Lot Sizing Problem

Boualem LAMRAOUI, COULIBALY Seydou

November 2017

Introduction

On s'intéresse à un problème de production de type lot-sizing mono-produit (dans une 1^{iere} partie) et à plusieurs produits (dans la seconde partie). L'idée est de proposer un modèle de programmation linéaire en nombre entiers mixte puis d'analyser les solutions possible avec une méthode de résolution exacte comme l'algorithme de Branch and Bound.

Part I

Problème mono-produit de lot-sizing ULS & CLS

Chapter 1

Problème de lot-sizing sans capacité

1.1 Les données du problème

- t : Nombre de périodes
- M : Constante symblique fixée à 100
- D_t : Demande du produit à la période t
- P_t : Coût de production à la période t
- H_t : Coût de stockage à la période t
- F_t : Coût de demarrage de production à la période t

1.2 Les variables du problème

- x_t : Quantités produites à la période t
- $y_t = 1$ si on produit à la période t et 0 sinon
- s_t : Quantités de stocks à la période t

1.3 Modèlisation linéaire

$$\begin{array}{ll} \text{Min} & \sum_{k=1}^{t}(P_k * x_k + F_k * y_k + H_k * s_k) \\ s/c & \end{array}$$

$$
\begin{array}{rcl}
x_i + s_{i-1} & = D_i + s_i & \forall i = 2..t \\
x_1 & = D_1 + s_1 & \\
x_i & \leq M * y_i & \forall i = 1..t \\
x_i & \geq 0 & \forall i = 1..t \\
y_i & \in \{0,1\} & \forall i = 1..t \\
s_i & \geq 0 & \forall i = 1..t
\end{array}
$$

1.4 Etude de cas

Soit l'instance pour le problème ULS (uncapacited lot-sizing problem) :

t	1	2	3	4	5
D_t	3	5	6	3	8
P_t	2	4	6	8	10
H_t	3	2	3	2	0
F_t	10	8	6	4	2

Table 1.1: Une instance de ULS ($T = 5$)

1.4.1 Le modèle linéaire de l'instance :

$$
\begin{array}{rc}
Min & 2x_1 + 4x_2 + 6x_3 + 8x_4 + 10x_5 + \\
& 3s_1 + 2s_2 + 3s_3 + 2s_4 + \\
& 10y_1 + 8y_2 + 6y_3 + 4y_4 + 2y_5
\end{array}
$$

s/c

$$
\begin{array}{rcl}
x_2 + s_1 - s_2 & = 5 & \\
x_3 + s_2 - s_3 & = 6 & \\
x_4 + s_3 - s_4 & = 3 & \\
x_5 + s_4 - s_5 & = 8 & \\
x_1 - s_1 & = 3 & \\
x_1 - 100 * y_1 & \leq 0 & \\
x_2 - 100 * y_2 & \leq 0 & \\
x_3 - 100 * y_3 & \leq 0 & \\
x_4 - 100 * y_4 & \leq 0 & \\
x_5 - 100 * y_5 & \leq 0 & \\
x_i \geq 0 & \forall i = 1..5 & \\
y_i \in \{0,1\} & \forall i = 1..5 & \\
s_i \geq 0 & \forall i = 1..5 &
\end{array}
$$

1.4.2 Implémentation du modèle sous JUMP

Le solveur renvoie une solution optimale entière en 2.125715 seconds, 1.32 M allocations: 69.587 MiB, 2.02 % gc time sur un PC core I5 dont :

- $Z^* = 188$
- $x^* = [3, 11, 0, 11, 0]$
- $y^* = [1, 1, 0, 1, 0]$
- $s^* = [0, 6, 0, 8, 0]$

On aperçoit ci-dessous l'illustration de la solution optimale à la manière de la première FIGURE présentée dans le sujet de TP.

