M2 ANDROIDE

Rapport du projet MAOA

Auteurs:
Emilie BIEGAS: 3700036
Alix ZHENG: 3700357

10/12/2021

Table des matières

1 Travail réalisé **1**

2 Fonctions utilitaires **2**

3 Résolution approchée **5**
3.1 Le problème de Lot-Sizing (LSP) 5
3.2 Le problème de tournée de véhicules (VRP) 5
3.2.1 Métaheuristiques pour le VRP 5
3.2.2 PLNE pour le VRP 7
3.3 Heuristique pour le PDI 7
3.4 Temps d'exécution et qualité de la solution 7

4 Résolution exacte **12**
4.1 PLNE du problème de production et distribution intégré 12
4.2 Test sans amélioration 12
4.2.1 Taille des Instance 12
4.3 Test avec amélioration 14
4.3.1 Résutats 16

5 Analyse **17**

6 Discussions **17**

References **20**

1 Travail réalisé

■ Ce que nous avons réussi à faire

— Fonctions utilitaires :

1. Lecture des instances
2. Représentation claire de la solution
3. Fonctions permettant d'obtenir les circuits au format désiré à partir des valeurs des variables de décisions récupérées de la résolution de PLNE : pour le PLNE du TSP et du VRP.

— Méthode heuristique :

1. Le PLNE du LSP (avec ou sans prise en compte des coûts de visite)
2. L'heuristique du "Bin packing" pour le VRP
3. L'heuristique de Clark-Wright (avec prise en compte de la dépendance des distances et des valeurs des demandes) pour le VRP
4. L'heuristique sectorielle pour le VRP
5. Le PLNE et l'approche heuristique du TSP
6. Plusieurs versions réfléchies de la méthode itérative pour le VRP (voir paragraphe correspondant)
7. Le PLNE basé sur les inégalités MTZ du VRP
8. L'heuristique itérative en deux phases pour la résolution du PDI
9. Tests et comparaison des temps d'exécution et de la qualité de la solution en fonction de la taille de l'instance et des heuristiques utilisées

— Méthode exacte :

1. Implémenter le PLNE 2 décrit dans l'article [1] [20 : 32]
2. Tester la robustesse en terme de taille d'instance pouvant être résolu par ce PLNE
3. Remplacer la contrainte (29) par la contrainte (33) en réalisant un Branch and Cut.
4. Implémenter le PLNE 1 [1 : 16]
5. Remplacer la contrainte (11) et (12) par la contrainte (17) et (19) en réalisant un Branch and Cut.
6. Tester la robustesse en terme de taille d'instance pouvant être résolu par ce PLNE.

— Comparer le temps d'execution en fonction de la taille des instances

— Comparer la qualité des solutions renvoyées

— Afficher le graphe des tournées

■ Ce que nous avons fait, mais que nous n'avons pas eu le temps de finir

— Tester la robustesse avec le Branch and Cut du PLNE 2

2 Fonctions utilitaires

Nous avons, tout d'abord implémenté des fonctions utilitaires permettant de lire les fichiers d'instance, et permettant de représenter la solution du PDI obtenue de façon claire et concise grâce à un ensemble de graphe. En effet, une solution est de la forme [p, y, I, q, tournees] où p sont les variables de production : p[t] représente la quantité produite à la période t ; y sont les variables de lancement : y[t] est une variable binaire qui vaut 1 si une production est lancée à la période t, et 0 sinon (ces variables se déduisent donc de p et n'ont pas besoin d'être explicitées) ; I sont des variables de stockage : I[i][t] représente la quantité en stock à la fin de la période t pour le revendeur i (ces variables se déduisent du stock de départ, de la demande en i, de p et de q) ; q sont les variables d'approvisionnement : q[i][t] représente la quantité produite pour le revendeur i à la période t ; et tournees est l'ensemble des tournées (c'est-à-dire l'ensemble de tournées pour tous les pas de temps).

Nous avons donc décidé de représenter une solution par le vecteur p et plusieurs graphes orientés (un par pas de temps où il y a des distributions à effectuer) décrivant l'ensemble des tournées (avec une couleur différente sur les nœuds et les arcs pour chaque tournée) et dont les nœuds sont labellisé par un couple (numéro du nœud dans le problème, q[i,t]). Notons que la position des nœuds est définie par les données de l'instance.

Par exemple, les graphiques retournés pour l'instance "B_100_instance30.prp" et pour les pas de temps 1 et 2 avec la méthode heuristique du PDI en utilisant l'heuristique de Clark-Wright pour résoudre le VRP sont donnés dans les deux pages suivantes.
Lors du premier pas de temps, il y a donc 5 tournées à effectuer (avec donc 5 camions) : la tournée bleue, orange, violette, verte et jaune. Lors du deuxième pas de temps, il y a par contre 4 tournées à effectuer (avec donc 4 camions) : la tournée verte, bleue, violette et orange. Le sommet bleu foncé est le centre de dépôt (en haut à gauche).
Pour chaque pas de temps, sur chaque sommet est indiqué la quantité à livrer. Pour savoir quelle quantité chaque camion doit transporter, on n'a qu'à sommer ces valeurs sur l'ensemble des sommets de son parcours (chaque camion est associé à une tournée et donc à une couleur). Chaque camion n'a plus qu'à suivre l'itinéraire indiqué par les arcs de sa couleur (en partant du centre de dépôt bien sûr). Il est donc clair que si l'on fournit, en plus du vecteur p ce type de graphe pour chaque pas de temps où il y a des distributions à effectuer (il se peut qu'aucune distribution ne soit effectuée pour un pas de temps donné), la solution est compréhensible.
Notons que sur des pas de temps différents comme ici, il se peut que ce ne soit pas les mêmes clients considérés (on le voit bien en haut à droite de chaque figure où dans le premier temps il y avait trois sommets proches à délivrer mais pas dans le second temps). En effet, rappelons que les sommets sont représentés dans l'espace à l'aide des coordonnées fournies dans les instances et que leur position est donc représentative de la réalité. En ce qui concerne la qualité des tournées obtenues, elle semble plutôt bonnes étant donné la proximité des points de même couleur (et donc appartenant à la même tournée). Le schéma de tournées semble cohérent. Nous formaliseront ce résultat par la suite.

