Rapport PROJ, partie théorique

Michaël LAPORTE et Guillaume BRIVARY

Master Parisien de Recherche Opérationnelle
15 février 2025

Table des matières

1 Rappel du problème **2**

2 Modélisation théorique **3**

2.1 Modélisation du problème compacte déterministe 3

2.2 Modélisation du problème robuste 4

2.3 Résolution par des plans coupants 5

2.4 Résolution par dualisation 6

3 Résolution numérique **9**

3.1 Principe 9

3.2 Option : Heuristique 9

3.3 Résultats 11

3.4 Conclusion 18

3.5 Annexe 18

1 Rappel du problème

Problème statique

On considère un problème de tournées de véhicules dans lequel un entrepôt doit livrer des vaccins à un ensemble de clients. Le problème est caractérisé par :

— un ensemble de sommets $V = \{1, ..., n\}$, tel que le sommet 1 correspond à l'entrepôt des véhicules où se trouve le stock de vaccins, et les autres sommets représentent les clients à livrer. Soit $A = \{ij \in V^2 | i \neq j\}$;

— une durée $t_{ij} \in \mathbb{N}$ associée à chaque couple de sommets $(i, j) \in A$, correspondant au temps minimal pour qu'un véhicule se rende du sommet i au sommet j ;

— une demande $d_i \in \mathbb{N}$ associée à chaque sommet $i \in V \setminus \{1\}$, correspondant au nombre de vaccins à livrer au client i ;

— une capacité maximale $C \in \mathbb{N}$, correspondant au nombre maximal de vaccins qu'il est possible de stocker dans un véhicule. Cette capacité est identique pour tous les véhicules.

On définit une tournée T comme une suite de sommets $T = \{t_1, t_2, ..., t_{|T|}\}$ telle que :

— la tournée débute et termine à l'entrepôt $(t_1 = t_{|T|} = 1)$;

— la somme des demandes des clients visités n'excède pas la capacité d'un véhicule :

$$\sum_{i=2}^{|T|-1} d_{t_i} \leq C;$$

— tous les clients visités sont différents :

$$t_i \neq t_j, \quad \forall i, j \in \{2, ..., |T| - 1\}, i \neq j.$$

L'objectif du problème statique consiste à déterminer un ensemble de tournées permettant de visiter l'ensemble des clients tout en minimisant la durée totale des trajets.

Précisions :

— Le nombre de véhicules n'est pas limité et n'intervient pas dans la fonction objectif.

— Chaque client doit être visité dans exactement une tournée. Par exemple, une solution comportant les deux tournées $\{1, 2, 3, 1\}$ et $\{1, 4, 5, 6, 1\}$ sera admissible seulement si $d_2 + d_3 \leq C$ et $d_4 + d_5 + d_6 \leq C$. La valeur de l'objectif associée à cette solution est

$$t_{1,2} + t_{2,3} + t_{3,1} + t_{1,4} + t_{4,5} + t_{5,6} + t_{6,1}.$$

Problème robuste

Nous souhaitons résoudre une version robuste du problème statique, car il est possible que les valeurs choisies pour les temps de trajets soient sous-évaluées.

Incertitude sur les distances

À chaque sommet $i \in V$ est associé un entier $\widehat{t}_i$. Nous supposons que l'augmentation de la durée de trajet entre deux sommets i et j peut au maximum être $\widehat{t}_i + \widehat{t}_j + 2\widehat{t}_i\widehat{t}_j$. L'augmentation totale des durées est limitée par un paramètre $T \in \mathbb{N}$.

L'ensemble des valeurs que peuvent prendre les durées est défini comme suit :

$$\mathcal{U} = \left\{ \{t'_{ij} = t_{ij} + \delta^1_{ij}(\widehat{t_i} + \widehat{t_j}) + \delta^2_{ij}\widehat{t_i}\widehat{t_j}\}_{ij \in A} \;\middle|\; \sum_{ij \in A} \delta^1_{ij} \leq T,\ \sum_{ij \in A} \delta^2_{ij} \leq T^2,\ \delta^1_{ij} \in [0,1],\ \delta^2_{ij} \in [0,2],\ \forall ij \in A \right\}.$$

Remarque : Seules δ^1_{ij} et δ^2_{ij} sont des variables. En effet, t_{ij}, $\widehat{t_i}$ et T sont des données du problème.

2 Modélisation théorique

Avant de nous lancer dans la résolution de ce problème intéressons nous à sa modélisation :

2.1 Modélisation du problème compacte déterministe

Pour modéliser le problème nous allons introduire 3 types de variables :

$$\forall i, j \in V^2, \forall T \in [\![1, |V|]\!], \quad x_{ijT} = \begin{cases} 1 & \text{si l'\textbf{arc} ij fait partie de la tournée T} \\ 0 & \text{sinon} \end{cases} \tag{1}$$

$$\forall T \in [\![1, |V|]\!], \quad b_T = \begin{cases} 1 & \text{si la tournée T contient au moins une arête} \\ 0 & \text{si la tournée T est vide} \end{cases} \tag{2}$$

$$\forall i \in V, \forall T \in [\![1, |V|]\!], \quad u_{iT} \text{ est le compteur d'ordre de chaque sommet i dans la tournée T} \tag{3}$$

La polynomialité du problème vient du fait que l'on peut majorer le nombre de tournée par le nombre de sommets grâce à la contrainte expliquée dans l'énoncé [1], ainsi on se retrouve avec n^3 variables issues de (1), n variables issues de (2) et n^2 variables issues de (3).

