Projet MAOA
Autour du problème de Production et Distribution Intégré

December 9, 2022

Etudiant: Muyang Shi 28714332

Professeur: M.Pierre Foulihoux

Table des matières

1 Introduction

1.1 Résumé

Ce projet s'intéresse au *problème de production et distribution intégré* (PDI) avec véhicules identique, dont la description détaillée est donnée dans la partie suivante. L'objectif de ce problème est de minimiser les coûts dans une chaîne de production d'un seul type. Il s'agit de la jonction de deux problèmes classiques, le *problème de lot-sizing* (LSP) et le problème de tournée de véhicules (VRP).

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 de ces méthodes.

1.2 Description des problèmes traités

1.2.1 Problème de lot-sizing (LSP)

On utilise dans ce projet une variante du problème LSP qui essaie de prendre en compte les prix de visite des clients. Cela est motivé par le fait que cette variante est utilisée dans les solutions approchées au PDI mises en œuvre sur ce projet. Basée sur la formulation (1)–(5) de l'énoncé du projet, la formulation utilisée prend aussi en compte une heuristique du coût de transport et s'écrit comme un PLNE de la façon suivante :

$$
\begin{aligned}
\min \quad & \sum_{t=1}^{l}(up_t + fy_t + \sum_{i=1}^{n} h_i I_{it}) \\
s.t. \quad & I_{0,t-1} + p_t = \sum_{i=1}^{n} q_{it} + I_{0,t} \qquad \forall t \in \{1,\dots,l\} && (1)\\
& I_{i,t-1} + q_{it} = d_{it} + I_{i,t} \qquad \forall i \in \{1,\dots,n\}, \forall t \in \{1,\dots,l\} && (2)\\
& p_t \leq M_t y_t \qquad \forall t \in \{1,\dots,l\} && (3)\\
& I_{0,t-1} \leq L_0 \qquad \forall t \in \{1,\dots,l\} && (4)\\
& I_{i,t-1} + q_{it} \leq L_i \qquad \forall i \in \{1,\dots,n\}, \forall t \in \{1,\dots,l\} && (5)\\
& I_{it}, q_{it} \in \mathbb{R}^+ \qquad \forall i \in \{1,\dots,n\}, \forall t \in \{1,\dots,l\} \\
& p_t \in \mathbb{R}^+, y_t \in \{0,1\} \qquad \forall t\{1,\dots,l\}
\end{aligned}
$$

FIGURE 1 – Formulation classique du problème LSP

Donnée	Indices	Description
1	0	fournisseur
$\mathbb{N}$	1 à n	un ensemble de n revendeurs
τ	1 à l	un horizon discret
$(d_{it})_{i\in\mathbb{N},t\in\tau}$		les demandes de chaque revendeur i à la période t
$(h_i)_{i\in\mathbb{N}}$		les coûts de stockqge unitqire chez le vendeur i
$(L_i)_{i\in 0\cup\mathbb{N}}$		les capacités de stockage maximale pour l'individu i
f	1 à l	un coût fixe de *setup* pour une période
u	1 à l	un coût unitaire de production

TABLE 1 – Données du problème LSP

Variable	Description
p_t	quantité produite à la période t
y_t	variable binaire inquant si la production a été lancée à la période t
I_t	quantité en stock à la fin de la période t pour l'individu i
q_{it}	quantité produite pour le vendeur i à la période t

TABLE 2 – Variables de décision du probème LSP

1.2.2 Problème de tournée de véhicules (VRP)

On présente maintenant la formulation du problème de tournée de véhicules utilisée dans ce projet, qui, comme le LSP de la section précédente, est utilisée dans les solutions approchées au PDI mises en œuvre sur ce projet. Dans le problème VRP, on souhaite livrer des produits à une certaine quantité de clients, qui peuvent ne pas être tous les clients $\{1, ..., n\}$. Pour avoir une notation plus claire, on dénote ici par Nc l'ensemble des clients à être livrés et $N = Nc \cup 0$ l'ensemble contenant ces mêmes clients et le fournisseur. Le problème VSP peut alors se formuler, comme à l'énoncé du projet, par le PLNE suivant :

$$
\begin{aligned}
\min \sum_{(i,j)\in A} c_{ij}x_{ij} & & \\
\sum_{j=1}^{n} x_{0j} \leq m & & (6)\\
\sum_{i=1}^{n} x_{i0} \leq m & & (7)\\
\sum_{j=0}^{n} x_{ij} = 1 & \quad \forall i \in \mathcal{N}_C, & (8)\\
\sum_{i=0}^{n} x_{ij} = 1 & \quad \forall j \in \mathcal{N}_C, & (9)\\
w_i - w_j \geq d_i - (Q + d_i)(1 - x_{ij}) & \quad \forall i \in \mathcal{N}_C, \forall j \in \mathcal{N}_\mathcal{C}, & (10)\\
0 \leq w_i \leq Q & \quad \forall i \in \mathcal{N}_C & \\
w_i \in I\!R & \quad \forall i \in \mathcal{N}_C & \\
x_{ij} \in \{0,1\} & \quad \forall (i,j) \in A. &
\end{aligned}
$$