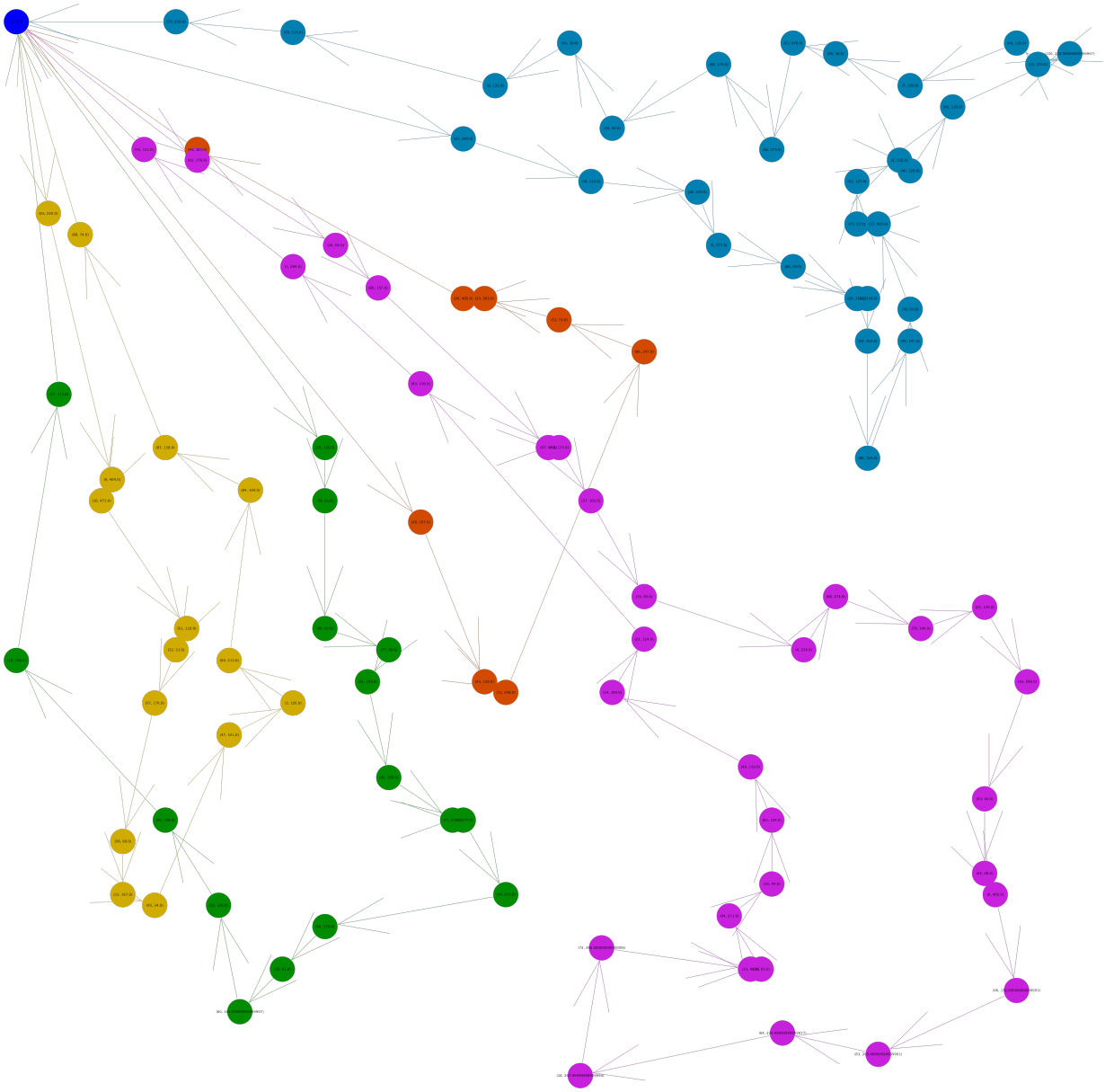

FIGURE 1 – Graphique de distribution aux revendeurs au temps 1

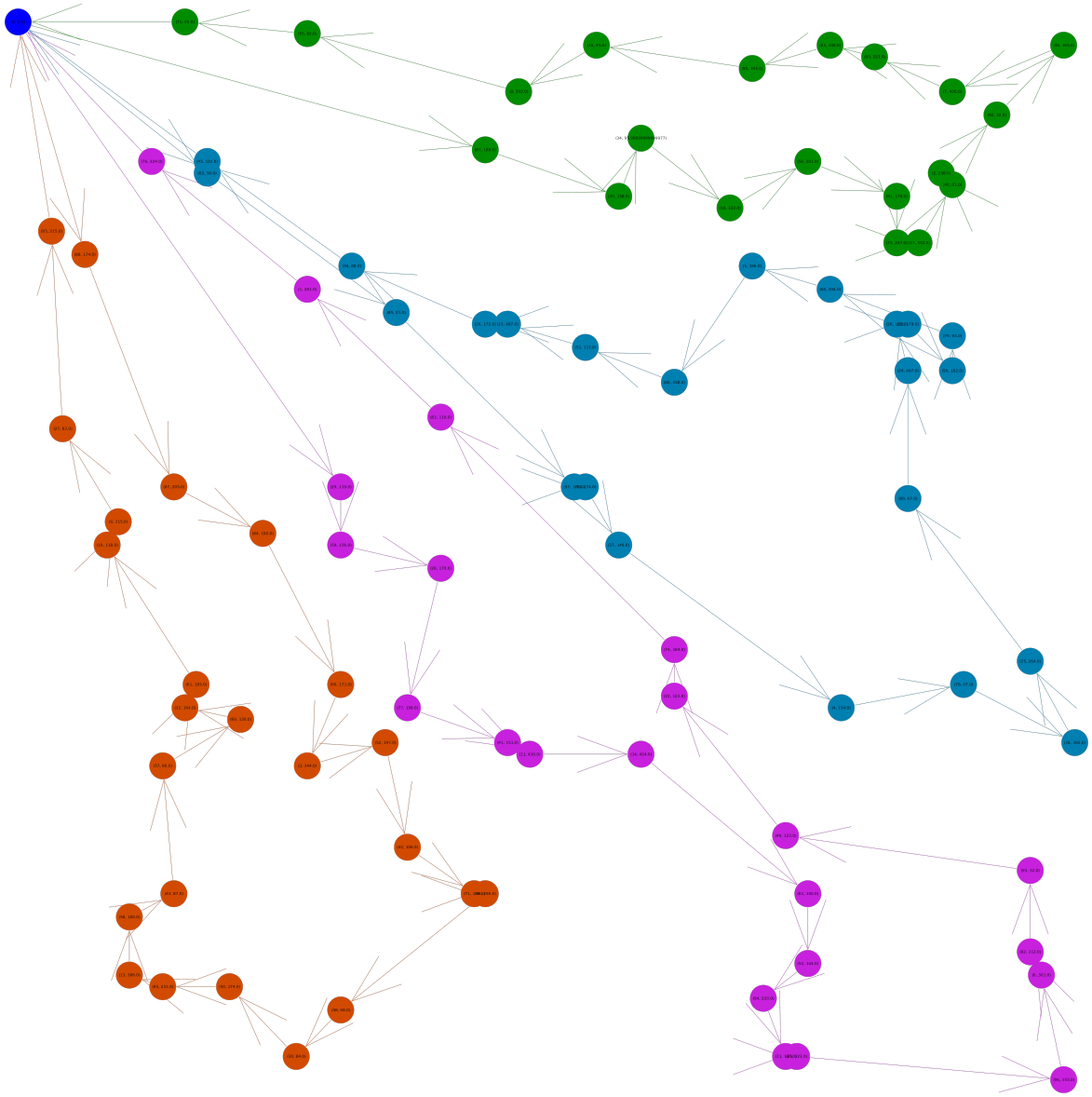

FIGURE 2 – Graphique de distribution aux revendeurs au temps 2

3 Résolution approchée

3.1 Le problème de Lot-Sizing (LSP)

Le problème LSP consiste à minimiser les coûts de production, de lancement et de stockage.

Nous avons implémenté la formulation classique pour ce problème : le PLNE. En ajoutant la contrainte suivante, pour tout t allant de 1 à l :
@constraint(m, sum(q[i,t] for i in 1 :n) <= Q*k).
Cette contrainte fait en sorte que le VRP soit réalisable (la quantité à distribuer ne doit pas excéder la capacité maximale totale pour chaque pas de temps).
Nous avons également ajouté un paramètre SC à cette fonction de résolution du LSP afin d'indiquer si on doit ou non prendre en compte les coûts de visite (utile pour l'heuristique du PDI).

3.2 Le problème de tournée de véhicules (VRP)

Le problème VRP consiste (pour un pas de temps t donné) à déterminer un ensemble d'au plus m tournées $(T_1, ..., T_k)$, $k \leq m$ (partant du dépôt et passant livrer la quantité demandée par chacun des revendeurs).