En utilisant les variables ainsi définies on peut écrire le problème sous la forme suivante :

$$\min_{x,b,u} \sum_{T \in [\![1,|V|]\!]} \sum_{\substack{i,j \in V^2 \\ i \neq j}} t_{ij} x_{ijT}$$

Sous contraintes :

1. Un sommet ne peut être visité que dans une seule tournée

$$\textbf{s.c.} \quad \sum_{T\in[\![1,|V|]\!]} \sum_{j\in N^+(i)} x_{ijT} = 1 \quad \forall i \in V \setminus \{1\} \tag{4}$$

$$\sum_{T\in[\![1,|V|]\!]} \sum_{i\in N^-(i)} x_{ijT} = 1 \quad \forall j \in V \setminus \{1\} \tag{5}$$

$$\sum_{\substack{i,j\in V^2 \\ i\neq j, j\neq 1}} x_{ijT} d_j \leq C \quad \forall T \in [\![1,|V|]\!] \tag{6}$$

$$u_{iT} - u_{jT} + 1 \leq (n-1)(1-x_{ijT}) \quad \forall T \in [\![1,|V|]\!], \forall i,j \in V^2, i\neq j, i\neq 1, j\neq 1 \tag{7}$$

$$b_T \geq x_{1jT} \quad \forall T \in [\![1,|V|]\!], \forall j \in V \tag{8}$$

$$\sum_{j\in N^+(1)} x_{1jT} = b_T \quad \forall T \in [\![1,|V|]\!] \tag{9}$$

$$\sum_{i\in N^-(1)} x_{i1T} = b_T \quad \forall T \in [\![1,|V|]\!] \tag{10}$$

$$x_{ijT} \in \{0,1\} \quad \forall ij \in V^2, \forall T \in [\![1,|V|]\!] \tag{11}$$

$$b_T \in \{0,1\} \quad \forall T \in [\![1,|V|]\!] \tag{12}$$

$$u_{iT} \in \mathbb{R}_+ \quad \forall i \in V, \forall T \in [\![1,|V|]\!] \text{ et } u_{1T} = 0 \quad \forall T \in [\![1,|V|]\!] \tag{13}$$

Explications des contraintes :

- Les contraintes (5) et (4) forcent les arêtes d'une tournée à entrer/sortir au plus une seule fois d'un sommet, elles permettent aussi d'assurer que tous les sommets sont visités une seule fois par une seule tournée.
- La contrainte (6) permet de s'assurer que la contrainte de capacité sur les tournées est respectée.
- Les contraintes (7) et (13) permettent d'empêcher des solutions avec des tournées non connexes d'apparaître, pour ça on donne à chaque sommet d'une tournée un compteur augmentant strictement à chaque sommet, la contrainte (7) assure la stricte monotonie de ce compteur.
- Les contraintes (8), (9) et (10) permettent d'assurer qu'une tournée commence et termine en 1 (le dépôt), **SI** la tournée n'est pas vide, c'est à dire qu'au moins une arête la constitue [2].
- Les contraintes (11) et (12) sont les contraintes d'intégrité des variables x et b.

2.2 Modélisation du problème robuste

Maintenant modifions notre modélisation compacte afin de résoudre le problème robuste lié à l'incertitude sur les temps de livraison :

2. La contrainte (8) devrait alors s'écrire $b_T \geq x_{ijT} \quad \forall T \in [1,|V|], \forall i,j \in V^2$, cependant les autres contraintes obligent une tournée à sortir de 1, donc si la tournée n'est pas vide elle sort forcément de 1, la contrainte (8) est alors simplifiée sur les arêtes sortant de 1.

$$\min_{x,b,u} \quad \max_{t'_{ij} \in \mathcal{U}} \sum_{T \in [\![1,|V|]\!]} \sum_{\substack{i,j \in V^2 \\ i \neq j}} t'_{ij} x_{ijT}$$

$$\textbf{s.c.} \quad x, b, u \in \{(x, b, u) \text{ vérifiant les contraintes (4) à (13)}\}$$

Avec $\mathcal{U}$ défini dans l'énoncé du problème.

2.3 Résolution par des plans coupants

Pour pouvoir utiliser un algorithme de plans coupants pour résoudre le problème, on commence par reformuler l'objectif afin que la robustesse n'apparaisse plus que dans les contraintes :

$$\min_{x,b,u,z} \ z \tag{14}$$

$$\textbf{s.c.} \ z \geq \sum_{T \in [\![1,|V|]\!]} \sum_{\substack{i,j \in V^2 \\ i \neq j}} t'_{ij} x_{ijT} \ \forall t'_{ij} \in \mathcal{U} \tag{15}$$

$$x, b, u \in \{(x, b, u) \text{ vérifiant les contraintes (4) à (13)}\} \tag{16}$$

On initialise l'ensemble des valeurs que peuvent prendre les durées de la façon suivante :

$$\mathcal{U}^* = \left\{ \{t'_{ij} = t_{ij}\} \middle| ij \in A \right\}.$$

On obtient alors le problème maître (MP) suivant :

$$\min_{x,b,u,z} \ z \tag{17}$$

$$\textbf{s.c.} \ z \geq \sum_{T \in [\![1,|V|]\!]} \sum_{\substack{i,j \in V^2 \\ i \neq j}} t'_{ij} x_{ijT} \ \forall t'_{ij} \in \mathcal{U}^* \tag{18}$$

$$x, b, u \in \{(x, b, u) \text{ vérifiant les contraintes (4) à (13)}\} \tag{19}$$

Avant de pouvoir résoudre le problème maître défini précédemment, il faut traiter le sous-

problème associé :

$$\max_{\delta^1,\delta^2} \sum_{T\in[\![1,|V|]\!]} \sum_{\substack{i,j\in V^2\\ i\neq j}} [t_{ij} + \delta^1_{ij}(\widehat{t_i}+\widehat{t_j}) + \delta^2_{ij}(\widehat{t_i}\widehat{t_j})]x^*_{ijT} \tag{20}$$

$$\textbf{s.c.} \sum_{\substack{i,j\in V^2\\ i\neq j}} \delta^1_{ij} \leq T \tag{21}$$

$$\sum_{\substack{i,j\in V^2\\ i\neq j}} \delta^2_{ij} \leq T^2 \tag{22}$$

$$0 \leq \delta^1_{ij} \leq 1 \tag{23}$$

$$0 \leq \delta^2_{ij} \leq 2 \tag{24}$$

Une solution du problème maître est une solution optimale si :

$$z^* \geq \sum_{T\in[\![1,|V|]\!]} \sum_{\substack{i,j\in V^2\\ i\neq j}} [\delta^{1}_{ij}{}^*(\widehat{t_i}+\widehat{t_j}) + \delta^{2}_{ij}{}^*(\widehat{t_i}\widehat{t_j})]\ x^*_{ijT} \tag{25}$$

$$\text{Avec } z^* \text{ la valeur optimale de (17)} \tag{26}$$

$$\text{Avec } \delta^{1*}, \delta^{2*} \text{ les solutions du problème (20)} \tag{27}$$