FIGURE 2 – Formulation classique du problème VRP

Donnée	Indices	Description
$G = (0 \cup \mathbb{N}, A)$		un graphe orienté complet avec $A = \{(i,j) \vert i,j \in 0 \cap \mathbb{N}, i \neq j\}$
c_{ij}	$i,j \in 0 \cup \mathbb{N}$	les coûts de transport du sommet i au sommet j
m		le nombre devéhicule
Q		la charge maximale d'une véhicule

TABLE 3 – Données en plus du problème VRP

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

TABLE 4 – Variables de décision du problème VRP

1.2.3 Problème de production et distribution (PDI)

Le problème de production et distribution intégré prend en compte, dans un seul problèmed'optimisation, les étapes d'optimiser la pro-

duction et d'optimiser la distribution. On présente ici une formulation utilisée dans ce projet.

$$
\begin{aligned}
\min \quad & \sum_{t\in\mathcal{T}}(up_t + fy_t + \sum_{i\in\{0\}\cup\mathcal{N}} h_i I_{it} + \sum_{(i,j)\in A} c_{ij}x_{ijt}) \\
s.t. \quad & I_{0t-1} + p_t = \sum_{i\in\mathcal{N}} q_{it} + I_{0t} \quad \forall\, t \in \mathcal{T} \\
& I_{it-1} + q_{it} = d_{it} + I_{it} \quad \forall\, i \in \mathcal{N}, \forall\, t \in \mathcal{T} \\
& p_t \leq \min\{C, \sum_{j=t}^{l}\sum_{i\in\mathcal{N}} d_{ij}\} y_t \quad \forall\, t \in \mathcal{T} \\
& I_{0t} \leq L_0 \quad \forall\, t \in \mathcal{T} \\
& I_{it-1} + q_{it} \leq L_i \quad \forall\, i \in \mathcal{N}, \forall\, t \in \mathcal{T} \\
& q_{it} \leq \min\{L_i, Q, \sum_{j=t}^{l} d_{ij}\} z_{it} \quad \forall\, i \in \mathcal{N}, \forall\, t \in \mathcal{T} \\
& \sum_{i\in\mathcal{N},\, t\in\mathcal{T}} q_{it} \leq Q z_{0t} \\
& \sum_{j\in\{0\}\cup\mathcal{N}} x_{ijt} = z_{it} \quad \forall\, i \in \mathcal{N}, \forall\, t \in \mathcal{T} \\
& \sum_{j\in\{0\}\cup\mathcal{N}} x_{jit} + \sum_{j\in\{0\}\cup\mathcal{N}} x_{ijt} = 2z_{it} \quad \forall\, i \in \{0\}\cup\mathcal{N}, \forall\, t \in \mathcal{T} \\
& z_{0t} \leq m \quad \forall\, t \in \mathcal{T} \\
& p_t,\, I_{it},\, q_{it} \;\geq 0 \quad \forall\, i \in \{0\}\cup\mathcal{N}, \forall t \in \mathcal{T} \\
& \\
& y_t,\, x_{ijt} \in \{0,1\} \quad \forall\, i,\, j \in \{0\}\cup\mathcal{N}, \forall\, t \in \mathcal{T} \\
& z_{it} \in \{0,1\} \quad \forall\, i \in \mathcal{N}, \forall\, t \in \mathcal{T} \\
& z_{0t} \in \mathbb{Z}^+ \quad \forall\, t \in \mathcal{T}
\end{aligned}
$$

FIGURE 3 – Formulation simple du problème PDI

Donnée	Indices	Description
I_{i0}		quantité initiale dans le stock de l'individu i

TABLE 5 – Données en plus du problème VRP

Variable	Description
x_{ijt}	variable binaire indiquant si l'arc $(i, j) \in A$ a été emprunté pendant la période t
z_{0t}	le nombre de véhicules quittant le dépôt à la période t

TABLE 6 – Variables de décision du problème VRP

2 Implementation

2.1 Résolution heuristique en 2 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).

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 :

