Rapport de projet

MAOA - Projet en Optimisation Combinatoire

Autour du problème de Production et Distribution Intégré

Master d'informatique spécialité androide

Deuxième année

Année universitaire 2021 - 2022

Professeur : Pierre Fouilhoux

Étudiants : Vincent Fu
Yuhao Liu

Table des matières

1. Introduction

Le *Production Routing Problem*[1] (PRP) est un problème d'optimisation combinatoire classique dans la littérature en Recherche Opérationnelle.
En raison de la difficulté du problème, résoudre le problème représente un défi. Le but du projet est alors de proposer des méthodes approchées et exactes et de les implémenter afin de mesurer les performances[2] de ces méthodes.

2. Méthodes utilisées

2.1 Résolution heuristique en deux phases

La résolution heuristique consiste à résoudre le *Production Routing Problem* en deux sous-problèmes indépendants en vue de combiner ces solutions pour fournir une solution approché du PRP. En effet, en divisant le problème principal d'optimisation combinatoire en deux sous-problèmes, la solution obtenue dans ce cas-ci ne peut qu'être approchée.

Ces deux sous-problèmes sont :

— Le problème du *Lot-Sizing* (LSP)

— Le *Capacitated Vehicle Routing Problem* (CVRP)

Pour résoudre ces sous-problèmes, on utilise des formulations compactes en *Mixed Integer Linear Programming* (MILP) définies de la manière suivante :

Pour le problème du *Lot-Sizing* :
Données du problème :

— 1 fournisseur (indice 0)

— $\mathcal{N}$ un ensemble de n revendeurs (d'indices 1 à n)

— $\mathcal{T}$ un horizon discret de 1 à l

— $(d_{it})_{i\in\mathcal{N},t\in\mathcal{T}}$ les demandes de chaque revendeur i à la période t

— $(h_i)_{i\in\mathcal{N}}$ les coûts de stockage unitaire chez le revendeur i

— $(L_i)_{i\in\{0\}\cup\mathcal{N}}$ les capacités de stockage maximale pour l'individu i

— f un coût fixe de *setup* pour une période

— u un coût unitaire de production

Variables de décision :

— p_t quantité produite à la période t

— y_t variable binaire indiquant si la production a été lancée à la période t

— I_{it} quantité en stock à la fin de la période t pour l'individu i

— q_{it} quantité produite pour le revendeur i à la période t

1. Voir le sujet en annexe pour plus de détails.
2. Qualité des solutions, temps d'exécution, etc ...

Le programme linéaire en nombres entiers (P_{LSP}) avec des constantes grandes $M_t \gg 0$:

$$\min \quad \sum_{t=1}^{l}(up_t + fy_t + \sum_{i=1}^{n} h_i I_{it}) \tag{1}$$

$$s.t. \quad I_{0t-1} + p_t = \sum_{i=1}^{n} q_{it} + I_{0t} \quad \forall\, t \in \mathcal{T} \tag{2}$$

$$I_{it-1} + q_{it} = d_{it} + I_{it} \quad \forall\, i \in \mathcal{N}, \forall\, t \in \mathcal{T} \tag{3}$$

$$p_t \leq M_t y_t \quad \forall\, t \in \mathcal{T} \tag{4}$$

$$I_{0t-1} \leq L_0 \quad \forall\, t \in \mathcal{T} \tag{5}$$

$$I_{it-1} + q_{it} \leq L_i \quad \forall\, i \in \mathcal{N}, \forall\, t \in \mathcal{T} \tag{6}$$

$$I_{it},\, q_{it},\, p_t \ \in \mathbb{R}^+ \quad \forall\, i \in \mathcal{N}, \forall\, t \in \mathcal{T} \tag{7}$$

$$y_t \in \{0,1\} \quad \forall\, t \in \mathcal{T} \tag{8}$$

Pour le problème du *Vehicule Routing* :
Données du problème en plus des données précédentes :

- $G = (\{0\} \cup \mathcal{N}, A)$ un graphe orienté complet avec $A = \{(i,j) \mid i,j \in \{0\} \cup \mathcal{N}, i \neq j\}$
- $(c_{ij})_{i \in \{0\} \cup \mathcal{N}, j \in \{0\} \cup \mathcal{N}}$ les coûts de transport du sommet i au sommet j
- m le nombre de véhicule
- Q la charge maximale d'une véhicule

Variables de décision :

- x_{ij} variable binaire indiquant si l'arc $(i,j) \in A$ a été emprunté
- w_i variable pour la formulation Miller-Tucker-Zemlin (MTZ)

Le programme linéaire en nombres entiers (P_{VRP}) pour une période t :

$$\min \quad \sum_{(i,j) \in A} c_{ij} x_{ij} \tag{9}$$

$$s.t. \quad \sum_{j=1}^{n} x_{0j} \leq m \tag{10}$$

$$\sum_{i=1}^{n} x_{i0} \leq m \tag{11}$$

$$\sum_{j=0}^{n} x_{ij} = 1 \quad \forall\, i \in \mathcal{N} \tag{12}$$

$$\sum_{i=0}^{n} x_{ij} = 1 \quad \forall\, j \in \mathcal{N} \tag{13}$$

$$w_i - w_j \geq d_{it} - (Q + d_{it})(1 - x_{ij}) \quad \forall\, i \in \mathcal{N}, \forall\, j \in \mathcal{N}, i \neq j \tag{14}$$

$$w_i \leq Q \quad \forall\, i \in \mathcal{N} \tag{15}$$

$$w_i \geq 0 \quad \forall\, i \in \mathcal{N} \tag{16}$$

$$x_{ij} \in \{0,1\} \quad \forall\, (i,j) \in A \tag{17}$$

Pour obtenir une solution approchée pour le PRP, il suffit alors de combiner les solutions du LSP et ceux du VRP de chaque période t en une heuristique itérative en deux phases définie de la manière suivante :
Variable de décision :