Les coupes ajoutées au problème maître par la résolution du sous-problème sont de la forme :

$$z \geq \sum_{T\in[\![1,|V|]\!]} \sum_{\substack{i,j\in V^2\\ i\neq j}} [\delta^{1}_{ij}{}^*(\widehat{t_i}+\widehat{t_j}) + \delta^{2}_{ij}{}^*(\widehat{t_i}\widehat{t_j})]\ x_{ijT} \tag{28}$$

2.4 Résolution par dualisation

En réutilisant la structure de $\mathcal{U}$ on peut réécrire le problème d'une autre façon afin d'isoler le problème interne :

$$\min_{x,b,u} \sum_{T\in[\![1,|V|]\!]} \sum_{\substack{i,j\in V^2\\ i\neq j}} t_{ij}x_{ijT} + \max_{\delta^1,\delta^2} \sum_{T\in[\![1,|V|]\!]} \sum_{\substack{i,j\in V^2\\ i\neq j}} [\delta^1_{ij}(\widehat{t_i}+\widehat{t_j}) + \delta^2_{ij}(\widehat{t_i}\widehat{t_j})]x_{ijT} \tag{29}$$

$$\sum_{\substack{i,j\in V^2\\ i\neq j}} \delta^1_{ij} \leq T \tag{30}$$

$$\sum_{\substack{i,j\in V^2\\ i\neq j}} \delta^2_{ij} \leq T^2 \tag{31}$$

$$0 \leq \delta^1_{ij} \leq 1 \tag{32}$$

$$0 \leq \delta^2_{ij} \leq 2 \tag{33}$$

$$\textbf{s.c.} \quad x,b,u \in \{(x,b,u) \text{ vérifiant les contraintes (4) à (13)}\} \tag{34}$$

Ainsi le problème interne est :

$$\max_{\delta^1,\delta^2} \sum_{T\in[\![1,|V|]\!]} \sum_{\substack{i,j\in V^2\\ i\neq j}} [\delta^1_{ij}(\widehat{t_i}+\widehat{t_j}) + \delta^2_{ij}(\widehat{t_i}\widehat{t_j})]x_{ijT} \tag{35}$$

$$\textbf{s.c.} \quad \delta^1,\delta^2 \in \left\{(\delta^1,\delta^2) \text{ vérifiant les contraintes (30) à (33)}\right\} \tag{36}$$

En dualisant ce problème on obtient :

$$\min_{\lambda^1,\lambda,\mu^2,\mu} \lambda^1 T + \mu^2 T^2 + \sum_{\substack{i,j\in V^2\\ i\neq j}} (\lambda_{ij} + 2\mu_{ij}) \tag{37}$$

$$\sum_{T\in[\![1,|V|]\!]} x_{ijt}(\widehat{t_i}+\widehat{t_j}) \leq \lambda^1 + \lambda_{ij} \quad \forall i,j \in V^2, i\neq j \tag{38}$$

$$\sum_{T\in[\![1,|V|]\!]} x_{ijt}\widehat{t_i}\widehat{t_j} \leq \mu^2 + \mu_{ij} \quad \forall i,j \in V^2, i\neq j \tag{39}$$

$$\lambda^1, \mu^2 \in \mathbb{R}_+ \tag{40}$$

$$\lambda_{ij}, \mu_{ij} \in \mathbb{R}_+ \quad \forall i,j \in V^2, i\neq j \tag{41}$$

Ainsi résoudre le problème robuste revient à résoudre le problème suivant :

$$\min_{x,b,u,\lambda^1,\lambda,\mu^2,\mu} \quad \lambda^1 T + \mu^2 T^2 + \sum_{\substack{i,j \in V^2 \\ i \neq j}} [(\lambda_{ij} + 2\mu_{ij}) + \sum_{T \in [\![1,|V|]\!]} t_{ij} x_{ijT}] \tag{42}$$

s.c. $\lambda^1, \lambda, \mu^2, \mu \in \big\{(\lambda^1, \lambda, \mu^2, \mu)$ vérifiant les contraintes (38) à (41)$\big\}$ (43)

$x, b, u \in \{(x, b, u)$ vérifiant les contraintes (4) à (13)$\}$ (44)

3 Résolution numérique

3.1 Principe

On décompose le problème robuste en un problème maître :

$$
(MP) \quad \begin{cases}
\min\limits_{z,x} & z & & \\
\text{s.c.} & z \geq \sum\limits_{ij \in A} t_{ij} x_{ij} & & \\
& \sum\limits_{j \in V \setminus \{i\}} x_{ji} = 1, & & i \in V \setminus \{1\} \\
& \sum\limits_{j \in V \setminus \{i\}} x_{ij} = 1, & & i \in V \setminus \{1\} \\
& \sum\limits_{j \in V \setminus \{1\}} x_{1j} = \sum\limits_{j \in V \setminus \{1\}} x_{j1} & & \\
& u_i \leq C - d_i, & & i \in V \setminus \{1\} \\
& u_j - u_i \geq d_i - C(1 - x_{ij}), & & ij \in (V \setminus \{1\})^2, i \neq j \\
& u_j \leq C(1 - x_{1j}), & & j \in V \setminus \{1\} \\
& x_{ij} \in \{0, 1\}, & & ij \in A \\
& u_i \geq 0, & & i \in V \setminus \{1\}
\end{cases}
$$

Et un sous-problème :

$$
(SP) \quad \begin{cases}
z(x^*) = \max\limits_{\delta^1, \delta^2} & \sum\limits_{ij \in A} \left[\delta^1_{ij} (\hat{t}_i + \hat{t}_j) + \delta^2_{ij} \hat{t}_i \hat{t}_j \right] x^*_{ij} & & \\
\text{s.c.} & \sum\limits_{ij \in A} \delta^1_{ij} \leq T & & \\
& \sum\limits_{ij \in A} \delta^2_{ij} \leq T^2 & & \\
& \delta^1_{ij} \in [0, 1], & & ij \in A \\
& \delta^2_{ij} \in [0, 2], & & ij \in A
\end{cases}
$$

On résout le problème en utilisant une méthode de Plans Coupants en ajoutant les scénarios, du sous-problème, infaisables au problème maître sous la forme de coupes jusqu'à ce qu'il soit réalisable (et donc optimale), un Branch-and-cut (basé sur le même principe que le plans coupants mais en intégrant les coupes au problème maître à partir de lazy constraints à chaque nœud de l'arbre de branchement du problème maître correspondant à une solution entière) et une méthode de dualisation.