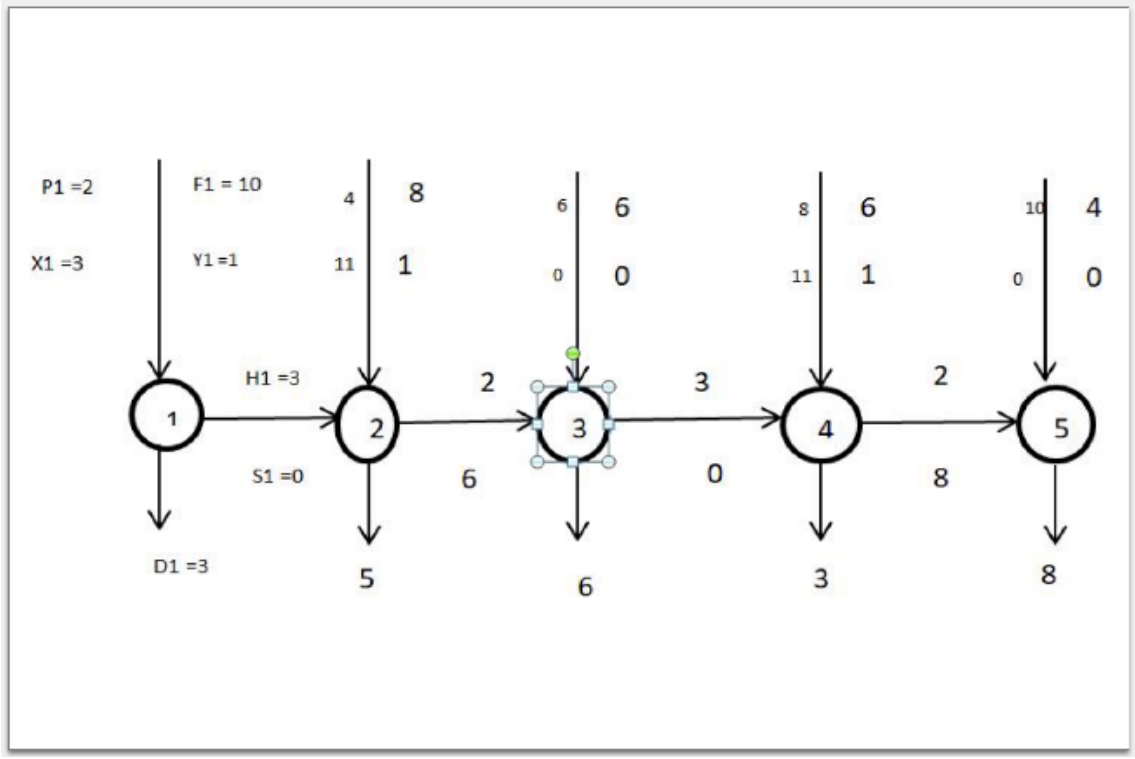


Figure 1.1: Exemple d'un ULS sur 5 périodes

1.5 Branch & Bound

1.5.1 Bornes primale et duale

Borne Primale (BP)

Dans cette proposition de borne primale, on se fixe de produire pendant toutes les périodes du cycle de la production. Ansi tous les Y_t vallent 1, on produit exactement la demande de la période t, mais on ne stocke aucune quantité.

C'est une solution du problème verifiant toutes les contraintes et la solution correspondante est $z_{primale} = \sum_{k=1}^{t} (P_k * x_k + F_k * y_k)$ avec $x_k = D_k$. Dans l'exemple ci-dessus $z_{primale}$ vaut 196

Borne Duale (BD)

La borne duale est le resultat de la rélaxation linéaire du problème sur la variable Y. On utilise le solveur GLPK pour retrouver cette borne. Dans notre exemple $z_{duale} = 167.34$

1.5.2 Algorithme

STEP 0 EVALUER :
Résoudre le probléme relacher de probléme linéaire LP

STEP 1 SEPARER :
Choisir la première variable Y_t disponible, non encore affectée comme variable de branchement et la fixée en imposant des contraintes : $Y_t = 0$ (Fils gauche) et $Y_t = 1$ (Fils droit)

STEP 2 BRANCHER :
Continuer en choisissant le fils gauche, l'explorer en entier avant de proceder au branchement sur le noeud du fils droit.

Un noeud est sondé Quand :

- Une solution entière est obtenue
- Le problème LP associé est non réalisable
- La solution optimale du LP est plus grande que la meilleure solution entière connue depuis l'arborescence initiale

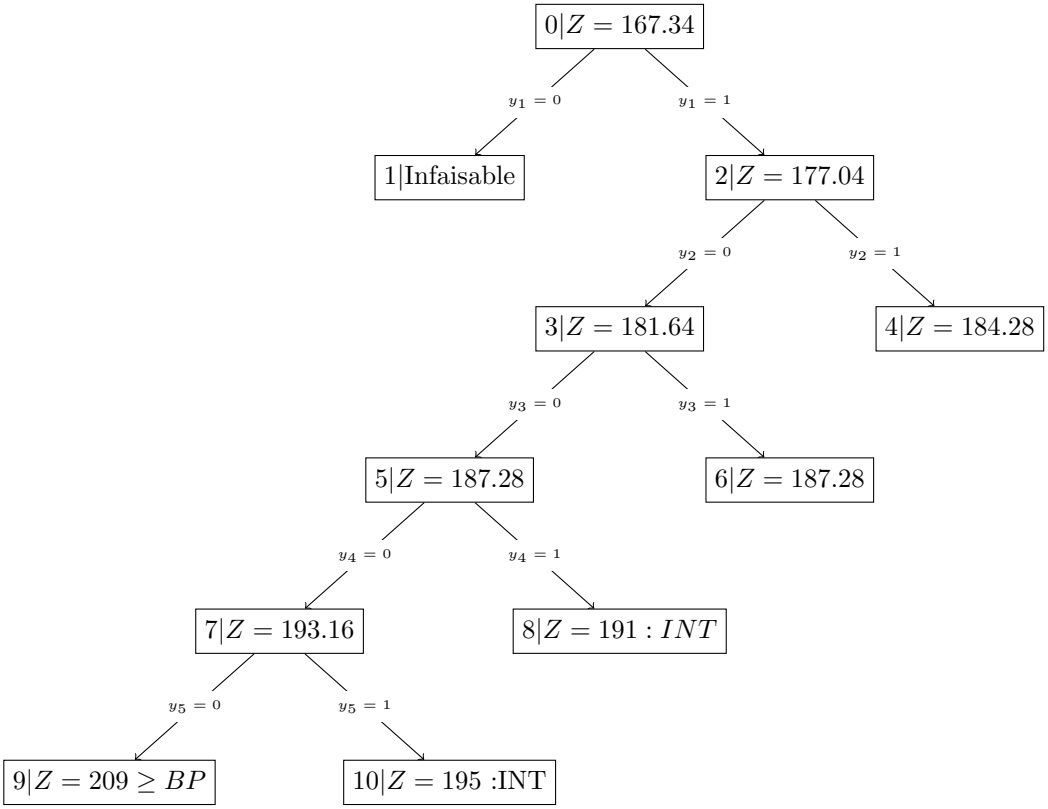
Le backtracking est effectué lorsque le noeud est sondé

