
14_burma_6.tsp	0.13	17.96	0	0.59	42.74	42.74	0	/	/	/	/	30	42.74	28.07	34.3%
14_burma_9.tsp	0.19	4.72	0	30	20.76	0	100%	/	/	/	/	30	21.45	0	100%
22_ulysses_3.tsp	0.48	284.21	0	10.24	358.64	358.64	0	/	/	/	/	30	447.52	193.62	34.3%
22_ulysses_6.tsp	6.84	82.63	0	30	122.69	49.55	59.6%	/	/	/	/	30	257.83	0	100%
22_ulysses_9.tsp	1.19	26.22	0	30	93.35	0	100%	/	/	/	/	/	/	/	/
26_eil_3.tsp	18.73	1850.42	0	30	2812.60	1401.51	50.2%	/	/	/	/	30	3303.43	754.17	77.2%
26_eil_6.tsp	30	651.79	45.3%	30	1410.59	189.48	86.6%	/	/	/	/	/	/	/	/
26_eil_9.tsp	30	320.17	41.5%	30	1189.96	0	100%	/	/	/	/	/	/	/	/

Table 3: Résultats en 30 secondes en ajoutant la contrainte $y_{1,1} = 1$

Instance	Heuristique	Plans coupants			Branch & Cut		
	Best Sol	Time	Best Sol	Gap	Time	Best Sol	Gap
10_ulysse_3.tsp	137.0	5.6	137.0	0	2.3	137.0	0
10_ulysses_6.tsp	55.12	5.28	55.12	0	25.0	55.12	0
10_ulysses_9.tsp	33.29	/	/	/	7.3	33.29	0
14_burma_3.tsp	116.90	10.73	93.39	0	3.7	93.39	0
14_burma_6.tsp	43.31	/	/	/	30	42.74	100%
14_burma_9.tsp	20.76	/	/	/	30	20.73	100%
22_ulysses_3.tsp	531.84	/	/	/	30	415.93	60.0%
22_ulysses_6.tsp	165.67	/	/	/	30	165.67	100%
22_ulysses_9.tsp	64.97	/	/	/	30	64.97	100%
26_eil_3.tsp	2906.35	/	/	/	30	2906.35	87.9%
26_eil_6.tsp	1100.52	/	/	/	30	1100.52	100%
26_eil_9.tsp	812.32	/	/	/	30	812.32	100%

Table 4: Résultats en 30 secondes avec solution initiale donnée par l'heuristique

Instance	Statique				Dualisation				Plans coupants				Branch & Cut			
	Time	Best Sol	Best relax	Gap	Time	Best Sol	Best relax	Gap	Time	Best Sol	Best relax	Gap	Time	Best Sol	Best relax	Gap
10_ulysse_3.tsp	0.048	54.35	54.35	0	0.214	137.0	137.0	0	7.79	137.0	137.0	0	2.15	137.0	137.0	0
10_ulysses_6.tsp	0.089	7.22	7.22	0	0.227	55.12	55.12	0	6.60	55.12	55.12	0	28.52	55.12	55.12	0
10_ulysses_9.tsp	0.082	0.72	0.72	0	30	33.29	0.72	97.8%	25.58	33.29	33.29	0	6.71	33.29	33.29	0
14_burma_3.tsp	0.28	66.21	66.21	0	0.65	93.39	93.39	0	20.46	93.39	93.39	0	12.47	93.39	93.39	0
14_burma_6.tsp	0.15	17.96	17.96	0	1.38	42.74	42.74	0	/	/	/	/	30	43.13	17.96	58.4%
14_burma_9.tsp	0.33	4.72	4.72	0	30	20.76	4.72	77.3%	/	/	/	/	30	20.76	4.72	77.3%
22_ulysses_3.tsp	1.05	284.21	284.21	0	30	358.64	337.78	5.8%	/	/	/	/	30	414.59	284.2	31.5%
22_ulysses_6.tsp	6.42	82.63	82.63	0	30	156.23	82.63	47.1%	/	/	/	/	30	242.75	82.63	66.0%
22_ulysses_9.tsp	1.93	26.22	26.22	0	30	77.34	26.22	66.1%	/	/	/	/	/	/	/	/
26_eil_3.tsp	14.55	1850.42	1850.42	0	30	2769.95	1850.42	33.2%	/	/	/	/	30	3285.20	1850.42	43.7%
26_eil_6.tsp	30	637.81	395.88	37.9%	30	1377.09	395.88	71.3%	/	/	/	/	/	/	/	/
26_eil_9.tsp	30	334.96	190.10	43.2%	30	1166.14	190.10	83.7%	/	/	/	/	30	928.18	190.10	79.5%