3.2 Option : Heuristique

Nous avons codé une heuristique gloutonne prenant comme critère une certaine matrice de Reward, voici son pseudo-code :

```
Entrée : Données du problème (n, t, th, T, d, C)
Paramètres : hyperparamètres, max_runtime
Sortie : Matrice X représentant les tournées
```

```
Début
    Charger les données
    Initialiser la matrice des récompenses reward[i, j](hyperparamètres)
    Marquer l'entrepôt comme visité
    remaining_demand ← somme(d)

    Tant que remaining_demand > 0 Faire
        current_node ← 1, capacity_used ← 0
        Tant que capacité disponible Faire
            Sélectionner le prochain nœud next_node maximisant reward, sans dépasser la capacité
            Si aucun nœud valide, retour à l'entrepôt
            Sinon, marquer next_node comme visité, mettre à jour X et remaining_demand
            current_node ← next_node
        Fin Tant Que
    Fin Tant Que

    Calculer H_value via solve_subproblem
    Retourner (H_value, X)
Fin
```

Deux types de reward ont été testées :

— Reward 1 : $R^1_{i,j} = \frac{d_j}{\alpha t_{i,j} + \beta \hat{t}_j}$

— Reward 2 : $R^2_{i,j} = \frac{d_j}{\alpha t_{i,j} + \beta(\hat{t}_j + \hat{t}_i) + \gamma \hat{t}_j \times \hat{t}_i}$

Pour fixer nos hyperparamètres α, β ou γ, nous avons effectué une grid-search sur différentes valeurs, et nous avons choisi celles qui minimisent le gap moyen à la solution optimale sur les $\tilde{1}0$ instances dont nous avons l'optimum [3].

La reward 2 semble beaucoup plus logique et naturelle que la reward 1 ; on remarque qu'elle est bien meilleure que la reward 1 sur les données de calibrage, cependant sur toutes les instances, elle a des résultats nettement moins bons que ceux de la reward 1. Sur toutes les instances, la moyenne du gap absolu entre R^2 et R^1 vaut :

$$MOY(Val(R^1) - Val(R^2)) = -2218$$

On choisit donc la reward 1 comme heuristique.

L'étape suivante serait de prendre en compte des rewards de la forme $R_{i,j,l}$ afin d'avoir un algorithme glouton qui permet d'anticiper sur 2 pas de temps au lieu d'un ; cependant, faire ceci augmente drastiquement le temps de calcul en fonction du nombre de pas de temps d'anticipation.

3. On a $\alpha = 0.5 \quad \beta = 2$ pour R^1 et $\alpha = 0.1 \quad \beta = 1 \quad \gamma = 5$ pour R^2

3.3 Résultats

Nous avons tenté de résoudre toutes les instances de test en utilisant les méthodes présentées précédemment. Les résultats sont résumés dans le tableau suivant :

Instance	PR	Plans coupants		Branch-and-cut		Dualisation		Heuristique	
		Time	Gap	Time	Gap	Time	Gap	Time	Gap
n_5-euclidean_false	17,41 %	0,07	0 %	0,01	0 %	0,06	0 %	0,00	0 %
n_5-euclidean_true	10,43 %	0,35	0 %	0,01	0 %	0,07	0 %	0,00	0 %
n_6-euclidean_false	29,48 %	0,08	0 %	0,01	0 %	0,09	0 %	0,00	17 %
n_6-euclidean_true	2,15 %	0,32	0 %	0,02	0 %	0,10	0 %	0,00	8 %
n_7-euclidean_false	11,25 %	0,07	0 %	0,02	0 %	0,07	0 %	0,00	61 %
n_7-euclidean_true	4,99 %	0,46	0 %	0,06	0 %	0,11	0 %	0,00	8 %
n_8-euclidean_false	40,04 %	0,30	0 %	0,03	0 %	0,11	0 %	0,00	22 %
n_8-euclidean_true	5,65 %	0,66	0 %	0,06	0 %	0,16	0 %	0,00	31 %
n_9-euclidean_false	15,34 %	0,18	0 %	0,07	0 %	0,13	0 %	0,00	35 %
n_9-euclidean_true	18,48 %	4,08	0 %	0,18	0 %	0,13	0 %	0,00	26 %
n_10-euclidean_false	22,56 %	0,49	0 %	-2,20	0 %	0,13	0 %	0,03	85 %
n_10-euclidean_true	12,16 %	1,93	0 %	0,21	0 %	0,15	0 %	0,01	17 %
n_11-euclidean_false	34,17 %	0,89	0 %	0,53	0 %	0,21	0 %	0,00	38 %
n_11-euclidean_true	12,63 %	60,71	-2 %	3,39	0 %	0,54	0 %	0,00	12 %
n_12-euclidean_false	15,04 %	0,93	0 %	0,56	0 %	0,22	0 %	0,00	44 %
n_12-euclidean_true	7,71 %	61,52	8 %	60,00	12 %	7,71	12 %	0,00	24 %
n_13-euclidean_false	20,92 %	1,32	0 %	1,60	0 %	0,25	0 %	0,00	77 %
n_13-euclidean_true	16,90 %	62,29	-2 %	14,28	0 %	1,47	0 %	0,00	42 %