Problème resolue s'il n'y a plus de noeud pendant(c'est à dire que tous les noeuds sont sondés)

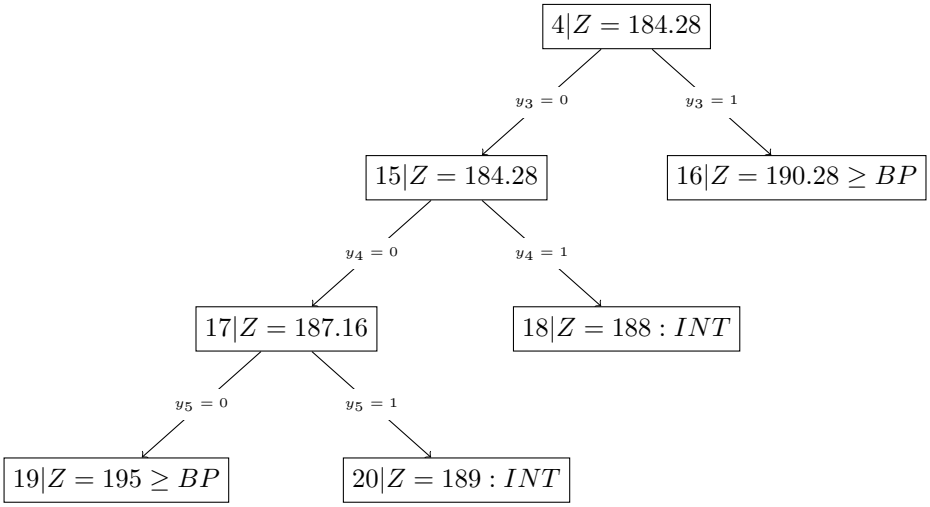
1.5.3 Arborescence de l'instance du cas d'etude

L'arborescence construite par l'algorithme de branch and bound est illustré dans la figure ci_dessous. On observe 20 noeuds contruites et l'optimum s'y trouve sur le noeud 18.

Remarque Notre algorithme de branch and bound explore en plus profonde descente, fils gauche puis fils droit : le parcours en profondeur


0|Z = 167.34
y1 = 0
y1 = 1
1|Infaisable
2|Z = 177.04
y2 = 0
y2 = 1
3|Z = 181.64
4|Z = 184.28
y3 = 0
y3 = 1
5|Z = 187.28
6|Z = 187.28
y4 = 0
y4 = 1
7|Z = 193.16
8|Z = 191 : INT
y5 = 0
y5 = 1
9|Z = 209 ≥ BP
10|Z = 195 :INT

Noeud4


4|Z = 184.28
y3 = 0
y3 = 1
15|Z = 184.28
16|Z = 190.28 ≥ BP
y4 = 0
y4 = 1
17|Z = 187.16
18|Z = 188 : INT
y5 = 0
y5 = 1
19|Z = 195 ≥ BP
20|Z = 189 : INT