— z_{it} variable binaire indiquant si le revendeur i a été visité au période t

Algorithm 1 Résolution itérative en deux phases

Result: Solution approchée pour le PRP
Poser $SC_{it} \leftarrow c_{0i} + c_{i0} \quad \forall\, i \in \mathcal{N}, \forall t \in \mathcal{T}$
$L \leftarrow \emptyset$
while *Critère d'arrêt non atteint* **do**
Résoudre le (P_{LSP}) modifié :

$$\min \quad \sum_{t=1}^{l}(up_t + fy_t + \sum_{i=1}^{n} h_i I_{it} + \sum_{i=1}^{n} SC_{it} z_{it}) \tag{18}$$

$$s.t. \quad \text{contraintes du } (P_{LSP}) \cup \{q_{it} \leq M_t z_{it} \quad \forall\, i \in \mathcal{N}, \forall\, t \in \mathcal{T}\} \tag{19}$$

$$z_{it} \in \{0,1\} \quad \forall\, i \in \mathcal{N}, \forall\, t \in \mathcal{T} \tag{20}$$

for $t \leftarrow 1$ **to** l **do**
Résoudre le (P_{VRP}) pour la période t.
end
Ajouter le couple (solution, $\sum$ valeurs objectives de (P_{LSP}) modifié et des (P_{VRP}) $-\sum_{i=1}^{n} SC_{it} z_{it}$) dans L
Mettre à jour $SC_{it} \leftarrow c_{i^- i} + c_{ii^+} - c_{i^- i^+} \forall\, i \in \mathcal{N}, \forall t \in \mathcal{T}$ avec i^-, i^+ les sommets prédécesseurs, successeurs de i fournis par les solutions du (P_{VRP}) de chaque période t
end
Retourner le couple (solution, valeur objective) ayant la plus petite valeur objective dans L

Le critère d'arrêt peut être un nombre maximum d'itération ou bien un critère sur l'optimalité de la solution. En particulier pour nous, le critère d'arrêt est lorsque on atteint 15 itérations ou bien lorsqu'on est sur un minimum local.

Malgré les formulations compactes des PLNE, l'heuristique en deux phases ne s'avère pas être efficace sur les instances de grande taille.
Sachant que le PLNE du LSP reste efficace même pour les instances de grande taille, on propose une métaheuristique basée sur les algorithmes évolutionnaires pour le VRP à grande échelle.

2.2 Algorithme génétique pour le VRP

Définissons à présent les opérateurs utilisés pour l'algorithme génétique :

— Initialisation d'un individu (méta-tournée) : liste aléatoire de sommets à livrer sans dépôt
— Croisement : *Order Crossover OX*
— Mutation : Échange de deux arêtes non adjacents
— Définition d'une tournée réalisable : Ajouter les séparateurs dans la méta-tournée sous contrainte de charge de véhicule et du nombre de véhicule maximal

Pour mieux comprendre ces opérateurs, voici quelques exemples pour 20 revendeurs :

20	1	2	4	3	5	6	7	10	8	9	11	12	13	14	16	15	17	18	19

FIGURE 1 – *Exemple de méta-tournée*

Les méta-tournées ne sont pas des tournées réalisables. En revanche, si on ajoute des séparateurs de dépôt 0 sous contraintes que les sous-chaînes sans dépôt délimitées par les dépôts vérifient la charge d'un véhicule et que le nombre de ces sous-chaînes soit inférieur ou égal au nombre de véhicule, alors cette tournée est réalisable.

0	20	1	2	4	3	0	5	6	7	10	0	8	9	11	12	13	14	0	16	15	17	18	19	0

FIGURE 2 – *Tournée réalisable issue de la séparation du méta-tournée*

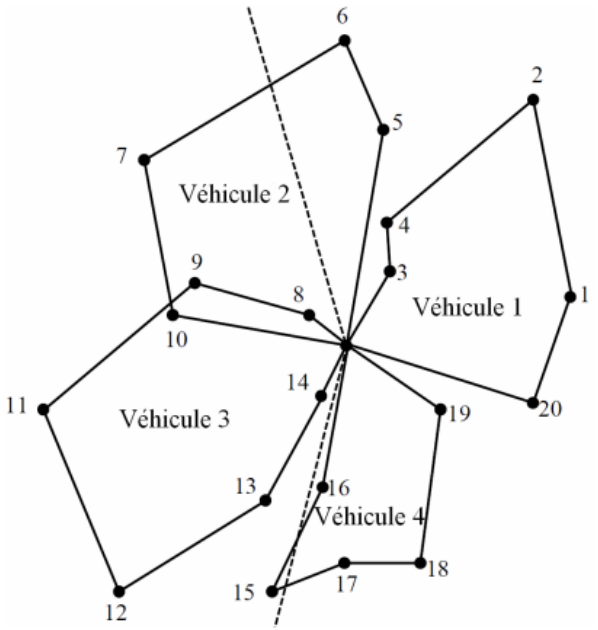

FIGURE 3 – *Graphe de tournée réalisable issu de la séparation du méta-tournée*

En définissant une coupe pour toute taille de méta-tournée, on peut définir un opérateur de croisement comme le *Order Crossover* :