3.2.1 Métaheuristiques pour le VRP

■ Heuristiques gloutonnes pour le VRP :

Nous avons implémenté les trois heuristiques gloutonnes à savoir l'heuristique gloutonne "Bin packing", l'heuristique gloutonne de Clark-Wright (avec les α, β, γ pris entre 0 et 2 afin de prendre un peu en compte le fait que la fusion dépend plus ou moins des distances et plus ou moins des valeurs des demandes), et l'heuristique sectorielle.

Elles ne produisent qu'une partition des clients, à la fin, il est donc nécessaire, pour chaque élément de la partion de déterminer un circuit unique passant par tous les sommets de la partition : c'est-à-dire résoudre le problème du voyageur de commerce (TSP) limité à ces sommets : nous avons testé deux méthodes pour le faire, dans la méthode itérative, une avec une heuristique gloutonne au plus proche sommet en démarrant du dépôt et une grâce à la PLNE.

■ Méthode itérative :

Nous avons implémenté une méthode itérative dont voici les étapes :

1. Dans un premier temps, on utilise une des heuristiques gloutonne précédemment présentées afin d'obtenir un ensemble de boîtes.
2. (optionnel) On applique le TSP à chaque boîte.
3. On fait en sorte qu'il y ait exactement k (le nombre de véhicules disponibles) tournées, c'est à dire k boîtes.
4. On itère un nombre donné de fois pour permettre à des clients de changer de boîte si ça améliore la solution et que ça respecte la capacité des camions.
5. (optionnel) On applique le TSP à chaque boîte.

6. On itère un nombre donné de fois, sur chaque boîte, un voisinage LSP 2-opt afin d'améliorer la solution.
7. On finit par supprimer les boîtes vides (ne contenant que le dépôt).

Nous avons mis en place différentes méthodes pour chacune de ces étapes.

À l'étape 1., nous pouvons utiliser chacune des trois heuristiques précédemment citées.

Pour l'étape 3., il y a deux cas. Soit la taille de la boîte précédemment obtenue (c'est-à-dire le nombre de tournées pour ce pas de temps) est inférieure au nombre de véhicules disponibles et dans ce cas il faut ajouter des tournées jusqu'à en avoir le même nombre que le nombre de camions. Cela peut s'effectuer de deux manières, soit en ajoutant des tournées vides (ne contenant que le dépôt), soit en séparant les tournées jusqu'à utiliser tous les véhicules. On choisit ainsi la boîte comportant le plus d'éléments et on la sépare en deux boîtes distinctes (notons que dans ce cas, les capacités des deux nouvelles boîtes ne dépassent forcément pas celle du camion puisque ce n'était pas le cas pour la boîte entière réunissant ces deux nouvelles boîtes).
Soit la taille de la boîte précédemment obtenue (c'est-à-dire le nombre de tournées pour ce pas de temps) est supérieure au nombre de véhicules disponibles et dans ce cas il faut retirer des tournées jusqu'à en avoir le même nombre que le nombre de camions. Cela peut s'effectuer en réunissant les tournées : on choisit les deux boîtes comportant le moins d'éléments (ou dont la somme des q[i,t] est minimale) et on les réunit en veillant à ce que la capacité du camion soit respectée, et on réitère l'opération jusqu'à n'utiliser que le nombre de véhicules disponibles. S'il n'existe pas de tel regroupement respectant les capacités des camions, nous résolvons le VRP exactement par PLNE. D'autres possibilités sont envisageables lorsque le nombre de tournées de la solution initialement obtenue est supérieur au nombre de véhicules disponibles. Il est, en effet, possible d'utiliser dans un premier temps une fonction objective artificielle pour forcer à réduire le nombre de tournées. Une autre possibilité serait de prendre les boîtes de demande la plus faible (en nombre nécessaire pour atteindre k boîtes au total) puis, dans l'ordre croissant des demandes, on répartit chacun des clients dans les autres boîtes jusqu'à ce que la capacité du camion soit atteinte (ou juste avant qu'elle ne le soit). Cela revient donc à un problème de sac à dos mais, de la même façon qu'avec la méthode mise en place précédemment, il n'est pas forcé que l'on trouve une solution réalisable respectant les capacités des camions.

Attardons nous désormais sur les étapes 2. et 5. et explicitons les cas où l'on effectue ces étapes. Notons d'abord qu'aux étapes 2. et 5., nous pouvons appliquer le TSP soit par la PLNE soit grâce à l'heuristique qui, commençant au dépôt, choisit le sommet le moins loin (parmi les sommets restants) jusqu'à ce qu'il ne reste plus aucun sommet à visiter.
Une première version de l'algorithme consistait à effectuer l'étape 5. mais pas la 2.. Dans cette première version, l'étape 4. s'effectue heuristiquement. En effet, on prend un coût heuristique pour évaluer l'amélioration de la solution dû à un changement de boîte pour un client car on n'a pas encore appliqué un voisinage de TSP dans chacune des boîtes et donc on ne peut pas utiliser le vrai coût qui serait faussé par l'ordre de visite des sommets dans chaque boîte. On cherche donc à définir les boîtes

de coût heuristique les plus faible. L'heuristique sur le coût d'une boîte est définie comme la somme, pour chaque sommet de ses deux arêtes de coût minimum (dont l'extrémité fait partie des sommets envisagés évidemment) le tout divisé par 2.
Une seconde version de l'algorithme consistait à effectuer l'étape 2. mais pas la 5.. Notons que nous n'avons pas choisi d'effectuer les deux étapes puisque, si l'on effectue l'étape 2., il est préférable de ne pas effectuer la 5.. En effet, en plus de l'ajout de temps de calcul (qui nous dissuade d'effectuer les deux étapes si l'on résout le TSP par PLNE), l'étape 5. est inutile et pourrait même baisser la valeur de la solution si on effectue le TSP heuristiquement. Effectivement, on commence avec des boîtes heuristiquement bonnes selon TSP (grâce à l'étape 2.), puis à l'étape 3., si la taille de la boîte est inférieure au nombre de véhicules disponibles, l'heuristique TSP est toujours vérifiée pour chacun des deux sous-ensemble (si on effectuait l'heuristique TSP sur ces sous-ensemble, la solution ne changerait pas) par définition de l'heuristique, nous n'avons donc pas besoin d'appliquer le TSP de nouveau. D'autre part, si la taille de la boîte est supérieure au nombre de véhicules disponibles, notre algorithme effectue déjà un TSP sur la nouvelle boîte réunissant les deux boîtes précédentes. Enfin, lors de l'étape 4, on insère les clients à l'endroit où le coût est minimisé (voir plus bas), d'où l'inefficacité d'appliquer un TSP heuristique sur les boîtes obtenues.
Maintenant cela explicité, nous effectuons donc l'étape 2. mais pas la 5., on a donc à l'issue de l'étape 2. une solution heuristiquement bonne dans chacune des boîtes, ce qui est mieux qu'avant. L'étape 4. ne s'effectue donc plus heuristiquement comme avec la première version de l'algorithme, mais s'effectue comme suit : pour un client donné, on regarde pour chaque boîte le coût minimum que l'insertion de ce client engendrerait et si bouger ce client dans cette nouvelle boîte (à l'endroit le moins cher) est moins coûteux alors on le déplace.

3.2.2 PLNE pour le VRP

Il existe plusieurs formulations PLNE classiques pour ce problème, nous avons implémenté dans ce projet une formulation compacte basée sur les inégalités de Miller-Tucker-Zemlin (MTZ) ainsi qu'une fonction permettant d'obtenir les circuits obtenus au format désiré à partir des valeurs des variables de décisions récupérées de la résolution de PLNE.

3.3 Heuristique pour le PDI

Nous avons implémenté l'heuristique itérative en deux phases proposée avec possibilité d'obtenir une représentation claire de la solution comme explicité dans le paragraphe sur les fonctions utilitaires. L'appel à cette fonction permet également de préciser comment va être résolu le VRP (soit grâce à la méthode itérative en précisant l'heuristique à utiliser, soit par résolution exacte avec la PLNE), ainsi que de préciser si l'on veut résoudre le TSP heuristiquement ou exactement.