Noeud6

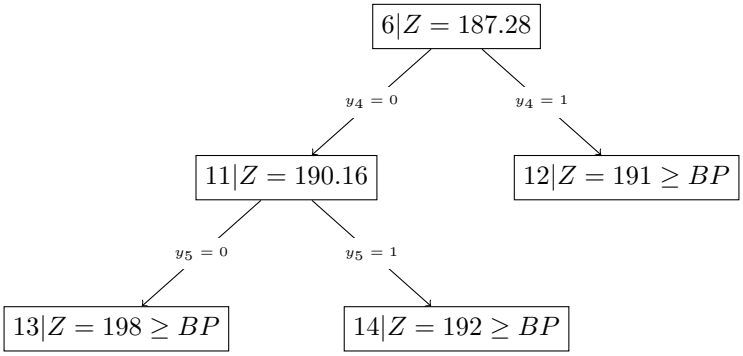


Figure 1.2: Arborescence Premier Branch & Bound

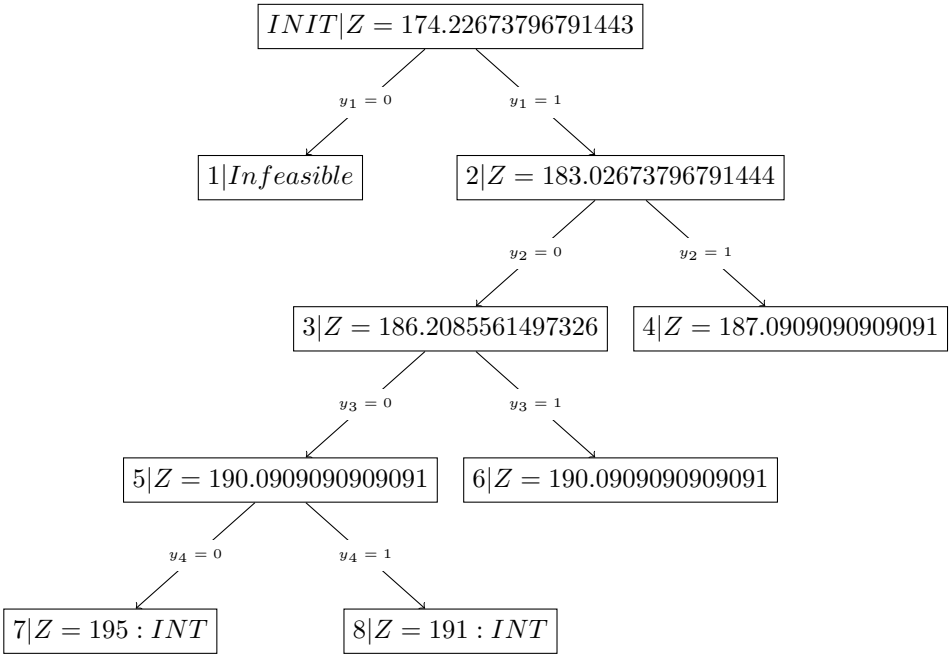
Chapter 2

Problème de lot-sizing avec capacité

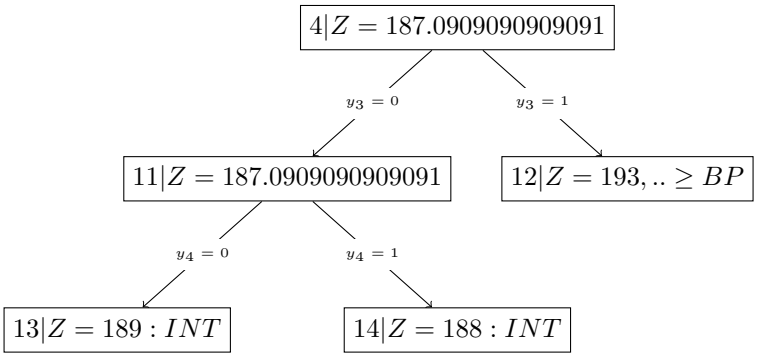
2.1 Première tentative d'amélioration de la résolution

La constante symbolique M fixée à 100 est une borne supérieure sur les quantités de productions x_t. En observant les valeurs des variables à l'optimum, on peut fixer des bornes supérieures encore plus bonne que M. On peut ainsi proposer comme valeur pour M, la somme des demandes du problèmes, soit la valeur 25 (sur l'exemple). En poussant plus loin l'analyse, on s'aperçoit que chacune des quantités de production x_t au temps t est borné par la somme des demandes de t au nombre de périodes du problème.

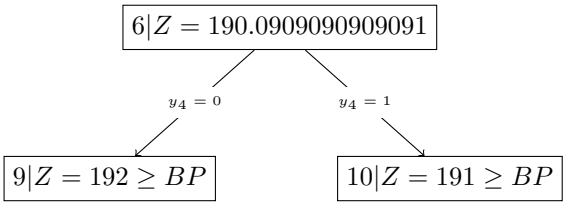
Ces nouvelles valeurs de M contribuent à imposer des contraintes de capacités sur les quantités à produire. L'espace des solutions se retrouve plus resséré. L'execution de l'algorithme Branch and Bound sur le problème avec les nouvelles valeurs de M est beaucoup plus meilleure que la précedente analyse, on se retrouve avec 14 noeuds construits au lieu de 20. La taille de l'arborescence diminue d'une difference de 6 noeuds (le gain) et l'optimum est visible au noeud 14 et vaut 188.



Noeud4



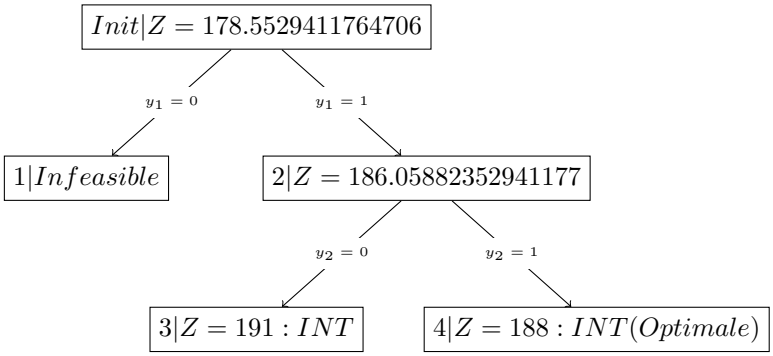
Noeud6



2.2 Seconde tentative d'amélioration de la résolution

L'inégalité valide (contrainte 22) de l'article de L. Wolsey "Progress with single-item lot-sizing", European Journal of Operational Research 86 (1995) 395-401 est la suivante : $S_{t-1} \geq D_t * (1 - Y_t)$. Ces inégalités valides contribuent à ressérer la valeur optimale du problème relaché et ainsi permettre à améliorer la qualité de la résolution. La solution optimale du Branch and Bound ne change pas des précedents, elle vaut 188. A l'aide de l'inégalité valide, la qualité est nettement visible, en seulement 4 noeuds contruit au total, on retrouve la solution optimale. Pour rappelle la première implémentation construisait 20 noeuds pour avoir l'optimum. Le gain est enorme : 16 noeud de gains par rapport à la première implementation et 10 noeud de gains sur la deuxième implementations.