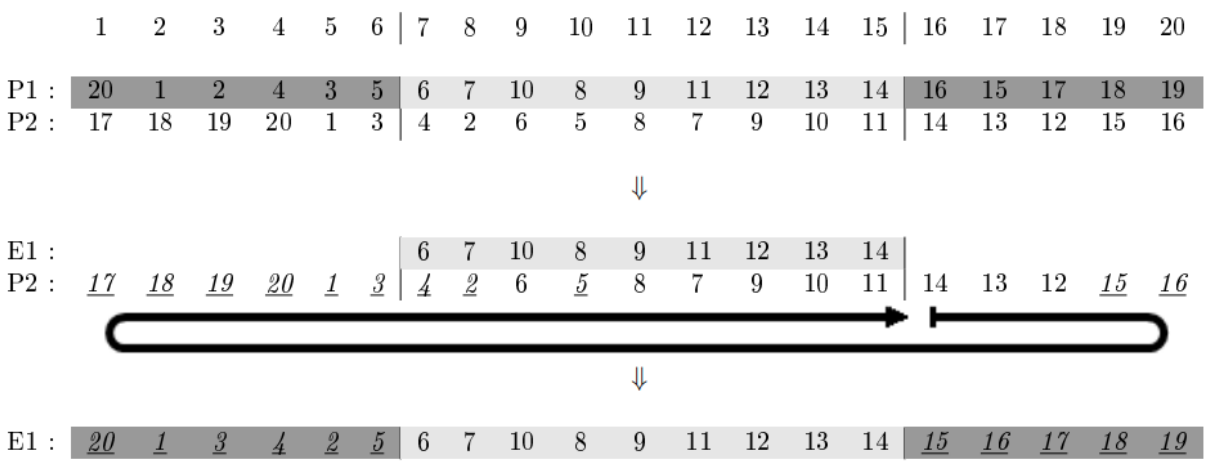

FIGURE 4 – *Exemple de Order Crossover avec les parents P1 et P2*

En définissant une coupe entre la 6ème position et la 15ème position, on remarque que l'enfant E1 hérite du des sommets de la méta-tournée P1 entre les coupes et hérite les sommets de la méta-tournée de P2 en dehors des coupes selon l'ordre d'apparition de ces sommets indiquée par la flèche noire à partir de la 16ème position. On obtient de manière similaire l'enfant E2. Il existe évidemment d'autres opérateurs de croisement qui fonctionne tout aussi bien que l'*Order Crossover*.
Concernant la coupe, nous avons décidé de couper à partir du 1/4 de la taille ème position du tournée et du 3/4 de la taille ème position pour définir un chromosome.

La mutation consiste quant à elle à échanger 2 sommets. En effet, par exemple si on échange le sommet 1 et 3 de E1, on obtient au début du chromosome 20, 3, 1, 4, ce qui correspond à échanger les arcs $20 \to 1$ et $3 \to 4$ en $20 \to 3$ et $1 \to 4$. Comme une tournée est dirigée, il faut en plus d'effectuer une échange d'arcs réparer le circuit en changeant l'arc $1 \to 3$ qui a été supprimé en $3 \to 1$, ce qui est fait implicitement par la mutation.

Ainsi, grâce à ces opérateurs, il est facile de former des tournées réalisables. La question est maintenant savoir quelle *fitness* définir afin de réduire la distance totale parcourue tout en réduisant le nombre de véhicule utilisé.
Berger et al. [1998] ont proposé le fitness suivant :

$$R - R_{min} + \frac{\min(D, 2D_{min})}{D_{min}} \tag{21}$$

où

- R est le nombre de tournées dans la solution à évaluer
- R_{min} est le nombre de tournées dans la meilleure solution dans la population courante
- D est la distance totale parcourue dans la solution à évaluer
- D_{min} est la distance totale parcourue dans la meilleure solution de la population courante

Cette formulation permet après évolution génétique d'obtenir des solutions ayant de petites distances parcourues et en utilisant un minimum de véhicules.
Il est possible également de définir un *fitness* multi-objectifs comme le couple (distance parcourue, nombre de véhicules utilisés) afin de mieux mesurer les solutions obtenues mais en contre-partie la sélection naturelle est légèrement plus complexe comme par exemple dans le cas de *Non-dominated sorting genetic algorithm* II (NSGA II).

2.3 Résolution exacte par *branch and cut*

Pour résoudre de manière exacte le PRP, il est indispensable de regrouper le LSP et le VRP dans un seul PLNE. Cependant, en raison de la formulation non compacte de ce dernier, il faut alors le résoudre dans le cadre d'un *branch and cut*.

Voici le PLNE (P_{PRP}) pour le PRP :
Données du problème en plus des données précédentes :

— I_{i0} quantité initiale dans le stock de l'individu i

Varaibles de décision en plus des variables précédentes sans les x_{ij} :

— z_{0t} le nombre de véhicule quittant le dépôt à la période t

— x_{ijt} variable binaire indiquant si l'arc $(i,j) \in A$ a été emprunté pendant la période t

$$\min \quad \sum_{t\in\mathcal{T}}(up_t + fy_t + \sum_{i\in\{0\}\cup\mathcal{N}} h_i I_{it} + \sum_{(i,j)\in A} c_{ij}x_{ijt}) \tag{22}$$

$$s.t. \quad I_{0t-1} + p_t = \sum_{i\in\mathcal{N}} q_{it} + I_{0t} \quad \forall\, t \in \mathcal{T} \tag{23}$$

$$I_{it-1} + q_{it} = d_{it} + I_{it} \quad \forall\, i \in \mathcal{N}, \forall\, t \in \mathcal{T} \tag{24}$$

$$p_t \leq \min\{C, \sum_{j=t}^{l}\sum_{i\in\mathcal{N}} d_{ij}\} y_t \quad \forall\, t \in \mathcal{T} \tag{25}$$

$$I_{0t} \leq L_0 \quad \forall\, t \in \mathcal{T} \tag{26}$$

$$I_{it-1} + q_{it} \leq L_i \quad \forall\, i \in \mathcal{N}, \forall\, t \in \mathcal{T} \tag{27}$$