Algorithm 1 Résolution itérative en deux phases

$SC_{it} \leftarrow c_{0i} + c_{i0}$
$L \leftarrow \emptyset$
while Critère d'arrêt non atteint **do**
  Résourdre le P_{LSP} modifié :
  min $\sum_{t=1}^{l} \left(up_t + fy_t + \sum_{i=1}^{l} h_n I_{it} + \sum_{i=1}^{l} SC_{itZ_{it}} \right)$
  s.t. contraintes du $P_{LSP} \cup \{q_{it} \leq M_{tZ_{it}}\}$ avec $z_{it} \in \{0, 1\}$
  for $i = 1\,;\, i <= l\,;\, i++$ **do**
    Résourdre le P_{VRP} pour le période t
  end for
  Ajouter le couple (solution, $\sum$ valeur objectives de P_{LSP} modifié et des $P_{VRP} - \sum_{i=1}^{n} SC_{itZ_{it}}$ dans L ;
  Metter à jour $SC_{it} \leftarrow c_{i-i} + c_{ii} - c_{i-i} + \forall i \in \mathbb{N}, \forall t \in \tau$ avec i^-, i^+ les sommets prédécesseurs, successeurs de i fornis par les solutions du P_{VRP} de chaque période t
end while
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'op- timalité 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 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*.

La 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 :

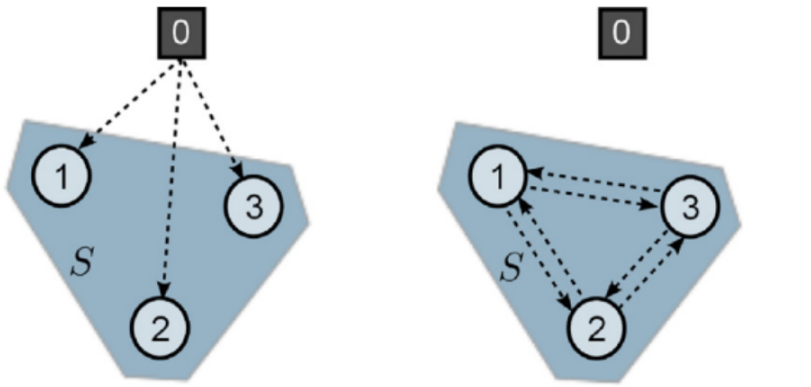


FIGURE 4 – Exemples de séparation : FFCs à gauche et GFSECs à droite

Soit S l'ensemble de sommets d'un sous-retour :

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

$$\sum_{i \in S, j \notin S} x_{ijt} \geq \frac{\sum_{i \in S} q_{it}}{Q}$$
$$\forall S \subseteq \mathbb{N}, |S| \geq 1, \forall t \in \tau$$

- Les *generailized fractional subtour elimination constraints* 5GFSECs) permettenr de briser les sous-tours en imposant un nombre d'arc limite strictement inférieur aux nombre d'arcs du sous-tour S :

$$\sum_{i,j \in S} x_{ijt} \leq |S| - \frac{\sum_{i \in S} q_{it}}{Q}$$
$$\forall S \subseteq \mathbb{N}, |S| \geq 2, \forall t \in \tau$$

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 Evaluation Expérimentale

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*.

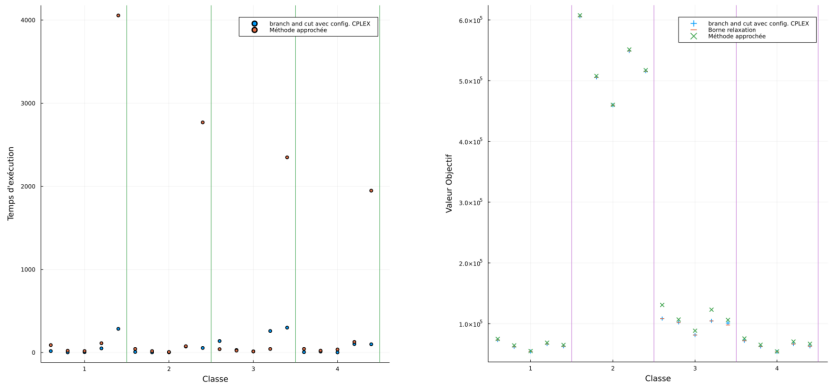
De plus, voici quelques détails sur les classes d'instance :

- 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 Résultat des algorithmes par rapport la taille des instances