n_14-euclidean_false	34,28 %	63,01	-2 %	32,04	0 %	2,92	0 %	0,00	45 %
n_14-euclidean_true	0,00 %	60,04	24 %	60,00	45 %	60,05	45 %	0,00	79 %
n_15-euclidean_false	14,67 %	1,50	0 %	0,33	0 %	0,16	0 %	0,00	72 %
n_15-euclidean_true	1,09 %	75,74	10 %	60,00	14 %	23,04	14 %	0,01	54 %
n_16-euclidean_false	23,69 %	60,02	-40 %	60,00	18 %	26,06	14 %	0,00	31 %
n_16-euclidean_true	4,57 %	73,46	12 %	60,00	16 %	60,03	16 %	0,00	34 %
n_17-euclidean_false	142,51 %	63,64	-19 %	60,00	5 %	0,29	0 %	0,00	44 %
n_17-euclidean_true	0,00 %	60,02	49 %	339,06	59 %	60,03	59 %	0,00	116 %
n_18-euclidean_false	26,52 %	60,03	-19 %	60,00	117 %	60,03	103 %	0,00	203 %
n_18-euclidean_true	6,56 %	89,22	59 %	60,00	80 %	60,02	81 %	0,00	134 %
n_19-euclidean_false	49,82 %	98,11	-51 %	60,00	52 %	60,05	41 %	0,00	96 %
n_19-euclidean_true	0,00 %	60,03	43 %	60,00	49 %	60,02	49 %	0,46	113 %
n_20-euclidean_false	59,70 %	67,94	-46 %	60,00	15 %	60,04	13 %	0,00	88 %
n_20-euclidean_true	0,00 %	60,03	45 %	60,00	69 %	60,02	66 %	0,00	111 %
n_25-euclidean_false	18,90 %	60,03	-39 %	60,01	147 %	60,03	130 %	0,00	181 %
n_25-euclidean_true	1,29 %	60,68	56 %	60,01	77 %	60,03	77 %	0,01	114 %
n_30-euclidean_false	14,33 %	60,70	-54 %	60,00	159 %	60,03	141 %	0,01	269 %
n_30-euclidean_true	1,45 %	60,03	99 %	60,49	143 %	60,04	136 %	0,01	171 %
n_35-euclidean_false	111,04 %	60,03	-55 %	60,00	257 %	60,03	182 %	0,15	326 %
n_35-euclidean_true	-0,15 %	60,04	194 %	60,00	223 %	60,03	211 %	0,01	280 %

n_40-euclidean_false	104,42 %	60,05	57 %	60,05	352 %	60,03	367 %	0,02	497 %
n_40-euclidean_true	-3,18 %	60,16	61 %	60,00	109 %	60,04	94 %	0,23	118 %
n_45-euclidean_false	25,90 %	60,05	-52 %	60,01	155 %	60,05	111 %	0,02	363 %
n_45-euclidean_true	-1,36 %	60,05	138 %	60,01	170 %	60,04	163 %	0,02	221 %
n_50-euclidean_false	9,50 %	60,10	-57 %	60,01	165 %	60,05	79 %	0,03	366 %
n_50-euclidean_true	-1,95 %	60,09	82 %	60,01	123 %	60,04	98 %	0,03	166 %
n_55-euclidean_false	43,10 %	60,09	-60 %	60,01	259 %	60,06	161 %	0,09	419 %
n_55-euclidean_true	4,94 %	60,09	81 %	60,01	122 %	60,06	109 %	0,03	175 %
n_60-euclidean_false	31,13 %	60,09	-32 %	60,01	293 %	60,06	229 %	0,03	438 %
n_60-euclidean_true	6,48 %	60,10	192 %	60,01	212 %	60,06	192 %	0,03	232 %
n_65-euclidean_false	27,03 %	60,11	-31 %	60,01	307 %	60,08	184 %	0,08	385 %
n_65-euclidean_true	-4,55 %	60,12	178 %	60,01	217 %	60,06	198 %	0,08	248 %
n_70-euclidean_false	5,89 %	60,13	108 %	60,02	504 %	60,08	458 %	0,04	582 %
n_70-euclidean_true	-3,10 %	62,86	22 %	60,02	97 %	60,10	81 %	0,03	120 %
n_75-euclidean_false	34,12 %	60,14	-57 %	60,02	352 %	60,11	216 %	0,04	531 %
n_75-euclidean_true	-1,19 %	60,15	139 %	60,02	166 %	60,10	143 %	0,05	201 %
n_80-euclidean_false	25,11 %	60,20	-25 %	60,03	181 %	60,10	96 %	0,06	374 %
n_80-euclidean_true	-11,82 %	60,19	75 %	60,02	155 %	60,10	125 %	0,09	171 %
n_85-euclidean_false	11,97 %	60,22	210 %	60,02	813 %	60,11	800 %	0,06	881 %
n_85-euclidean_true	-3,87 %	60,20	27 %	60,04	100 %	60,11	98 %	0,06	120 %

n_90-euclidean_false	-20,70 %	60,16	-17 %	60,03	570 %	60,32	539 %	0,07	712 %
n_90-euclidean_true	3,04 %	60,15	293 %	60,03	376 %	60,11	350 %	0,06	405 %
n_95-euclidean_false	20,22 %	60,25	-27 %	60,03	596 %	60,13	550 %	0,09	660 %
n_95-euclidean_true	0,28 %	60,14	324 %	60,03	345 %	60,12	327 %	0,09	392 %
n_100-euclidean_false	21,72 %	60,14	49 %	60,03	537 %	60,14	494 %	0,07	669 %
n_100-euclidean_true	-17,89 %	60,19	98 %	60,04	180 %	60,13	148 %	0,07	167 %

TABLE 1 – Tableau des résultats pour toutes les instances et les méthodes suivantes : Plans coupants, Branch-and-cut, Dualisation, Heuristique.