$$q_{it} \leq \min\{L_i, Q, \sum_{j=t}^{l} d_{ij}\} z_{it} \quad \forall\, i \in \mathcal{N}, \forall\, t \in \mathcal{T} \tag{28}$$

$$\sum_{i\in\mathcal{N}, t\in\mathcal{T}} q_{it} \leq Q z_{0t} \tag{29}$$

$$\sum_{j\in\{0\}\cup\mathcal{N}} x_{ijt} = z_{it} \quad \forall\, i \in \mathcal{N}, \forall\, t \in \mathcal{T} \tag{30}$$

$$\sum_{j\in\{0\}\cup\mathcal{N}} x_{jit} + \sum_{j\in\{0\}\cup\mathcal{N}} x_{ijt} = 2z_{it} \quad \forall\, i \in \{0\}\cup\mathcal{N}, \forall\, t \in \mathcal{T} \tag{31}$$

$$z_{0t} \leq m \quad \forall\, t \in \mathcal{T} \tag{32}$$

$$p_t,\, I_{it},\, q_{it} \;\geq 0 \quad \forall\, i \in \{0\}\cup\mathcal{N}, \forall t \in \mathcal{T} \tag{33}$$

$$y_t,\, x_{ijt} \in \{0,1\} \quad \forall\, i,\, j \in \{0\} \cup \mathcal{N},\, \forall\, t \in \mathcal{T} \tag{34}$$
$$z_{it} \in \{0,1\} \quad \forall\, i \in \mathcal{N},\, \forall\, t \in \mathcal{T} \tag{35}$$
$$z_{0t} \in \mathbb{Z}^+ \quad \forall\, t \in \mathcal{T} \tag{36}$$

Cette formulation (P_{PRP}) sans *cut* ne permet pas d'obtenir une tournée réalisable en raison des possibles apparitions de sous-tours.
On ajoute alors des contraintes de séparation pour le *branch and cut* afin d'éliminer ces sous-tours.
Deux manières sont possibles pour éliminer les sous-tours :

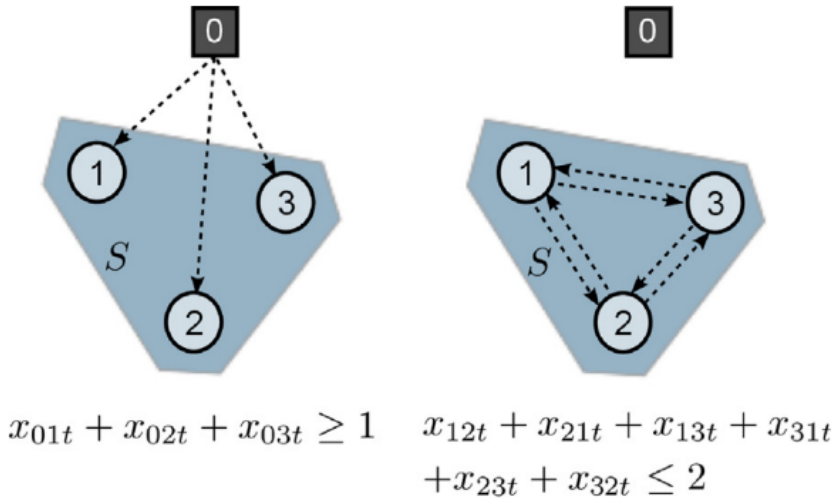

FIGURE 5 – *Exemples de séparation : FFCs à gauche et GFSECs à droite*

Soit S l'ensemble des sommets d'un sous-tour,

— Les *fractional capacity constraints* (FFCs) permettent de garantir la connexité du chemin solution obtenue :

$$\sum_{i \notin S,\, j \in S} x_{ijt} \geq \frac{\sum_{i \in S} q_{it}}{Q} \quad \forall\, S \subseteq \mathcal{N},\, |S| \geq 1,\, \forall\, t \in \mathcal{T} \tag{37}$$

— Les *generalized fractional subtour elimination constraints* (GFSECs) permettent de briser les sous-tours en imposant un nombre d'arc limite strictement inférieur aux nombre d'arcs du sous-tour S :

$$\sum_{i,\, j \in S} x_{ijt} \leq |S| - \frac{\sum_{i \in S} q_{it}}{Q} \quad \forall\, S \subseteq \mathcal{N},\, |S| \geq 2,\, \forall\, t \in \mathcal{T} \tag{38}$$

Ainsi, en combinant ces contraintes de séparation et (P_{PRP}) dans un algorithme de *branch and cut*, on arrive à obtenir des solutions optimales pour des instances de taille petite à taille moyenne. En effet en pratique, pour les instances de grande taille, l'algorithme de *branch and cut* qu'avec seulement une contrainte de séparation n'est pas suffisante pour obtenir une solution dans un temps court.

3. Résultats expérimentaux

Pour tous les résultats expérimentaux qui sont présenté dans cette partie, nous avons fixé l'*integrality tolerance* à 10^{-10} pour la méthode exacte afin d'éviter au maximum les problèmes que peuvent engendrer les constantes *big M*.
De plus, voici quelques détails sur les classes d'instance :

— Classe 1 : classe de référence

— Classe 2 : classe ayant un coût de production variable 10 fois plus élevé

— Classe 3 : classe ayant un coût de trajet 5 fois plus grand

— Classe 4 : classe ayant un coût de stockage nul

3.1 Résultats pour les instances de taille 15

Pour les instances de taille 15, nous avons limité le temps d'exécution de CPLEX à 5 mins.