3.1.1 Résultat sur des petites instances de type A (taille =15)

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



(a) Temps d'ecécution (b) Valeurs objectifs

FIGURE 5 – Graphe en fonction des classes des instances de type A de taille 15

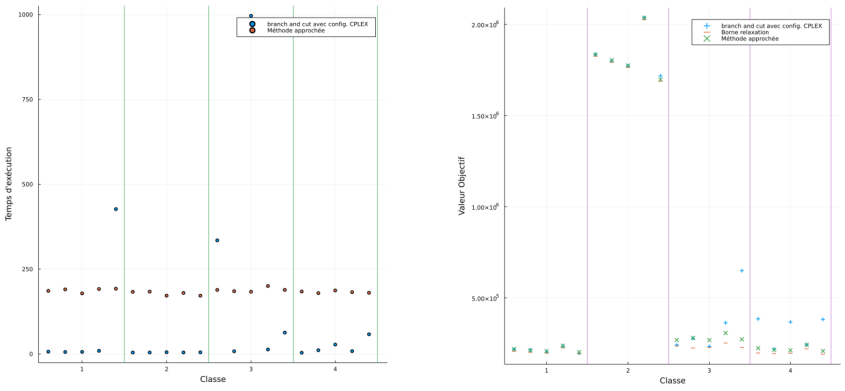
D'après les résultats ci-dessus, la méthode exacte est toujours meilleure que la méthode approchée sur toutes les classes d'instance. La méthode exacte est également capable de renvoyer une solution ayant une gap relative proche de 0 en moins de 5 mins. Cependant,

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

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.

3.1.2 Résultat sur des petites instances de type A (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.



(a) Temps d'exécution (b) Valeurs objectifs

FIGURE 6 – Graphe en fonction des classes des instances de type A de taille 50

D'après les résultats ci-dessus, 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 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.

3.1.3 Résultat pour toutes les instances de taille jusqu'à 100

Pour les instances de taille 100, la méthode exacte n'est pas efficace en raison de la com- plexité 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.

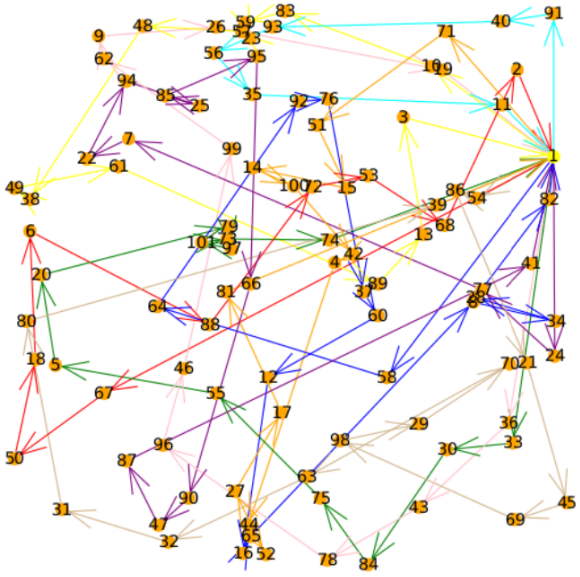
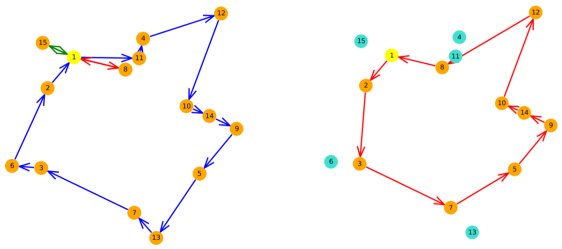


FIGURE 7 – Exemple de tournée réalisable obtenue par l'algorithmes génétique pour une instance de taille 100

3.2 Comparaison entre 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.



(a) la méthode exacte

(b) la méthode heuristique en deux phases

FIGURE 8 – Tournée réalisable de la période 2 pour l'instance A_014_ABS36_15_1 obtenue

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 Annexess

Le lien de github sur le projet : Projet_Production_Routing
Le lien de soutenance pour ce projet : Soutenance_MAOA_Muyang_SHI

Utilisation des fichiers :

La documentation sur l'utilisation des fonctions sont dans les fichiers .jl. Mais, en particulier le fichier main.jl contient les principaux fonctions qui permettent d'obtenir les solutions qui sont placées dans le fichier Results ou Example_results selon les fonctions que vous utilisez.

Pour pouvoir lancer les fonctions, veuillez installer Cplex.

Pour obtenir tous les résultats des instances de type A pour 6 véhicules :

- Lancez Julia sur un terminal ayant le path sur le production_routing_problem
- Entrez sur le terminal : include("main.jl")
- Puis entrez : Provide_full_results("A", 6)

Les résultats (les logs et les graphes de chemins pour les VRP) seront placés dans le dossier *Example_results*.

Pour obtenir tous les résultats par classe des instances de type A pour 6 véhicules :

- Entrez : Provide_class_results("A", 6)

Les résultats (les logs et les graphes de chemins pour les VRP) seront placés dans le dossier *Results*.