Arborescence



2.3 Variante Coût de WagnerWhitin

Lorsque ULS repond aux coûts de WagnerWhitin, la consequence est qu'il est toujours optimale de produire le plus tard possible vu que $P_t + H_t \geq P_{t+1}$. Cela ce traduit par le fait qu'il est préferable de produire lorsque les coûts de productions sont minimes au lieu de produire à des périodes précedentes avec des coûts de stockages enormes.

2.4 Troisième tentative d'amélioration de la résolution

La rélaxation linéaire du modèle fournit dans cette section, nous retourne une solution optimale entière vallant 188.La première itération suffit pour avoir une arborescence de l'algorithme Branch and Bound stable.

Arborescence du Branch & Bound $\boxed{INIT|Z = 188 : Int(optimale)}$

Part II

Problème de production multi-produits de type lot-sizing

Chapter 3

problème de lot-sizing à plusieurs references avec capacité

Soit la contrainte : $\sum_{i=1}^{N}(v_t^i * x_t^i + \Phi_t^i * y_t^i) \leq C_t$ avec t = 1...T

Cette contrainte délimite le nombre de ressources à utiliser suivant une période donnée. Le premier terme correspond à la ressource totale utilisée par les N produits dans une période t et doit rester inférieur à la capacité totale disponible (terme 2).

3.1 Les données ou paramètres du problème

- T : Nombre de périodes
- D_t^i : Demande du produit i à la période t
- P_t^i : Coût de production du produit i à la période t
- H_t^i : Coût unitaire de stockage du produit i à la période t
- F_t^i : Coût fixe de démarrage de la production de i à la période t
- v_t^i : Besoin en ressources pour le produit i à la période t
- C_t : capacité disponible à la période t
- M_t^i : Capacité de production du produit i à la période t
- ϕ_t^i : Besoin fixe en ressources pour le produit i à la période t, c'est le setup.

- B_t^i : Le coût unitaire de rupture sur la demande pour la réféerence ou produit i à la période t.

3.2 Les variables du problème

- x_t^i : Quantités produites par i à la période t
- $y_t^i = 1$ si on produit i à la période t et 0 sinon
- s_t^i : Quantités de stocks du produit i à la période t

3.3 Modèlisation linéaire

$$\text{Min} \quad \sum_{i=1}^{N} \sum_{t=1}^{T} (P_t^i * x_t^i + F_t^i * y_t^i + H_t^i * s_t^i)$$

s/c

$$\begin{array}{cccc}
x_t^i + s_{t-1}^i & = D_t^i + s_t^i & i = 1..N, t = 2..T \\
x_1^i & = D_1^i + s_1^i & i = 1..N \\
x_t^i & \leq M * y_t^i & i = 1..N, t = 1..T \\
\sum_{i=1}^{N} (v_t^i * x_t^i + \Phi_t^i * y_t^i) & \leq C_t & t = 1..T \\
x_t^i & \geq 0 & i = 1..N, t = 1..T \\
y_t^i & \in \{0, 1\} & i = 1..N, t = 1..T \\
s_t^i & \geq 0 & i = 1..N, t = 1..T
\end{array}$$

3.4 Notions de la classe de problème de lot-sizing

3.4.1 Backlog

On parle de backlog, lorsque des retards sur la demande des clients sont autorisés. On accepte que ces demandes soient satisfaites à des périodes future. Ce retard est pénalisé puisqu'il peut avoir un impact négatif sur la satisfaction des clients.

3.4.2 Small bucket

C'est la classe des problèmes sur lesquels l'horizon de planification est discret avec des périodes de temps courtes. Le small bucket est utilisé sur pour résoudre des problèmes opérationnels à court terme.

3.4.3 Big bucket

C'est la classe des problèmes sur lesquels l'horizon de planification est discret avec de longues périodes de temps. Le big bucket est employé pour résoudre des problèmes tactiques à moyen terme.

3.5 Mis à jour du modèle linéaire avec Backlog

Pour ajouter des backlog, on intègre une nouvelle variable $r_t^i \geq 0$ réprésentant la rupture sur la demande de la période t qui sera livrée en retard au temps t+1, t+2, ...,T.

$$\begin{array}{rl} \text{Min} & \sum_{i=1}^{N}\sum_{t=1}^{T}(P_t^i * x_t^i + F_t^i * y_t^i + H_t^i * s_t^i + B_t^i * r_t^i) \\ s/c & \end{array}$$

$$\begin{array}{ccc} x_t^i + s_{t-1}^i + r_t^i - r_{t-1}^i & = D_t^i + s_t^i & i = 1..N, t = 2..T \\ x_1^i + r_1^i & = D_1^i + s_1^i & i = 1..N \\ x_t^i & \leq M * y_t^i & i = 1..N, t = 1..T \\ \sum_{i=1}^{N}(v_t^i * x_t^i + \Phi_t^i * y_t^i) & \leq C_t & t = 1..T \\ r_t^i & \leq D_t^i & i = 1..N, t = 1..T \\ x_t^i & \geq 0 & i = 1..N, t = 1..T \\ y_t^i & \in \{0,1\} & i = 1..N, t = 1..T \\ s_t^i & \geq 0 & i = 1..N, t = 1..T \end{array}$$

3.6 Résolution du problème

3.6.1 Utilisation de la ressource NEOS

La résolution sous NEOS en utilisant le solver CBC nous fournit une solution entière. Le fichier de log de sortie nous montre l'utilisation de plusieurs coupes, la résolution des mini-branch and bound.