La lecture du tableau étant fastidieuse, nous représentons également les résultats sous la forme du diagramme de performances suivant :

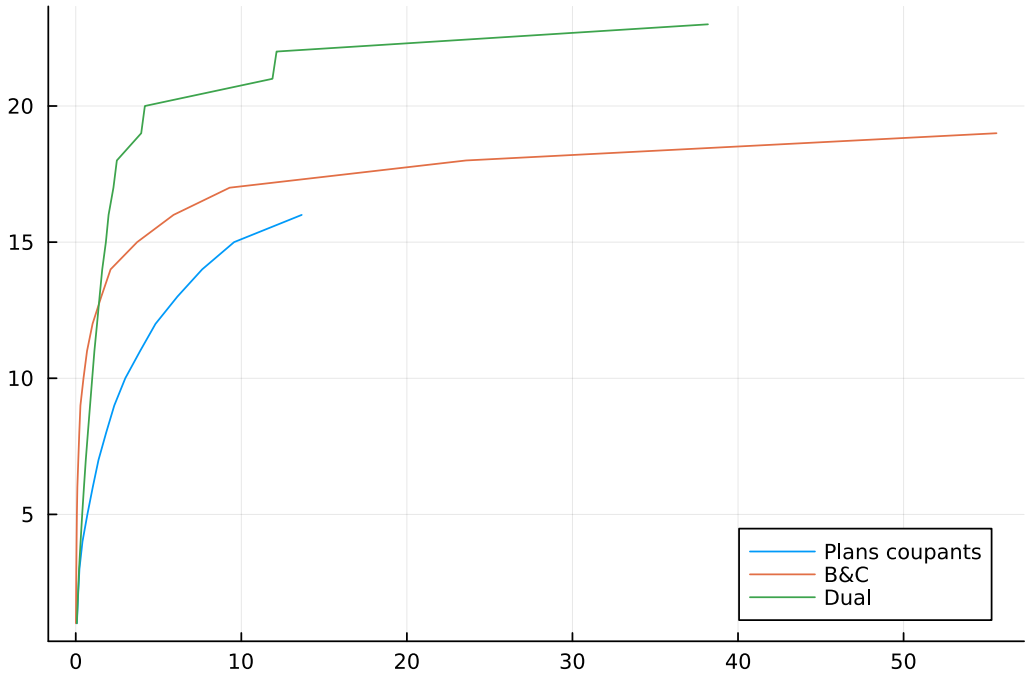

FIGURE 1 – Diagramme de performances.

Afin de mieux comparer l'efficacité des différentes méthodes, nous ajoutons d'autres graphiques.

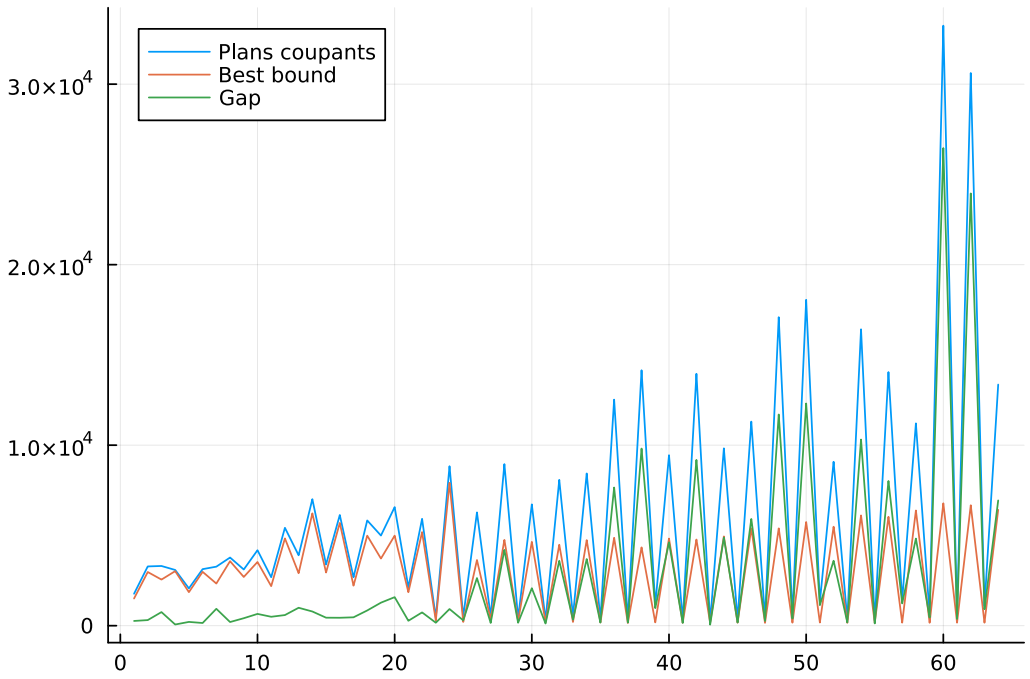

FIGURE 2 – Diagramme de la meilleure solution (non réalisable) trouvée en 60s par la méthode des Plans coupants.