3.4 Temps d'exécution et qualité de la solution

Pour le temps d'exécution en fonction de la méthode utilisée :

Instance	Clark-Wright	Bin Packing	Sectorielle	PLNE
A3	0.10299992	1.03099989	0.10599994	0.18800020
A4	0.10299992	0.09400010	0.08300018	0.26699995
A5	0.06200003	0.07299995	0.14000010	0.36100006
A6	0.06399989	0.04699993	0.06199979	0.30099987
A62	0.13899993	0.07899999	0.07499980	0.36699986
A10	0.08999991	0.09400010	0.12600016	0.52200007
A-014-ABS11-15-1	1.37100005	1.375	1.32599997	—
A-014-ABS21-15-1	1.32899999	1.29799985	1.35900020	—
A-014-ABS31-15-1	1.11399984	1.11199998	1.09899997	—
A-014-ABS41-15-1	1.75900006	1.67700004	1.79900002	—
A-014-ABS51-15-1	1.68400001	1.66100001	1.77699995	—
A-014-ABS61-15-1	1.93599987	1.70300006	1.76600003	—
A-014-ABS71-15-1	1.72499990	1.73500013	1.73600006	—
A-014-ABS71-15-4	1.48799991	1.52200007	1.46499991	—
A-014-ABS80-15-4	1.33999991	1.33699989	1.37600016	—
A-014-ABS81-15-1	1.44000005	1.43499994	1.44899988	—
A-014-ABS91-15-1	1.55699992	1.56800007	1.54299998	—
A-014-ABS94-15-5	1.35199999	1.32500004	1.35899996	—
A-014-ABS96-15-5	1.19000005	3.57000017	1.18700003	—
A-050-ABS1-50-2	2.94499993	2.94400000	2.76800012	—
A-050-ABS13-50-3	2.84399986	2.96900010	2.70300006	—
A-050-ABS19-50-5	2.07200002	2.11899995	2.01399993	—
A-050-ABS77-50-3	2.32599997	2.26699995	2.28999996	—
A-050-ABS78-50-2	2.01699995	2.06699991	2.04099988	—
A-100-ABS1-100-1	5.20300006	4.97099995	5.00200009	—
A-100-ABS12-100-4	3.95399999	4.0	3.84700012	—
A-100-ABS20-100-3	5.02199983	4.81299996	4.66400003	—
A-100-ABS28-100-3	5.58200001	5.30899977	5.09299993	—
A-100-ABS92-100-1	4.67300009	4.63600015	4.45399999	—
B-050-instance14	9.87000012	9.54699993	9.12599992	—
B-050-instance2	8.21100020	7.71399998	7.65899991	—
B-050-instance22	11.2829999	9.19099998	10.2850000	—
B-050-instance30	14.6310000	14.2250001	13.8680000	—
B-050-instance7	8.27999997	8.97399997	8.01699995	—
B-100-instance1	28.3919999	26.0219998	25.8220000	—
B-100-instance16	23.5460000	19.1170001	19.0489997	—
B-100-instance20	31.9979999	28.1889998	28.5239999	—
B-100-instance30	23.7760000	20.0370001	19.1940000	—
B-100-instance6	31.5519998	25.5980000	29.4110000	—

TABLE 1 – Temps d'exécution en fonction de la méthode utilisée

Pour le coût de la solution obtenue en fonction de la méthode utilisée :

Instance	Clark-Wright	Bin Packing	Sectorielle	PLNE
A3	10910.0	10910.0	10910.0	10910.0
A4	27295.0	27295.0	27295.0	27295.0
A5	27542.0	27542.0	27542.0	27542.0
A6	16899.0	16899.0	16899.0	16899.0
A62	28051.0	28051.0	28051.0	28051.0
A10	48316.0	48316.0	48316.0	48316.0
A-014-ABS11-15-1	67761.0	67761.0	67761.0	—
A-014-ABS21-15-1	36300.0	36177.0	36177.0	—
A-014-ABS31-15-1	230516.0	230516.0	230516.0	—
A-014-ABS41-15-1	613451.0	613451.0	613451.0	—
A-014-ABS51-15-1	85714.0	85100.0	85100.0	—
A-014-ABS61-15-1	133523.0	133523.0	133523.0	—
A-014-ABS71-15-1	162284.0	162284.0	162284.0	—
A-014-ABS71-15-4	157502.0	157502.0	157502.0	—
A-014-ABS80-15-4	29310.0	29310.0	29310.0	—
A-014-ABS81-15-1	32850.0	32522.0	32556.0	—
A-014-ABS91-15-1	231295.0	231295.0	231295.0	—
A-014-ABS94-15-5	512871.0	512871.0	512871.0	—
A-014-ABS96-15-5	514685.0	514685.0	514685.0	—
A-050-ABS1-50-2	106879.0	107352.0	107653.0	—
A-050-ABS13-50-3	113210.0	113363.0	112718.0	—
A-050-ABS19-50-5	95456.0	95979.0	95438.0	—
A-050-ABS77-50-3	183637.0	183576.0	183723.0	—
A-050-ABS78-50-2	186671.0	187078.0	186718.0	—
A-100-ABS1-100-1	202592.0	197390.0	200197.0	—
A-100-ABS12-100-4	351779.0	350170.0	351088.0	—
A-100-ABS20-100-3	190075.0	191857.0	189648.0	—
A-100-ABS28-100-3	3.401454e6	3.401894e6	3.402949e6	—
A-100-ABS92-100-1	1.322883e6	1.322839e6	1.322378e6	—
B-050-instance14	480253.356	459843.680	498521.731	—
B-050-instance2	512132.237	493627.956	519990.803	—
B-050-instance22	445205.506	450692.078	450792.248	—
B-050-instance30	452681.726	461948.743	464504.249	—
B-050-instance7	472724.369	492016.986	466796.058	—
B-100-instance1	696837.664	707005.318	696192.620	—
B-100-instance16	724991.063	723673.799	735568.147	—
B-100-instance20	721265.742	728361.080	722411.531	—
B-100-instance30	685098.257	700107.131	700931.606	—
B-100-instance6	686456.238	712225.968	694231.576	—

TABLE 2 – Coût de la solution obtenue en fonction de la méthode utilisée

Nous avons donc représenté les données en fonction de la taille de l'instance, du type d'instance et de l'heuristique utilisée sur les graphiques des pages suivantes.