- Objective value: 7350
- Enumerated nodes: 777
- Total iterations: 35962
- Time (CPU seconds): 11.76
- Time (Wallclock seconds): 51.58

On rappelle que la relaxation linéaire est 2748.35.

3.6.2 Mis en oeuvre informatique (LP)

Le paramètre M_t^i impose une contrainte de capacité sur la quantité de production d'un produit à une période donnée. Ce paramètre se differentie de C_t, qui joue un peu le même rôle de capacités, mais en agissant sur le nombre de ressources à utiliser. Dans la résolution du problème, plus la valeur de M est grand, c'est à dire plus on relache la contrainte de capacité, plus la valeur de solution optimale est meilleure. On a teste deux cas de figures :

M_t^i est la somme des demandes futures :

C'est la valeur la plus evidente à prendre pour M sur un problème où toutes les conditions sont reunis au tour des paramètres Phi_t^i, P_t^i, v_t^i . La valeur de Z observée à l'optimum est 4309.59 dont les variables :

variable x^*

$Produits/Périodes$	$T1$	$T2$	$T3$	$T4$	$T5$	$T6$	$T7$	$T8$
$P1$	0	70	50	100	20	80	0	0
$P2$	20	70	60	0	0	20	20	0
$P3$	40	50	0	100	40	80	90	160
$P4$	0	100	100	150	160	90	100	0
$P5$	100	0	50	0	0	0	0	0
$P6$	70	40	40	40	160	0	0	0
$P7$	0	20	50	10	20	100	0	0
$P8$	40	0	0	0	0	30	0	0

variable y^*

$Produits/Périodes$	$T1$	$T2$	$T3$	$T4$	$T5$	$T6$	$T7$	$T8$
$P1$	0.0	0.17	0.14	0.33	0.1	0.44	0.0	-0.0
$P2$	0.08	0.32	0.33	0.0	-0.0	0.22	0.22	-0.0
$P3$	0.07	0.1	0.0	0.21	0.11	0.24	0.36	1.0
$P4$	0.0	0.12	0.14	0.25	0.36	0.31	0.5	-0.0
$P5$	0.62	-0.0	0.45	0.0	-0.0	-0.0	-0.0	-0.0
$P6$	0.18	0.12	0.14	0.16	0.76	-0.0	-0.0	-0.0
$P7$	0.0	0.08	0.23	0.06	0.12	0.71	-0.0	-0.0
$P8$	0.4	-0.0	0.0	0.0	-0.0	0.5	-0.0	-0.0

variable s^*

$Produits/Périodes$	$T1$	$T2$	$T3$	$T4$	$T5$	$T6$	$T7$	$T8$
$P1$	0	0	0	0	0	0	0	0
$P2$	0	30	40	30	0	20	0	0
$P3$	0	0	0	0	0	0	0	0
$P4$	0	0	0	0	0	0	0	0
$P5$	50	50	80	40	30	20	0	0
$P6$	0	0	0	0	60	40	0	0
$P7$	0	0	0	0	0	40	0	0
$P8$	30	10	10	10	0	20	0	0

variable r^*

$Produits/Periodes$	$T1$	$T2$	$T3$	$T4$	$T5$	$T6$	$T7$	$T8$
$P1$	0	0	0	0	0	0	0	100
$P2$	0	0	0	0	0	0	0	50
$P3$	0	0	0	0	0	0	0	0
$P4$	0	0	0	0	0	0	0	100
$P5$	0	0	0	0	0	0	0	10
$P6$	0	0	0	0	0	0	0	50
$P7$	0	0	0	0	0	0	0	40
$P8$	0	0	0	0	0	0	0	30

Cas où $M_t^i = C_t$ indépendamment du produit i :

on prend le cas de figure ou $M_t^i = C_t$ et on fait notre simplexe pour le problème relâcher du problème MCLS avec le solveur GLPK LP et on obtient les résultats suivants: $Z^* = 2240{,}53$

variable x^*

Produits/Périodes	$T1$	$T2$	$T3$	$T4$	$T5$	$T6$	$T7$	$T8$
$P1$	0	70	50	100	20	80	0	100
$P2$	20	40	50	10	30	0	40	50
$P3$	40	50	0	100	40	80	90	160
$P4$	0	100	100	150	160	90	100	100
$P5$	50	0	20	40	10	10	20	10
$P6$	70	50	70	0	100	20	40	50
$P7$	0	20	60	0	20	60	40	30
$P8$	10	20	0	0	10	10	20	0

variable y^*

Produits/Périodes	$T1$	$T2$	$T3$	$T4$	$T5$	$T6$	$T7$	$T8$
$P1$	0.0	0.2	0.14	0.25	0.05	0.2	0.0	0.2
$P2$	0.06	0.11	0.14	0.02	0.08	0.0	0.1	0.1
$P3$	0.11	0.14	0.0	0.25	0.1	0.2	0.22	0.32
$P4$	0.0	0.29	0.29	0.38	0.4	0.22	0.25	0.2
$P5$	0.14	0.0	0.06	0.1	0.02	0.02	0.05	0.02
$P6$	0.2	0.14	0.2	0.0	0.25	0.05	0.1	0.1
$P7$	0.0	0.06	0.17	-0.0	0.05	0.15	0.1	0.06
$P8$	0.03	0.06	0.0	0.0	0.02	0.02	0.05	-0.0