Table 5: Résultats en 30 secondes en coupant les solutions de valeur inférieure à la relaxation statique

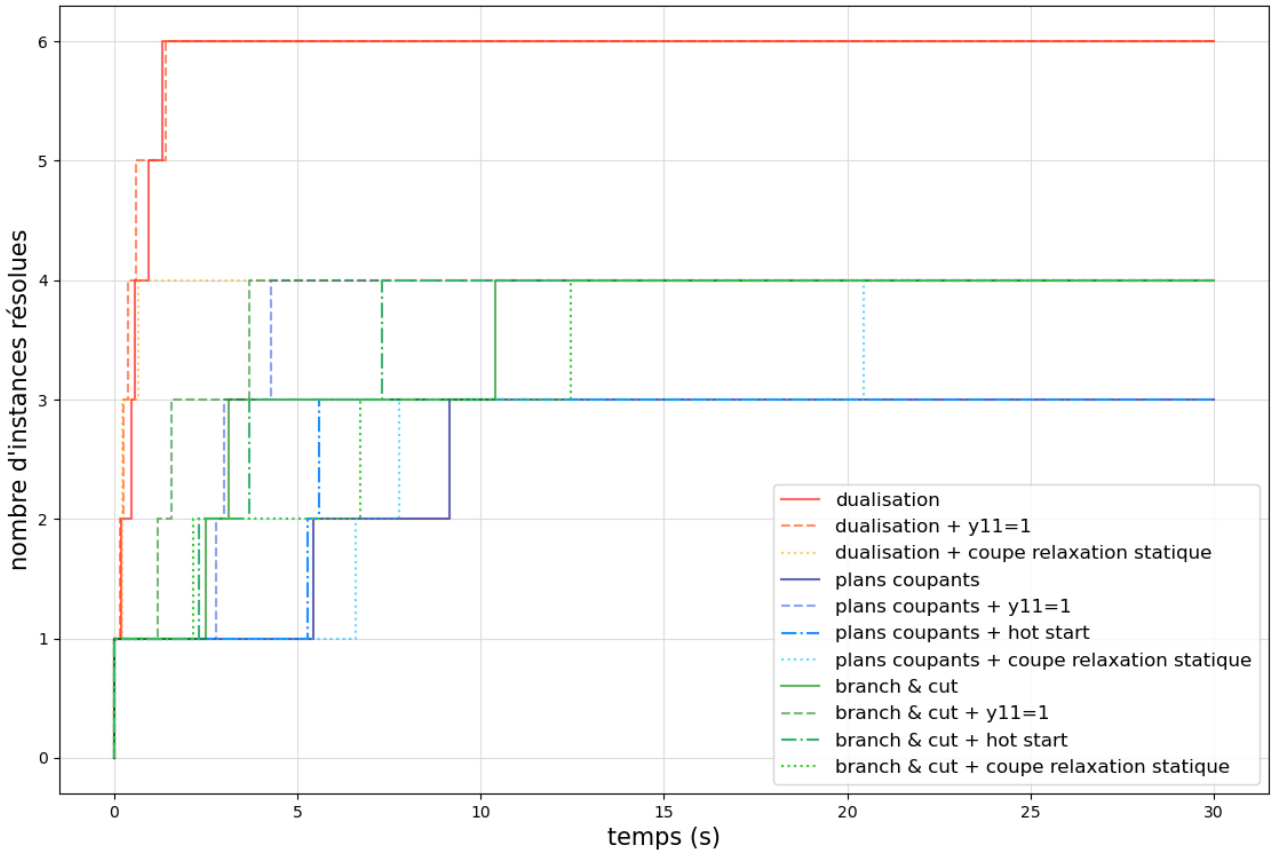
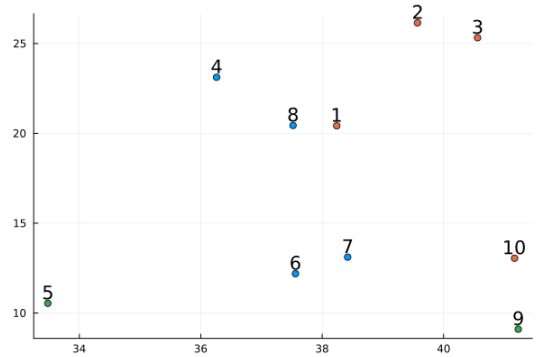
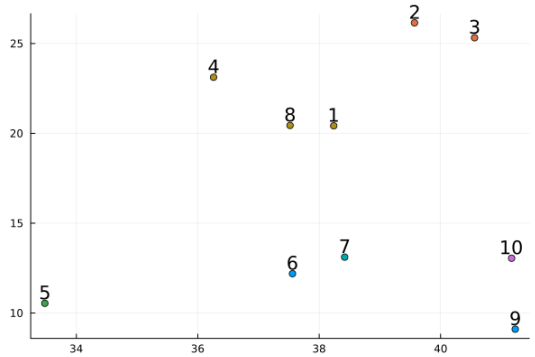