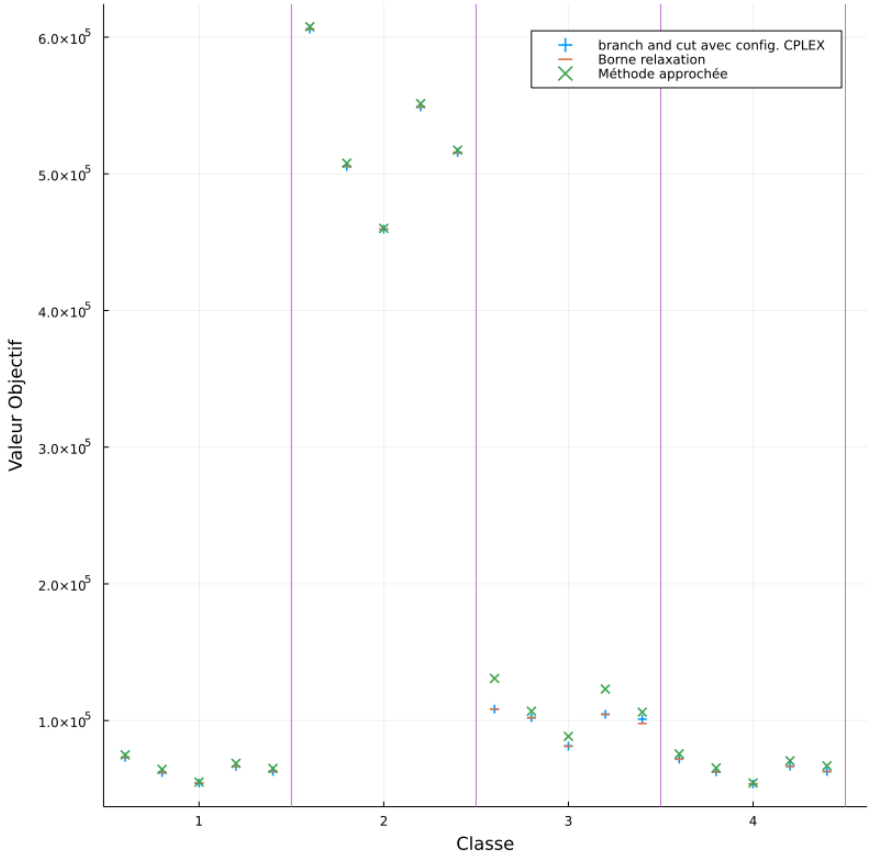

FIGURE 6 – *Graphe des valeurs objectifs en fonction des classes des instances de type A de taille 15*

D'après les résultats ci-dessus, la méthode exacte est toujours meilleure que la méthode approchée sur toutes les classes d'instance. La méthode exacte est également capable de renvoyer une solution ayant une *gap* relative proche de 0 en moins de 5 mins. Cependant, la

méthode approchée reste très performante sur les instances de petites tailles.

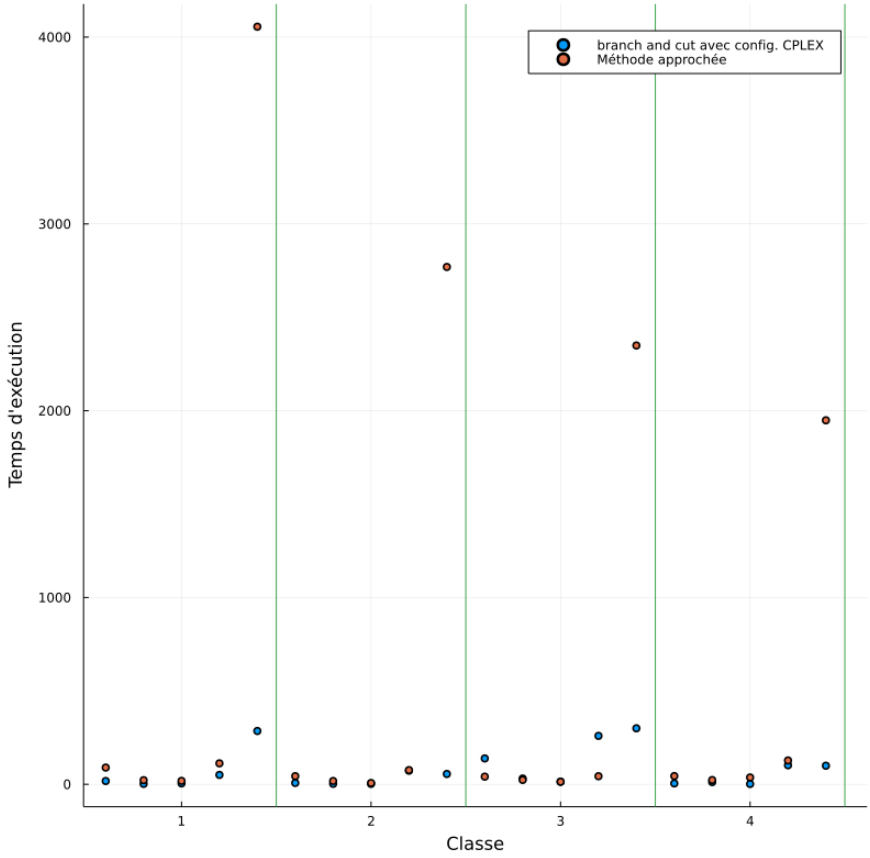

FIGURE 7 – *Graphe des temps d'exécution en fonction des classes des instances de type A de taille 15*

Le temps d'exécution de la méthode exacte et de la méthode approchée sont relativement faibles. Néanmoins, la méthode approchée peut être très lente comme dans le cas de la 5ème instance de la classe 1 en raison du critère d'arrêt et de la structure du problème liée à l'instance.