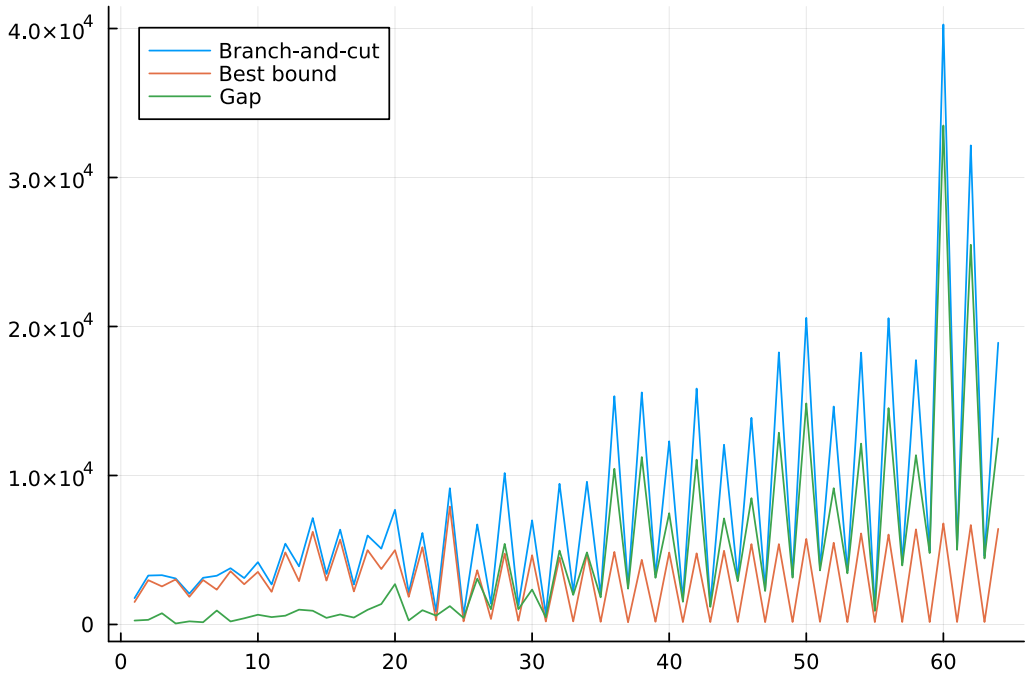

FIGURE 3 – Diagramme de la meilleure solution réalisable trouvée en 60s par la méthode Branch-and-cut.

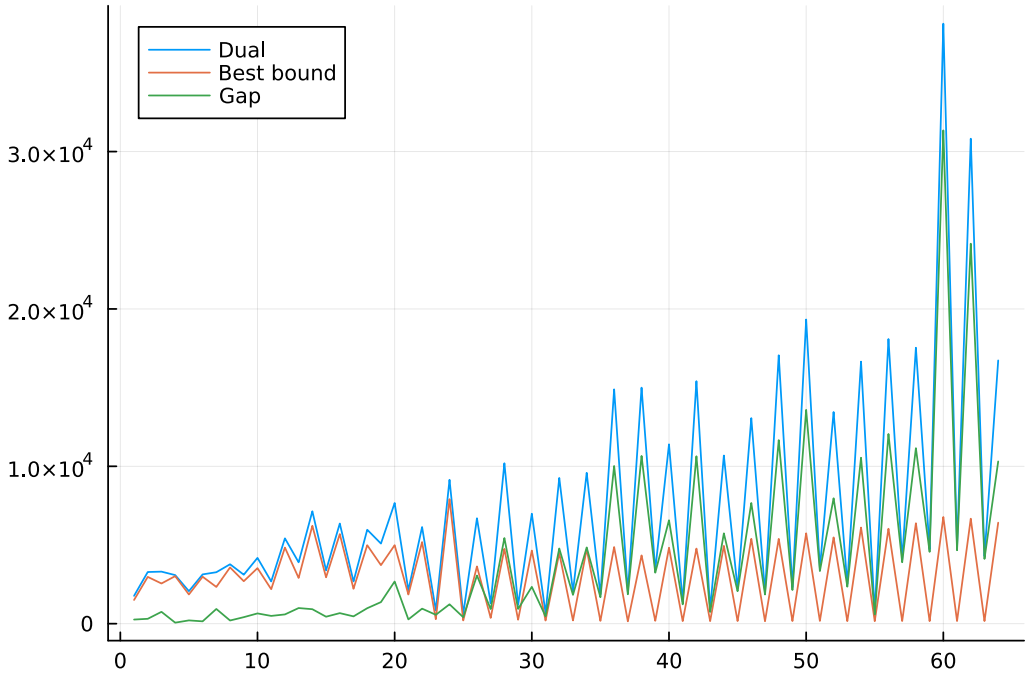

FIGURE 4 – Diagramme de la meilleure solution réalisable trouvée en 60s par la méthode de Dualisation.

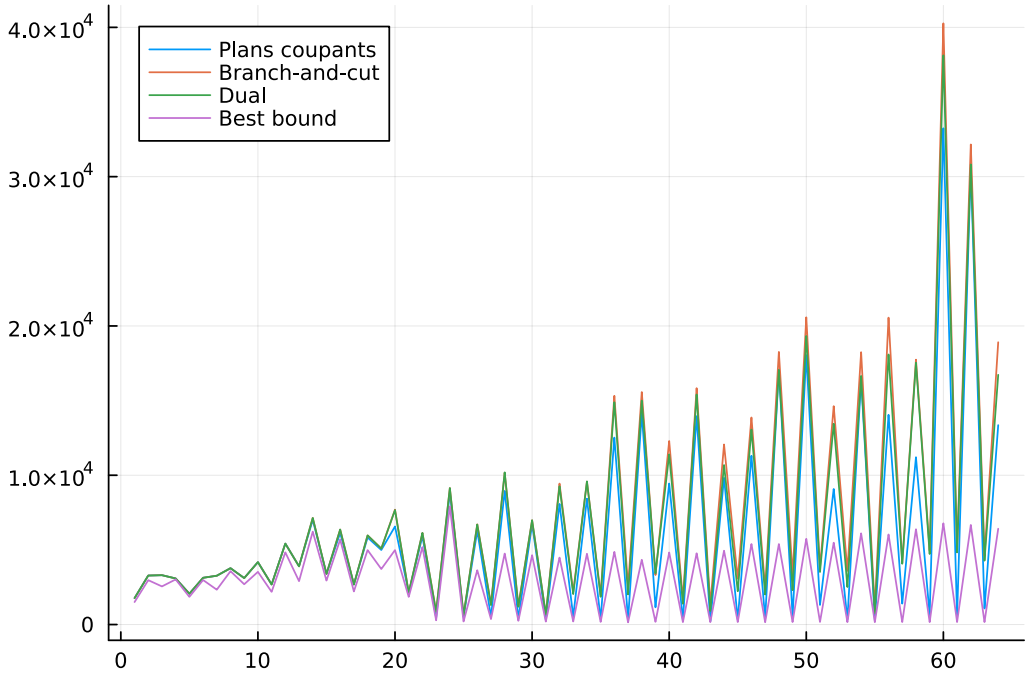

FIGURE 5 – Diagramme de comparaison des meilleures bornes obtenues en moins de 60s pour chacune des méthodes.