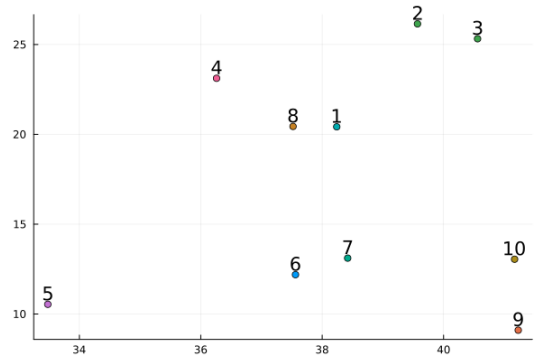
Figure 9: Comparaison des différentes améliorations sur 30s



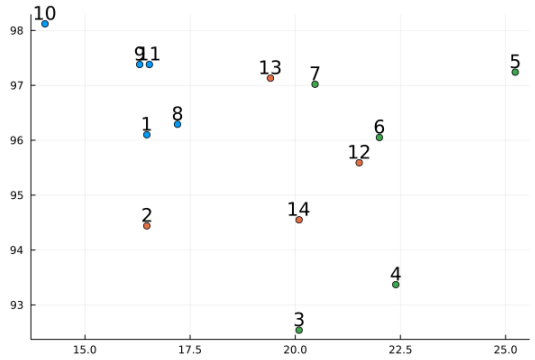
(a) 10_ulysses_3.tsp



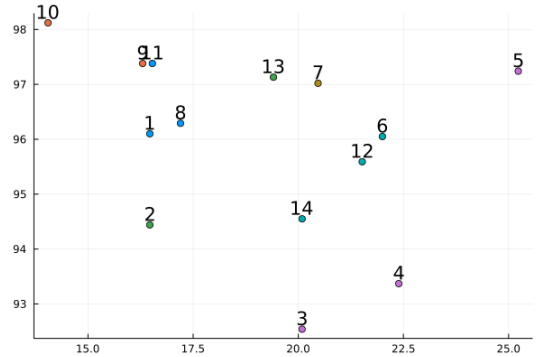
(b) 10_ulysses_6.tsp



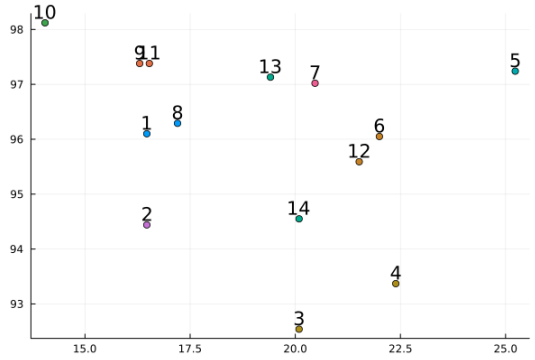
(c) 10_ulysses_9.tsp



(d) 14_burma_3.tsp

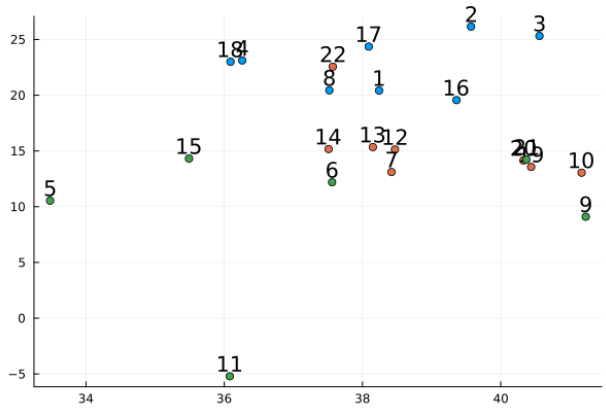


(e) 14_burma_6.tsp

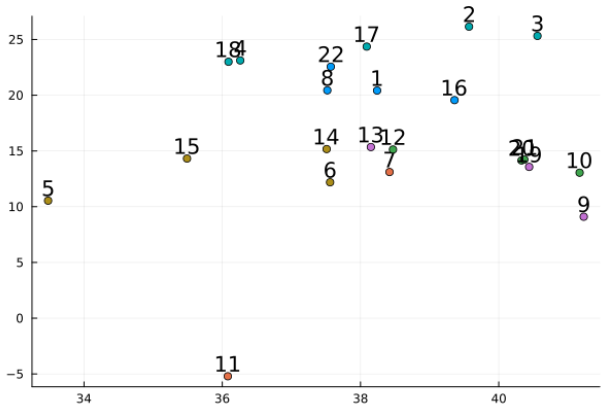