3.2 Résultats pour les instances de taille 50

Pour les instances de taille 50, en raison de la durée d'exécution pour obtenir une solution optimale, nous avons limité la recherche de solutions en maximum 5 solutions réalisables et de ne retourner que le meilleur parmi les 5.

D'après les résultats ci-dessous, la méthode exacte est meilleure que la méthode approchée sur les classes 1 et 2 alors que l'inverse se produit sur les classes 3 et 4. En effet, on remarque que parmi les 5 premiers solutions réalisables trouvés par l'algorithme, ces valeurs sont très loin des optima alors que ce n'est pas le cas pour la classe de référence. Néanmoins, les bornes des relaxations sont toujours meilleures que les solutions approchées. Cela implique donc que

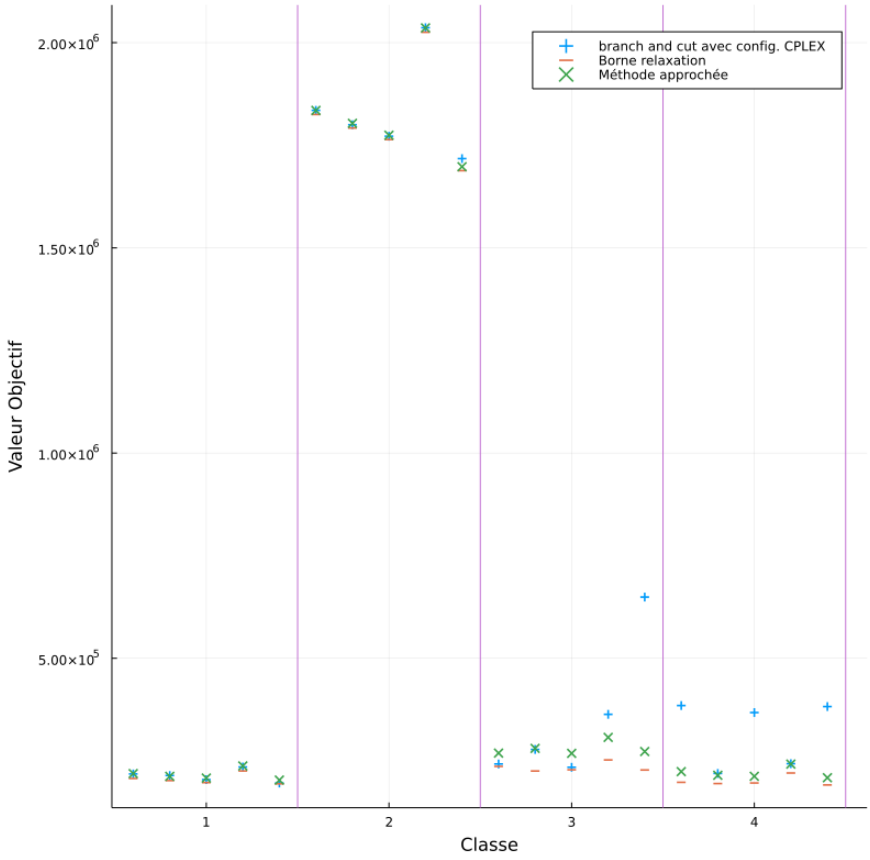

FIGURE 8 – *Graphe des valeurs objectifs en fonction des classes des instances de type A de taille 50*

la méthode exacte a besoin beaucoup de temps d'exécution pour se rapprocher de la solution exacte. En pratique, il arrive parfois qu'on dépasse les 5 mins d'exécution.

En observant les temps d'exécution ci-dessous, la méthode exacte est capable dans la plupart des cas de fournir une solution réalisable en un minimum de temps. Néanmoins, il est intéressant d'observer que la métaheuristique garde un très bon compromis entre temps d'exécution et solution approchée puisque dans la plupart des cas, cette dernière est capable de fournir une solution proche de la borne de relaxation aux alentours de 4 mins et ce peu importe la structure du problème.

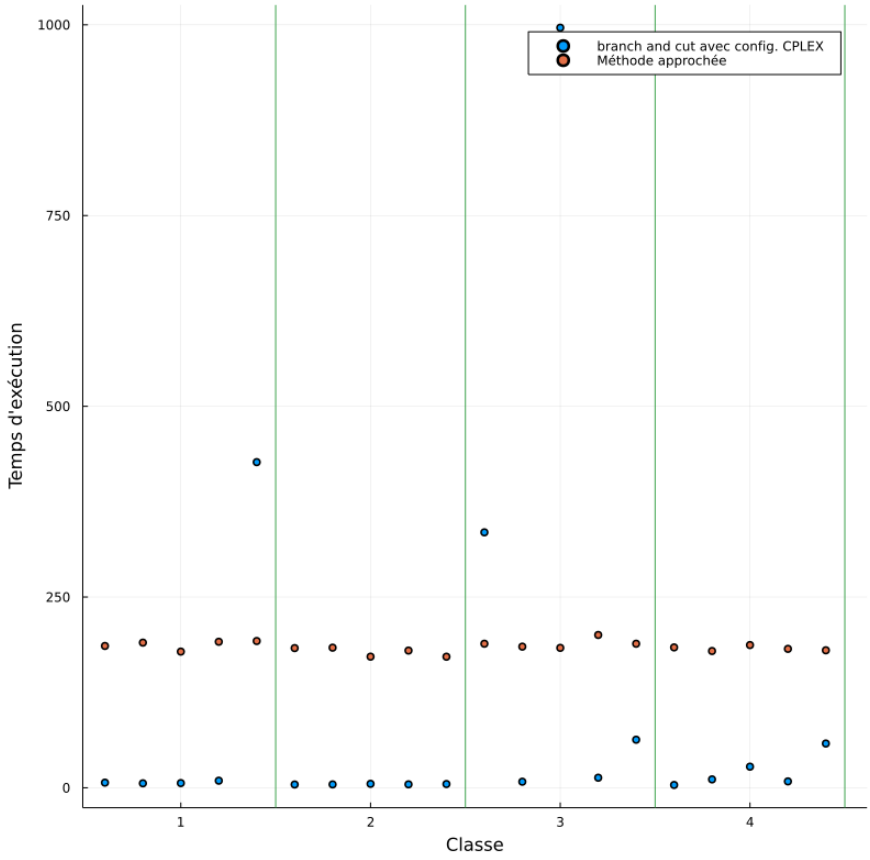

FIGURE 9 – *Graphe des temps d'exécution en fonction des classes des instances de type A de taille 50*