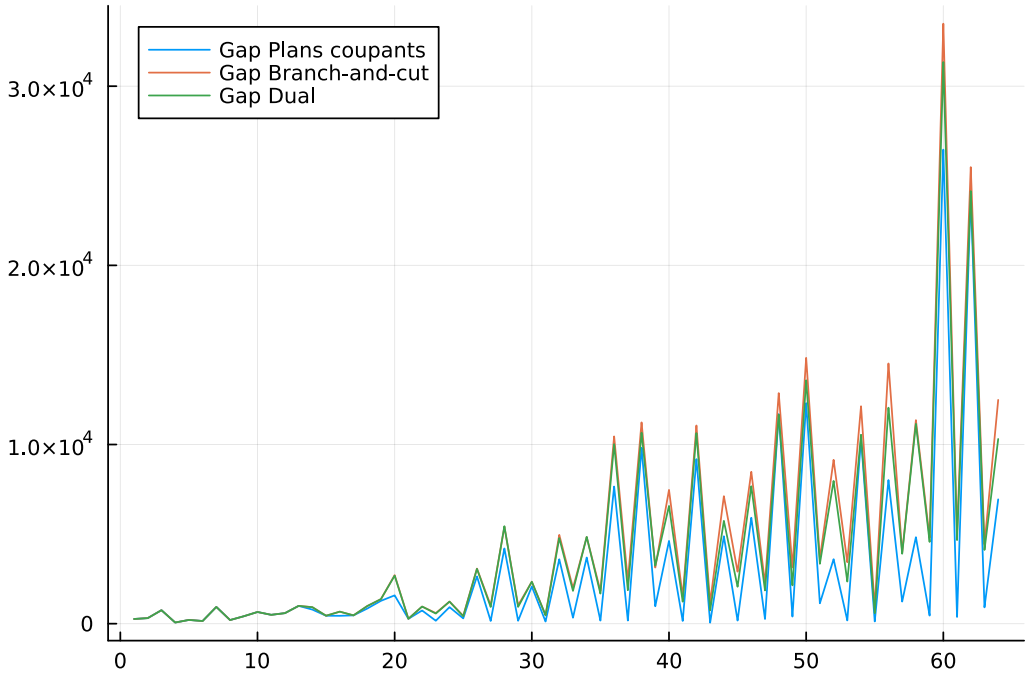

FIGURE 6 – Diagramme de comparaison des gaps pour chacune des méthodes.

On peut donc voir que les différentes méthodes implémentées fournissent des résultats différents. On effet, bien que toutes les méthodes fournissent des solutions optimales équivalentes pour des instances de tailles modérées (moins de 20 nœuds), la qualité des solutions obtenues ne sont pas du tout équivalentes pour de plus gros problèmes. La méthode de dualisation va permettre de résoudre plus d'instances que les deux autres (plans coupants et branch-and-cut). De même, le branch-and-cut va se révéler plus performant que la méthode des plans coupants en résolvant plus d'instance en moins d'une minute par instance. De plus, cette méthode devient rapidement limitée puisqu'elle ne fournit pas de borne supérieure pour le problème puisque les solutions non optimales ne sont pas réalisables contrairement aux relaxations faites en chaque nœud de l'arbre du problème maître du branch-and-cut.

3.4 Conclusion

Les 3 méthodes classiques utilisées ici nous donnent des résultats concluants sur une partie des instances avec un léger avantage pour la méthode duale. Le branch-and-cut est toutefois intéressant puisqu'il fournit des bornes pour le problème même lorsque l'optimum n'est pas atteint.

Certaines pistes d'améliorations méritent d'être citées :

— Comme énoncé dans la partie heuristique il est possible d'étendre l'algorithme en anticipant sur n étapes au lieu de 1, ce qui certe va améliorer les résultats, cependant cela va aussi augmenter drastiquement la complexité combinatoire du calcul de la reward.

— Il aurait été intéressant de travailler sur une formulation non compacte du problème [4]. Cependant cette approche utilisant des coupes, on peut prédire des résultats assez proches de la méthode branch-and-cut, notamment des problèmes trop longs à résoudre pour des instances trop grandes.

3.5 Annexe

Instances résolues à l'optimum :

PC	BC	Dual
n_10-euclidean_false	n_7-euclidean_true	n_9-euclidean_false
n_10-euclidean_true	n_12-euclidean_false	n_12-euclidean_false
n_11-euclidean_false	n_9-euclidean_true	n_10-euclidean_true
n_12-euclidean_false	n_10-euclidean_true	n_8-euclidean_true
n_13-euclidean_false	n_15-euclidean_false	n_15-euclidean_false
n_15-euclidean_false	n_11-euclidean_false	n_11-euclidean_false
n_5-euclidean_false	n_6-euclidean_false	n_7-euclidean_false
n_5-euclidean_true	n_8-euclidean_false	n_8-euclidean_false
n_6-euclidean_false	n_5-euclidean_false	n_5-euclidean_true
n_6-euclidean_true	n_7-euclidean_false	n_7-euclidean_true
n_7-euclidean_false	n_10-euclidean_false	n_5-euclidean_false
n_7-euclidean_true	n_8-euclidean_true	n_10-euclidean_false
n_8-euclidean_false	n_6-euclidean_true	n_6-euclidean_true
n_8-euclidean_true	n_9-euclidean_false	n_9-euclidean_true
n_9-euclidean_false	n_5-euclidean_true	n_6-euclidean_false
n_9-euclidean_true	n_13-euclidean_false	n_13-euclidean_false
	n_11-euclidean_true	n_17-euclidean_false
	n_13-euclidean_true	n_11-euclidean_true
	n_14-euclidean_false	n_13-euclidean_true
		n_14-euclidean_false
		n_12-euclidean_true
		n_15-euclidean_true
		n_16-euclidean_false

TABLE 2 – Instances résolues jusqu'à l'optimum en fonction des différentes méthodes.

Solutions disponibles en suivant le lien :
https ://github.com/micagent007/PROJ-MPRO/blob/main/Last

4. ζ_C la variable binaire d'utilisation du cycle C, puis une approche génération de colonne sur les cycles réalisables