La variable s vaut 0 pour tout t $\in$ [0,T]. La variable r_t^i vaut egalement 0 pour tout t et i sauf : $r_8^7 = 10$, $r_8^8 = 30$

3.6.3 Mis en oeuvre informatique (GLPK MIP)

cette résolution a été effectué à l'aide du solveur GLPK MIP pour un sous problème de taille $(5 * 5)$ vu que la résolution faite sur le problème MLCS initial est trop lente à être réaliser . on obtient une solution entière $Z^* = 2380$ en un temps d'exécution de 11.23 secondes .

variable x^*

Produits/Périodes	$T1$	$T2$	$T3$	$T4$	$T5$
$P1$	0.0	0.0	220.0	0.0	0.0
$P2$	120.0	0.0	0.0	0.0	0.0
$P3$	90.0	0.0	100.0	0.0	0.0
$P4$	0.0	200.0	0.0	310.0	0.0
$P5$	110.0	0.0	0.0	0.0	0.0

variable y^*

$Produits/Périodes$	$T1$	$T2$	$T3$	$T4$	$T5$
$P1$	0	0	1	0	0
$P2$	1	0	0	0	0
$P3$	1	0	1	0	0
$P4$	0	1	0	1	0
$P5$	1	0	0	0	0

variable s^*

$Produits/Périodes$	$T1$	$T2$	$T3$	$T4$	$T5$
$P1$	0	0	100	0	0
$P2$	100	60	10	0	0
$P3$	50	0	100	0	0
$P4$	0	100	0	160	0
$P5$	60	60	40	0	0

variable r^*

$Produits/Périodes$	$T1$	$T2$	$T3$	$T4$	$T5$
$P1$	0	70	0	0	20
$P2$	0	0	0	0	30
$P3$	0	0	0	0	40
$P4$	0	0	0	0	0
$P5$	0	0	0	0	10

3.6.4 Mis en oeuvre informatique (CPLEX)

Cette résolution a été effectué à l'aide du solveur CPLEX. On obtient en 1.533 secondes, l'optimum entièr sur l'instance pp08a.dat. La valeur $Z^* = 6950$ dont les coupes :

- Clique cuts applied: 5
- Cover cuts applied: 3
- Implied bound cuts applied: 5
- Flow cuts applied: 74
- Mixed integer rounding cuts applied: 137
- Flow path cuts applied: 24
- Lift and project cuts applied: 2
- Gomory fractional cuts applied: 20

Remarque Dans ce qui suit, on prend $M_t^i = C_t$ indépendamment du produit.

A l'optimum, la valeur des variables :

variable x^*

$Produits/Périodes$	$T1$	$T2$	$T3$	$T4$	$T5$	$T6$	$T7$	$T8$
$P1$	0.0	0.0	220.0	0.0	0.0	100.0	0.0	0.0
$P2$	60.0	0.0	130.0	0.0	0.0	0.0	0.0	0.0
$P3$	90.0	0.0	0.0	100.0	-0.0	180.0	-0.0	190.0
$P4$	0.0	200.0	0.0	0.0	400.0	0.0	0.0	200.0
$P5$	70.0	0.0	0.0	80.0	0.0	0.0	0.0	0.0
$P6$	130.0	0.0	0.0	220.0	0.0	0.0	0.0	0.0
$P7$	0.0	80.0	0.0	0.0	0.0	120.0	0.0	0.0
$P8$	0.0	70.0	0.0	0.0	0.0	0.0	0.0	0.0

variable y^*

$Produits/Périodes$	$T1$	$T2$	$T3$	$T4$	$T5$	$T6$	$T7$	$T8$
$P1$	0	0	1	0	0	1	0	0
$P2$	1	0	1	0	0	0	0	0
$P3$	1	0	0	1	0	1	0	1
$P4$	0	1	0	0	1	0	0	0
$P5$	1	0	0	1	0	0	0	0
$P6$	1	0	0	1	0	0	0	0
$P7$	0	1	0	0	0	0	0	0
$P8$	0	1	0	0	0	0	0	0

variable s^*

$Produits/Périodes$	$T1$	$T2$	$T3$	$T4$	$T5$	$T6$	$T7$	$T8$
$P1$	0	0	100	0	0	0	0	0
$P2$	40	0	80	70	40	40	0	0
$P3$	50	0	0	0	0	59	0	0
$P4$	0	100	0	0	90	0	0	0
$P5$	20	20	0	40	30	20	0	0
$P6$	59	19	0	160	60	40	0	0
$P7$	0	60	10	0	0	40	0	0
$P8$	0	40	40	40	30	20	0	0

variable r^*

$Produits/Périodes$	$T1$	$T2$	$T3$	$T4$	$T5$	$T6$	$T7$	$T8$
$P1$	0	70	0	0	19	0	0	100
$P2$	0	0	0	0	0	0	0	50
$P3$	0	0	0	0	40	0	30	0
$P4$	0	0	0	150	0	0	99	0
$P5$	0	0	0	0	0	0	0	10
$P6$	0	0	20	0	0	0	0	50
$P7$	0	0	0	0	19	0	0	40
$P8$	10	0	0	0	0	0	0	30