3.3 Résultats pour les instances de taille 100

Pour les instances de taille 100, la méthode exacte n'est pas efficace en raison de la complexité combinatoire. En pratique, la méthode exacte dans ce cas-ci ne fournie pas de solution avant un grand temps d'exécution. Pour avoir un ordre d'idée, l'algorithme n'a toujours pas fourni de solution après plus de 4 heures d'exécution. Il est donc inenvisageable d'étudier la méthode exacte pour les instances de grandes tailles.

On remarque ci-dessous que le temps d'exécution de la méthode heuristique reste raisonnable par rapport à la taille des instances.

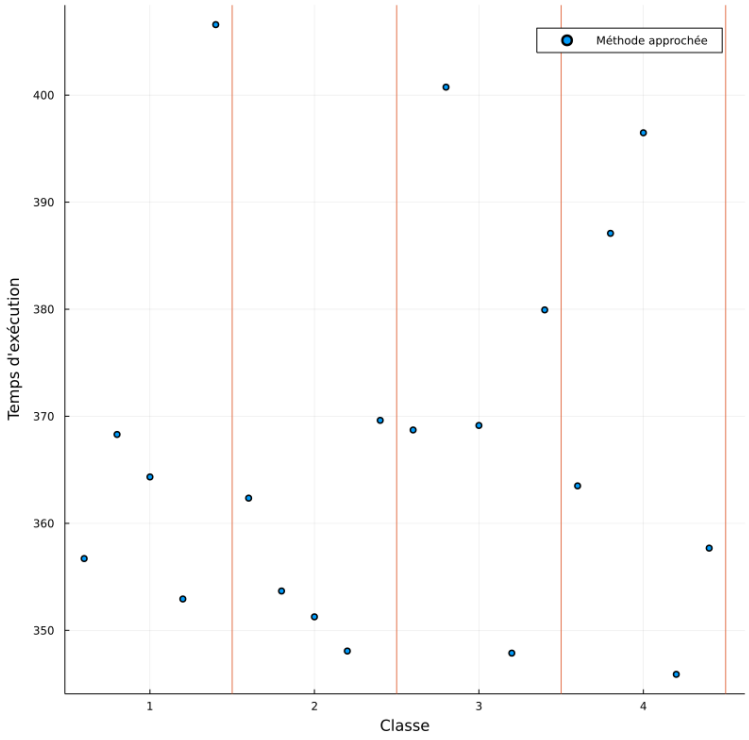

FIGURE 10 – *Graphe des temps d'exécution en fonction des classes des instances de type A de taille 100*

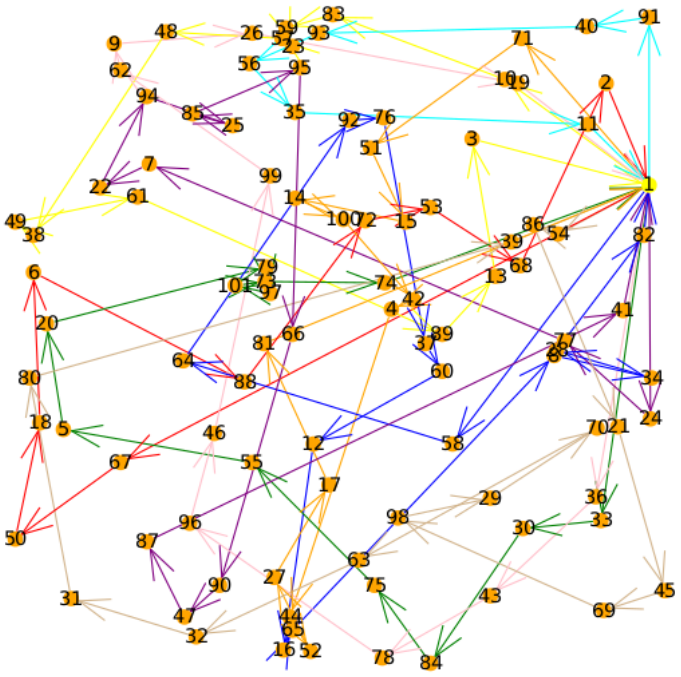

FIGURE 11 – *Exemple de tournée réalisable obtenue par l'algorithme génétique pour une instance de taille 100*

3.4 Analyses de la méthode approchée et de la méthode exacte

Dans cette partie, on souhaite comprendre pourquoi la méthode approchée malgré le fait que sa solution est très proche de la solution exacte n'est pas aussi précise que la méthode exacte en terme de valeur objectif optimale.
En observant les tournées réalisables graphiquement, on peut déjà donner quelques éléments de réponses.