(f) 14_burma_9.tsp

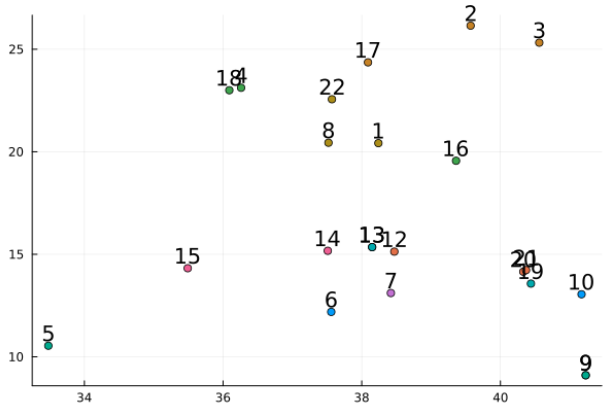
Figure 10: Solutions sur les petites instances



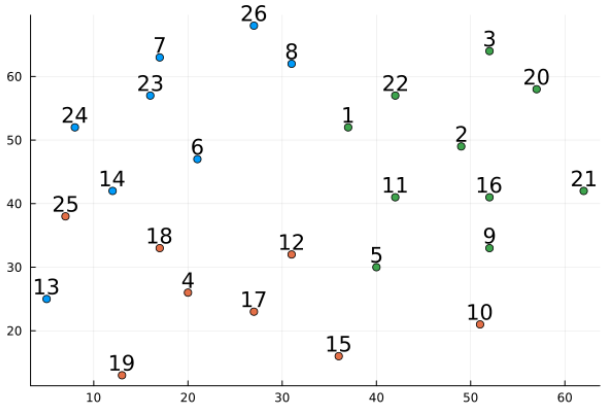
(a) 22_ulysses_3.tsp



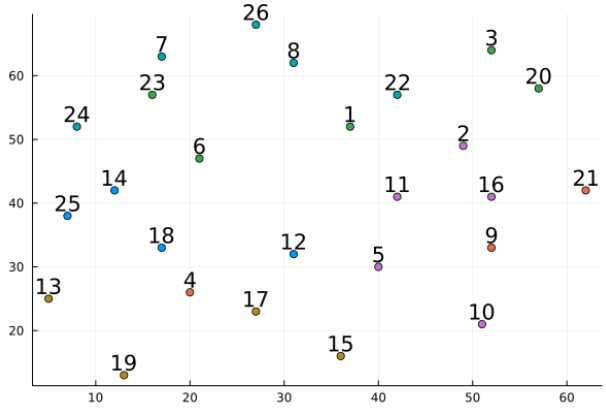
(b) 22_ulysses_6.tsp



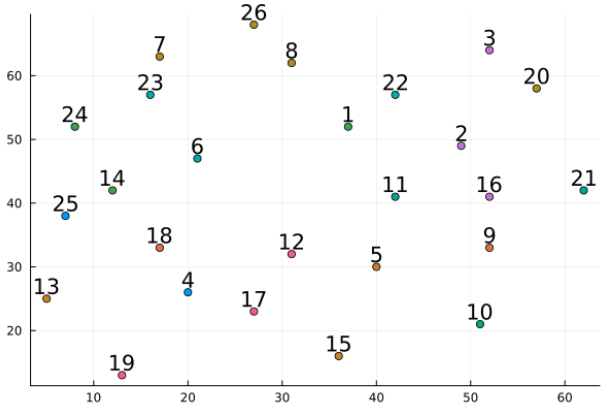
(c) 22_ulysses_9.tsp



(d) 26_eil_3.tsp



(e) 26_eil_6.tsp



(f) 26_eil_9.tsp

Figure 11: Solutions sur les petites instances