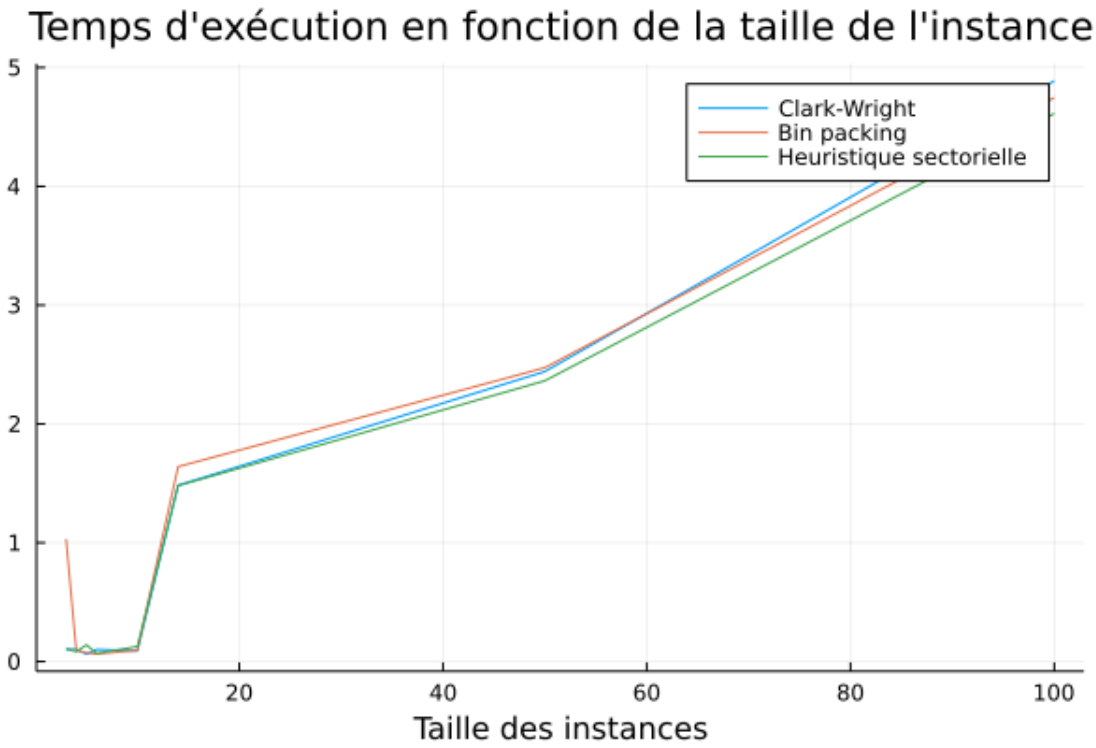

FIGURE 3 – Temps d'exécution en fonction de la taille de l'instance de type A

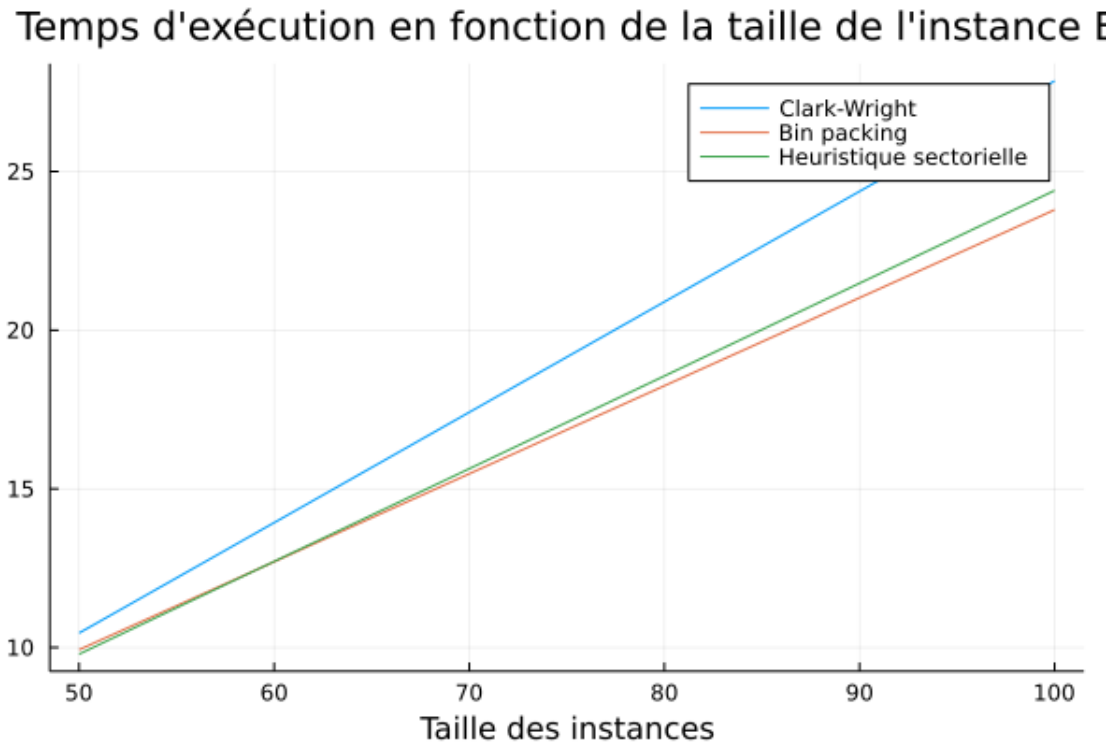

FIGURE 4 – Temps d'exécution en fonction de la taille de l'instance de type B

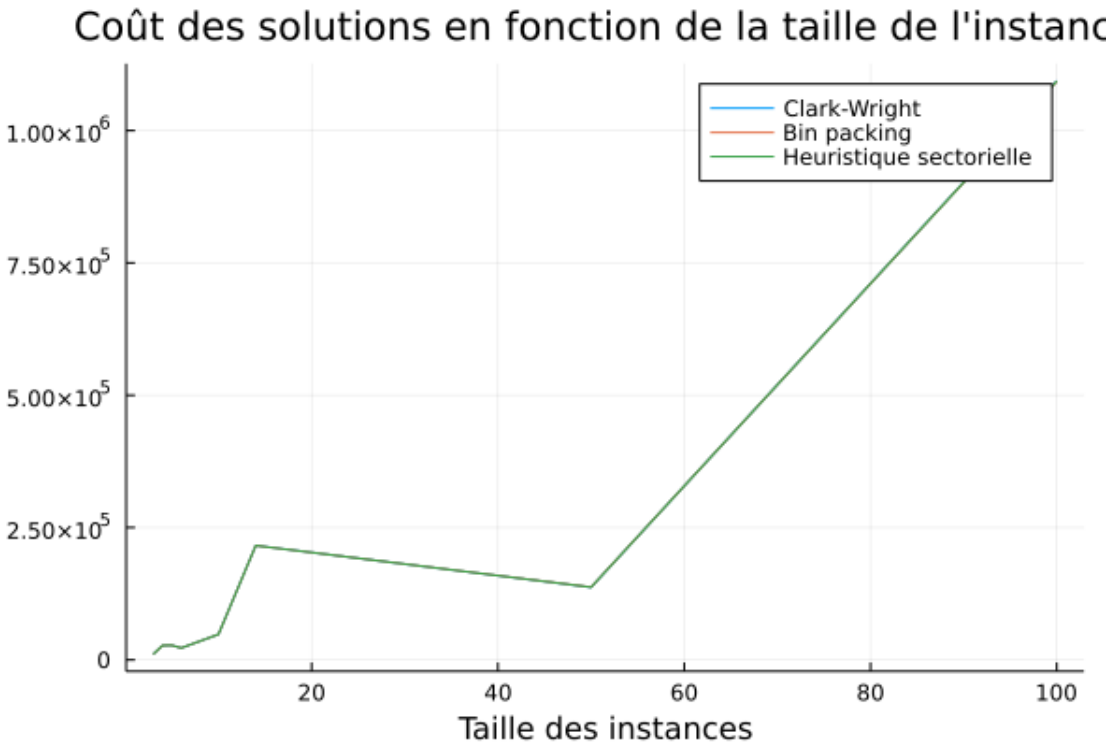

FIGURE 5 – Coût des solutions en fonction de la taille de l'instance de type A

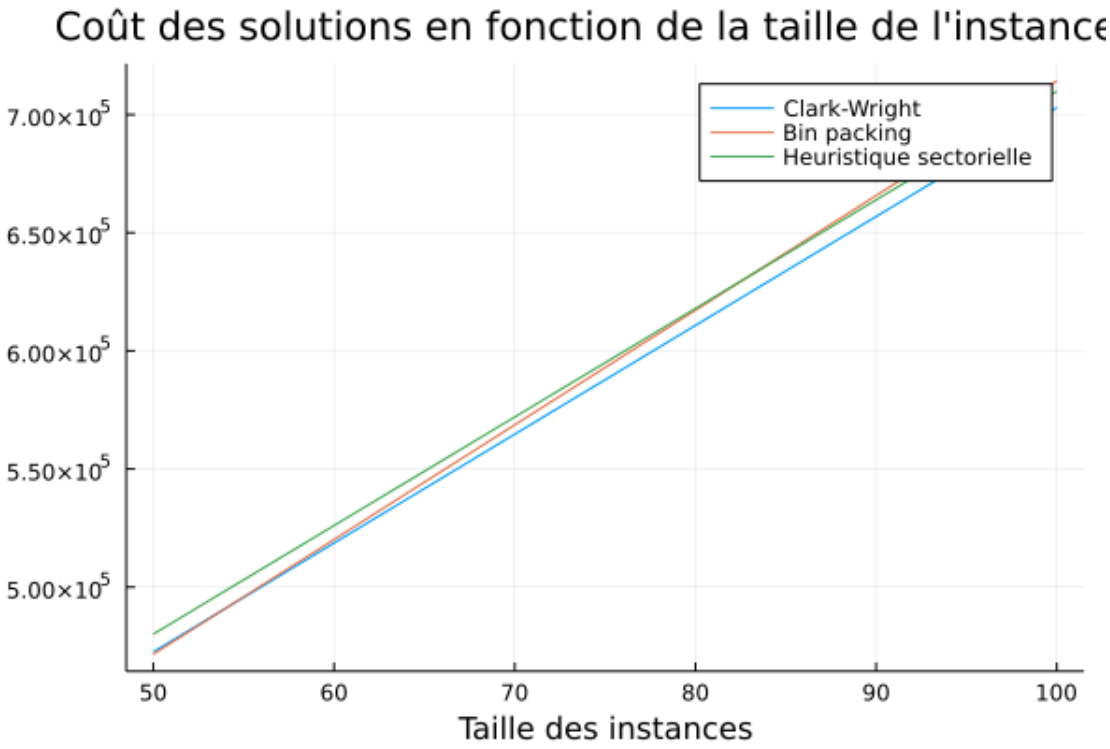

FIGURE 6 – Coût des solutions en fonction de la taille de l'instance de type B

Tout d'abord, on voit que le coût des solutions obtenues ne dépend quasiment pas de l'heuristique choisie mais dépend de la taille de l'instance, ce qui est plutôt

logique. Plus l'instance est grande, plus le coût va être élevé. Les coûts restent assez bas par rapport aux coûts exacts attendus (au plus à un facteur 10), ce qui est très positif quant à la qualité de nos solutions.
Nous voyons ensuite que les temps d'exécution sont sensiblement les même selon l'heuristique choisie bien que l'heuristique Clark-Wright semble être plus coûteuse en temps de calcul. On voit également que les temps d'exécution sont bien plus élevés (5 fois plus élevés) pour des instances de type B que pour des instances de type A, ce qui est également logique étant donné la structure de ces instances. Les temps de calcul restent cependant de l'ordre de la seconde. Nous pouvons donc observer l'efficacité des méthodes heuristique mises en place.

4 Résolution exacte

4.1 PLNE du problème de production et distribution intégré

Nous avons implémenté la formulation [1,16] décrit dans l'article [1] que nous allons nommer PDI1. Et, nous avons implémenté la formulation [21,33] que nous allons nommer PDI2.

4.2 Test sans amélioration

Dans cette partie, nous regardons pour quelle taille d'instances, le PLNE du PDI peut résoudre sans amélioration du PLNE pour les deux PDI.

4.2.1 Taille des Instance