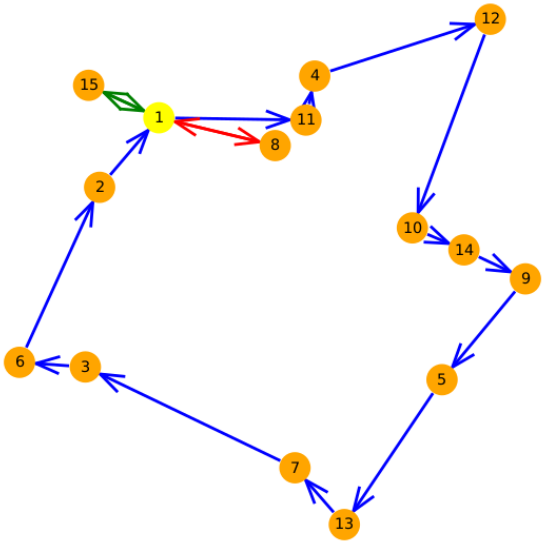

FIGURE 12 – *Tournée réalisable de la période 2 pour l'instance A_014_ABS36_15_1 obtenue par la méthode exacte*

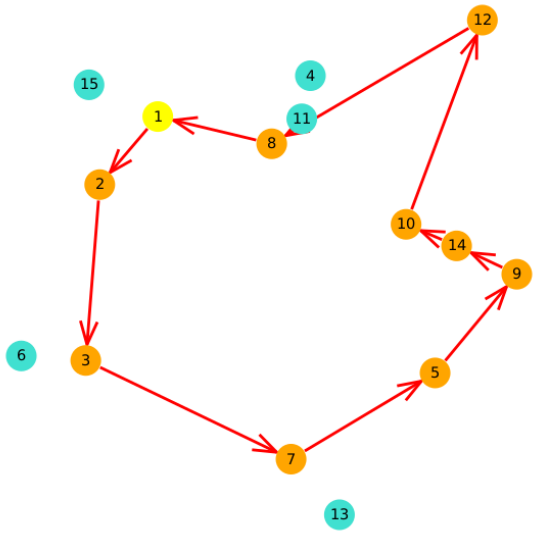

FIGURE 13 – *Tournée réalisable de la période 2 pour l'instance A_014_ABS36_15_1 obtenue par la méthode heuristique en deux phases*

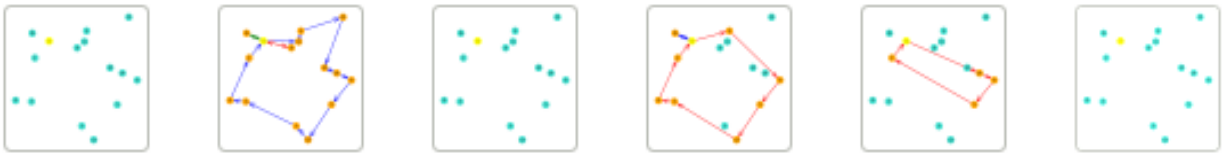

FIGURE 14 – *Liste des tournées réalisables pour l'instance A_014_ABS36_15_1 obtenue par la méthode exacte*

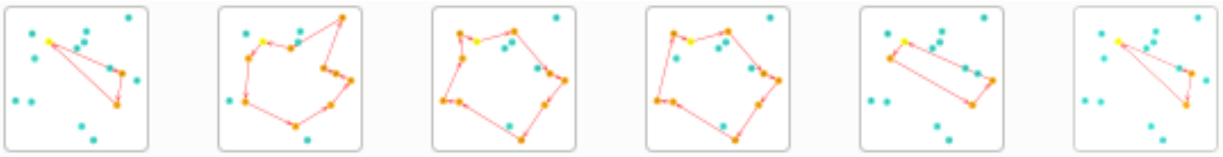

FIGURE 15 – *Liste des tournées réalisables pour l'instance A_014_ABS36_15_1 obtenue par la méthode heuristique en deux phases*

En effet, on observe que dans la méthode heuristique, le nombre de véhicule utilisé va être minimisé. En contre-partie, il est nécessaire de livrer aux revendeurs à quasiment tous les périodes. Or, dans la méthode exacte, on ne cherche pas à réduire le nombre de véhicule utilisé mais à minimiser la valeur objectif, ce qui permet en terme de décision d'essayer de réduire le nombre de périodes à livrer tout en minimisant la distance parcourue.

On en déduit ainsi que par le caractère global dans la prise des décisions du programme linéaire de la méthode exacte, cette dernière est nécessairement meilleur en terme de valeur objectif que par rapport à la méthode heuristique où seuls les caractères locaux issus des deux programmes linéaires sont pris en compte.

4. Conclusion

Lors de ce projet, nous avons pu implémenter différentes méthodes approchées et exactes pour le PRP, comparer leurs efficacités et en expliquer les raisons. De la résolution heuristique en deux phases à la résolution exacte par *branch and cut* en passant par un algorithme génétique, toutes ces méthodes se relèvent être efficaces pour des instances de taille raisonnable. Cependant, la difficulté de traiter des instances de très grande taille reste de l'actualité d'où l'intérêt de la recherche en Recherche Opérationnelle pour ces types de problèmes afin de développer de nouvelles techniques pour les résoudre.

5. Annexes et Références

Notre lien github sur le projet :
`https://github.com/Vincent-Fu-Lab/Production_Routing_Problem`

Le lien du site de l'UE :
`http://www-desir.lip6.fr/~fouilhoux/MAOA/`

Le lien de téléchargement de l'article sur l'algorithme génétique pour le VRP :
`https://portail.telecom-bretagne.eu/publi/public/fic_download.jsp?id=5745`

Références

[1] Yossiri Adulyasak, Jean-François Cordeau et Raf Jans. "The production routing problem : A review of formulations and solution algorithms". en. In : *Computers & Operations Research* 55 (mars 2015), p. 141-152. issn : 0305-0548. doi : 10.1016/j.cor.2014.01.011. url : https://www.sciencedirect.com/science/article/pii/S0305054814000240.