Commentaires métiers :

En utilisant les différents solveurs en obtient des résultats sous formes de tableau de deux dimensions

pour le tableau de x^* il représente la quantité de produit "i" à produire dans la période "t" prenons par exemples le cas de produit 1 , on constate que la production est faite uniquement à la période "T3" et qu'elle n'est pas faite sur toutes les autres périodes et cela est du à la forte demande sur le produit "i" à la période "t".

le tableau y^* à son tour est un tableau binaire $\{1,0\}$ $\begin{cases} 1 & = & \textit{si on décide de produire le produit "i" à l'instant "t"} \\ 0 & = & \textit{si non} \end{cases}$

le tableau s^* lui il nous donne la quantité de produit "i" non épuise à la période "t" , il est du au fait que la quantité de produit "i" produite à l'instant "t" x_t^i dépasse la demande offerte sur ce dernier à cette période .

en observant les tableaux x^*, y^*,s^* et r^* on remarque qu'ils sont dépendants l'un de l'autre .

3.6.5 Branch and bound

L'algorithme de branch & bound réalisé (MCLS) se différentier de la précédente (ULS et CLS) en séparant les noeuds sur les $y_t^i = 0$ et $y_t^i = 1$. On effectue une plus profonde descente et l'ordre de sélection de branchement des variables est la suivante :

```
For t in Periode
    For i in Produits
        Brancher sur $y_t^i$
    end
end
```

On fixe ainsi pour une période donnée toutes les variables avant de passer à la période suivante. La borne duale est la relaxation du problème initiale et la borne primale reste la résolution du problème en imposant pour chaque période, une production de la quantité x_t^i.

La résolution de l'instance pp08a.dat étant très longue, on se décide de le réduire en travaillant sur de plus petites instances. Dans cette idée de réduction d'instance, la résolution de la sous-matrice 6*6 de pp08.dat nous a pris 3 heures de temps avant de l'interrompre. Dans le tableau ci-dessous figure les résultats de quelques exécutions sur les sous matrices de l'instance pp08a.dat.

$N * T$	$Time(s)$	Nb_noeuds	Z^*
$2 * 2$	0.49	22	280
$3 * 3$	0.5	136	640
$4 * 4$	3.53	5254	1610
$5 * 5$	54.78	62678	2380

3.6.6 Indication utilisation de la solution logicielle

Le projet est structuré de la sorte :

- doc : ce rapport en format PDF
- src : Les fichiers sources de notre application accompagné d'un README reprenant cette section du rapport
- dat : Les fichiers d'instances de notre application

Remarque En sortie de notre application, des affichages sont effectués sur le terminale Juila. Le dossier res/ n'a donc pas d'intérêt à être présenter dans ce rapport. La description des fichiers d'instances est expliquée dans le fichier FORMAT-INSTANCE situé dans le dossier dat.

Pour exécuter l'application il suffit de se positionner dans le dossier src/ et de lancer la commande include("main.jl"). Ceci compile et interprète toutes nos fonctions, il suffira alors de d'exécuter l'instance et le type de problème désire par : solveLotSizing(typeProbleme,nomInstance,Parametre)
Où les valeurs possibles de parametre (type : String) sont :

1. "GLPK" : solveur GLPK en résolution MIPS
2. "CPLEX" : Solveur CPLEX en résolution MIPS
3. "BB": Notre Branch & Bound implémenté

Conclusion

La mise en oeuvre et la résolution du problème ULS est facile, mais la qualité d'une bonne méthode de résolution demande cependant des analyses plus approfondis sur les différentes contraintes et domaine de solutions du problème. Le choix des valeurs des données est très important notamment sur les quantités de productions qui peut vite faire passer le problème de ULS à CLS en imposant des contraintes de capacités sur certaines variables de décisions. L'importance des inégalités valides est sans conteste, on a puis réduire considérablement le nombre d'arborescence construite dans une méthode de résolution de branch and bound.

Le problème se complexifie très vite lorsque le nombre de produits augmente. Pour le résoudre, il est donc préférable de se limiter à quelques types ou classes d'instances tels que des instances avec des périodes courtes. Ceci ouvre le choix sur une diversité de sous-problème utilisant des notions comme des Backlog, Small bucket et Big bucket. L'algorithme de Branch & Bound implémenté requiert un temps assez consequent pour s'executer sur une instance de taille plus grande, plus de 3h pour 6 produits en 6 périodes alors que des solveurs comme CPLEX le resolvent 8 produits en 8 périodes en quelques secondes. En remarque GLPK traine lui aussi sur l'instance pp08a.dat. La perspective de cette dernière partie serait de faire mieux comme CPLEX en travaillant sur des coupes additionnelles et en essayant de réduire le problème pour le résoudre efficacement.