Nous avons instancié des instances de petites tailles et regardé jusqu'à quelle taille d'instances, le problème peut être résolu.

Les instances ont été crée à partir des instances fournies, dans lesquelles nous avons supprimé des revendeurs. Cependant, il se peut que certaines instances ne soient pas réalisables.

Voici les résultats :

Méthode	Nb Client	Sol OPT	Tps Exec	Exact
PLNE PDI 2	3	13 201	0.3323988914489746	oui
PLNE PDI 1	3	13 210	0.12732386589050293	oui
PLNE PDI 2	4	29960	2.093400001525879	oui
PLNE PDI 2	5	20418.0	0.6271119117736816	oui

Table 3 – Solution obtenue avec le PLNE du PDI exact sans Branch and cut

Méthode	Nb Client	Sol OPT	Tps Exec	Exact
PLNE PDI2	10	54781.0000	1482.75	non

Table 4 – Solution obtenue avec le PLNE PDI exact relaxé

Voici le résultat obtenu par le PDI exact sans Branch and cut pour une instance de type A avec 3 revendeurs. Le point bleu correspond au nœud de dépôt. Comme il n'y a aucune ressource à transporter à $t = 1$, aucun graphe n'est généré pour ce pas de temps. Sur le schéma ci-dessous, nous voyons les tournées de véhicule pour chaque pas de temps. Nous voyons clairement que la solution n'est pas optimale puisque toutes les tournées effectuées ne sont composées que d'un revendeur (un camion par revendeur est utilisé) alors que l'on pourrait regrouper les revendeurs dans une tournée pour diminuer le coût.

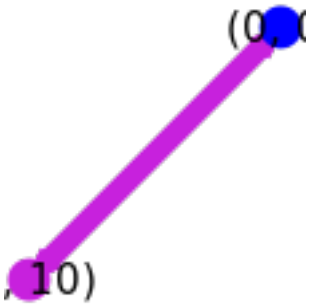

FIGURE 7 – Tournées à l'instant t=2

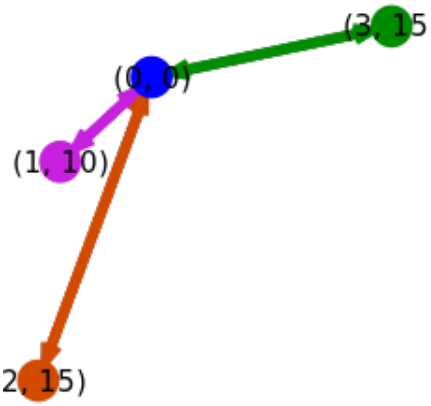

FIGURE 8 – Tournées à l'instant t=3

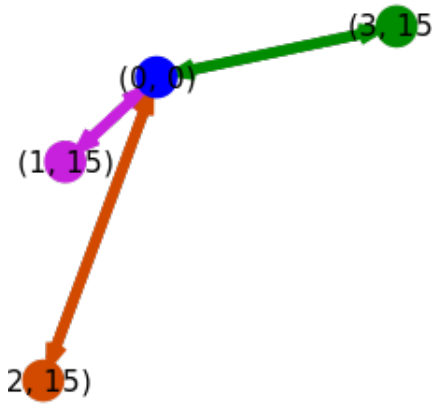

FIGURE 9 – Tournées à l'instant t=4

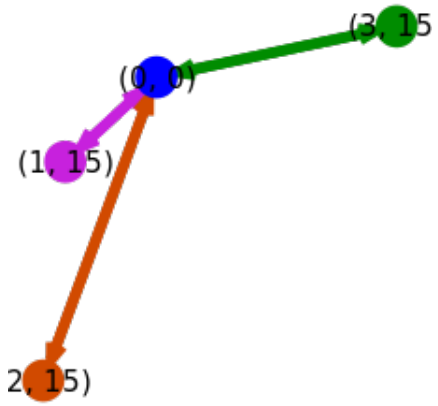

FIGURE 10 – Tournées à l'instant t=5

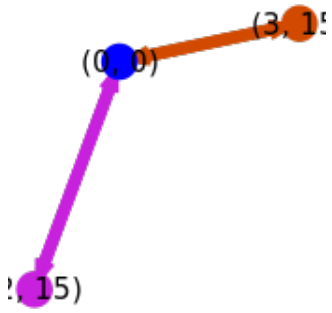

FIGURE 11 – Tournées à l'instant t=6

4.3 Test avec amélioration

Dans cette partie, nous allons utilisé le Branch and Cut.

— Pour le PDI 1, à la place des contraintes (11) et (12), nous allons utiliser la contrainte (19) et la contrainte (17), pour avoir un résultat plus rapide.

— Pour le PDI 2, à la place de la contrainte (29), qui élimine les sous-tours, nous allons utiliser le Branch and Cut en utilisant une autre contrainte (33).

Nous avons écrit des fonctions, pour déterminer les sous-tours. Après chaque call-

back, nous déterminons les sous-tours présents dans la matrice x. Et nous ajoutant les contraintes de coupes pour ces sous-tours.

Pour les instances de taille 50 et plus, nous pouvons seulement avoir un résultat approché de la solution.

Voici le résultat obtenu par le PLNE du PDI1 exact en utilisant un Branch and Cut pour une instance avec 14 revendeurs de type A. Il n'y a pas de ressources à livrer aux instants $t \in \{1, 5, 6\}$.

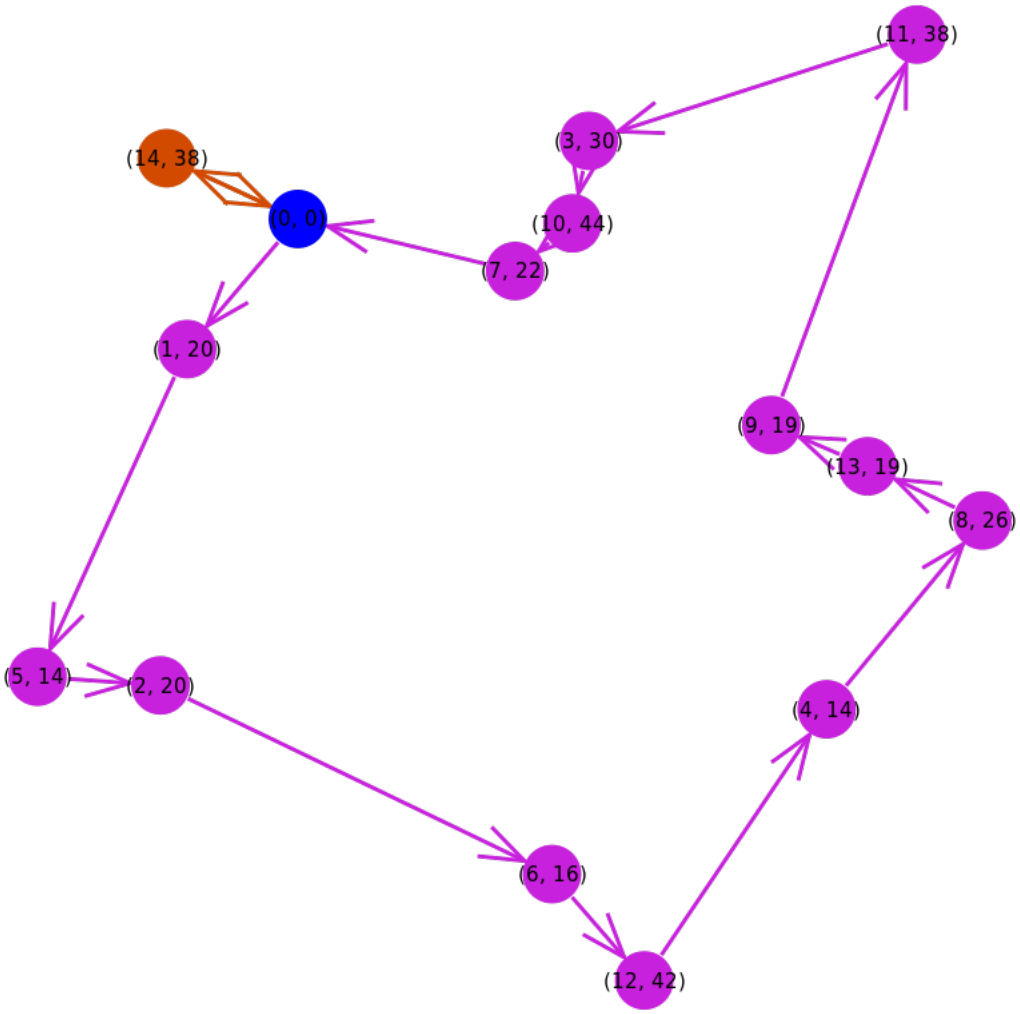

FIGURE 12 – Tournées à l'instant t=2

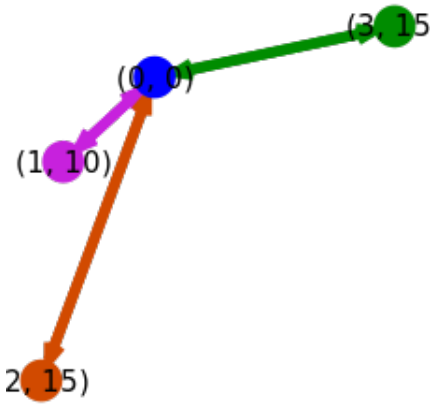

FIGURE 13 – Tournées à l'instant t=3

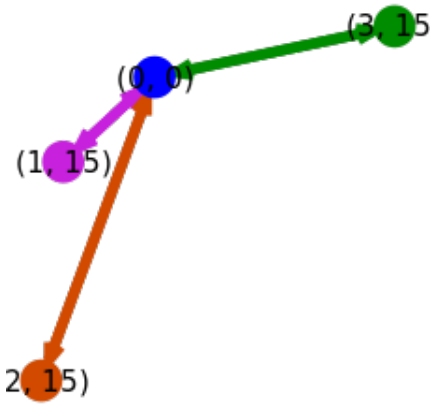

FIGURE 14 – Tournées à l'instant t=4

4.3.1 Résutats

Nous avons instancié des instances de petites tailles et regardé jusqu'à quelle taille d'instances, le problème peut être résolu.

Le tableau ci-dessous représente les résultats du PLNE du PDI exact en utilisant le branch and cut. Nous avons sur la colonne "Nb Client" utilisé une instance de taille "Nb CLient" et la classe de l'instance.

Méthode	Nb Client	Sol OPT	Tps Exec	Exact
PLNE PDI 1	3	13 210	0.09406614303588867	oui
PLNE PDI 1	10	54618.0	152.92014288902283	oui
PLNE PDI 1	4	29960.0	1.6998569965362549	oui
PLNE PDI 1	5	20418.0	0.8970139026641846	oui
PLNE PDI 1	6	18583.0	1.8155968189239502	oui
PLNE PDI 1	6	29739.0	1.62	oui
PLNE PDI 1	14 (1)	73624.0	8.584228992462158	oui
PLNE PDI 1	14 (10)	40929.0	76.48173093795776	oui
PLNE PDI 1	14 (20)	38051.0	241.81	oui
PLNE PDI 1	14 (30)	617510.0	193.84	oui
PLNE PDI 1	14 (40)	621877.0	440.99	oui
PLNE PDI 1	14 (30)	81998.0	2440.18	oui
PLNE PDI 1	14 (60)	114400.0	402.66	oui
PLNE PDI 1	14 (70)	158374.0	493.56	oui
PLNE PDI 1	14 (80)	33114.0	197.57	oui
PLNE PDI 1	14 (90)	601193.0	14.26	oui

TABLE 5 – Solution exacte obtenue avec le PLNE PDI exact utilisant des instances de type A

Méthode	Nb Client	Sol OPT	Tps Exec	Exact
PLNE PDI 1	10	54781.0000	1482.75	non

TABLE 6 – Solution obtenue avec le PLNE PDI1 exact relaxé avec Branch and Cut

5 Analyse

Une solution réalisable renvoyée par le PDI heuristique nous donne une borne supérieure de la solution optimale que nous notons $sol_{heuristique}$. La borne inférieure est donnée par l'exécution du PLNE PDI exact pendant un certain temps que nous notons sol_{relaxe}.

$$gap = \frac{sol_{relaxe} - sol_{heuristique}}{sol_{heuristique}}$$

6 Discussions

Nous allons comparer la méthode heuristique et la méthode exacte. Nous avons donc représenté les données en fonction de la taille de l'instance et de la méthode utilisée (heuristique ou exacte) sur les graphiques suivants :

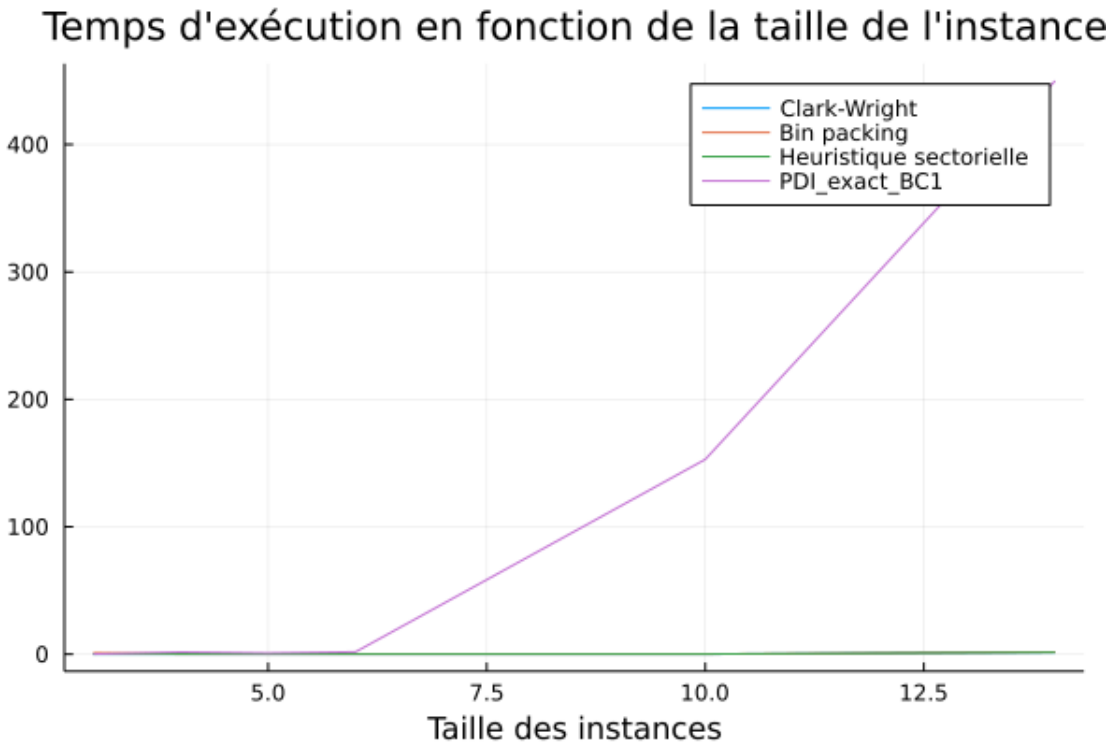

FIGURE 15 – Temps d'exécution en fonction de la taille de l'instance

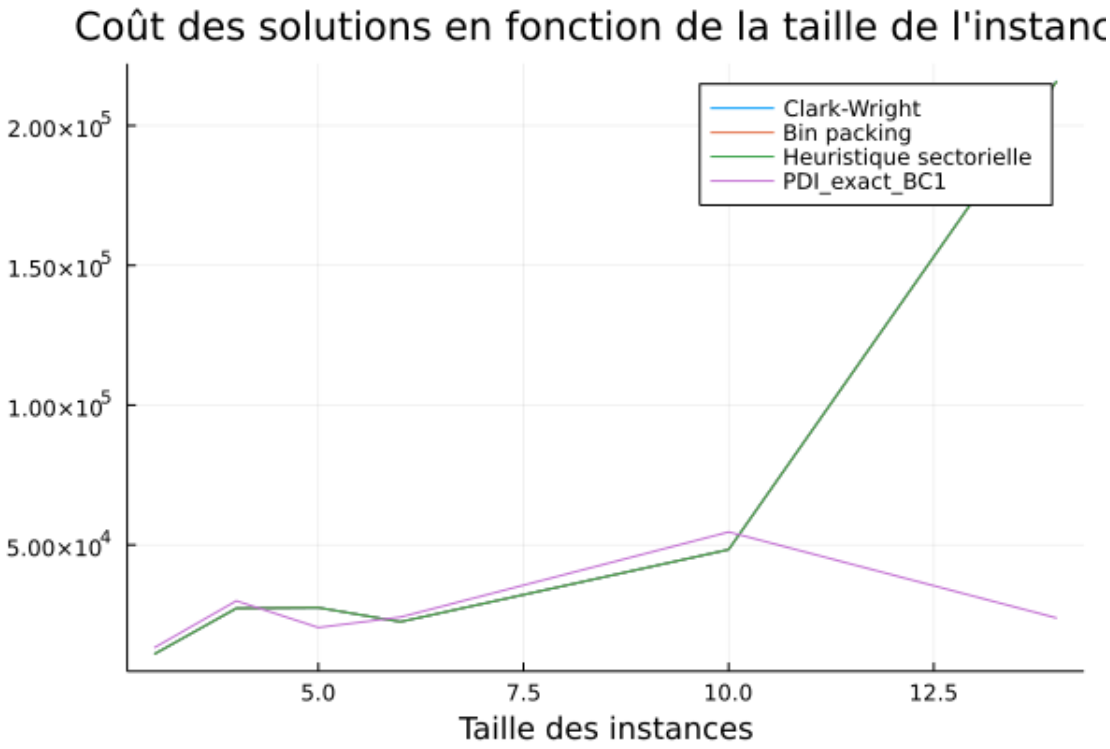

FIGURE 16 – Coût des solutions en fonction de la taille de l'instance de type B

— Méthode heuristique :
Nous avons observé plus haut l'efficacité des méthodes heuristique mises en

place, que ce soit en terme de temps de calcul ou en terme de coût de la solution (plutôt proche de celle obtenue par la méthode exacte, à un facteur au pire 10 près).

— Méthode exacte :

La méthode exacte nous donne une meilleure solution pour des instances petites (tailles 3 et 4), s'il n'y a pas de branch and cut utilisé. Le PDI 1 est plus rapide que le PDI 2 en terme de nombre de contraintes et de temps d'execution.

En faisant un branch and cut, nous pouvons avoir des résultats exacts pour de plus grandes instances.

— Comparaison

Nous observons très bien sur ces graphiques la rapidité des méthode approchées au prix de la qualité des solutions. Il est donc préférable d'utiliser une méthode approchée pour obtenir rapidement une solution quitte à ce qu'elle soit plus coûteuse et plutôt une méthode exacte lorsque l'on n'est pas pressé et que l'on préférerait prendre du temps pour obtenir une solution de bonne qualité.

Références

[1] Y. Adulyasak and J-F Cordeau and R. Jans (2015). The production routing problem : A review of formulations and solution algorithms Computers & Operations Research, 55 :141-152.