MÉTHODES DE LA RECHERCHE OPÉRATIONNELLE

Miniprojet

Dorian Campbell

Marion Cavaglia

Manuela Rodriguez

Groupe B5

20 mai 2019

Table des matières

Introduction

La méthode de la recherche opérationnelle a pour but de chercher les solutions optimales afin de réaliser des travaux aux plus bas coûts, au temps minimal, entre autres. Ce projet est l'occasion pour nous de mobiliser nos connaissances en la matière au travers de quatre exercices d'optimisation. Ce projet est par ailleurs l'occasion de résoudre des problèmes d'optimisation dynamique ou linéaire de façon informatique. On utilisera le logiciel MATLAB. Les scripts utilisés, ainsi que les résultats numériques et les graphes obtenus, seront joints en annexes.

I. Un problème de chargement de type sac à dos

1. Présentation du problème

Ce problème de chargement a pour but de trouver la méthode optimale afin de charger un cargo de 995 tonnes, puisque la méthode heuristique visant à considérer l'attractivité respective des marchandises n'est pas la meilleure. Nous cherchons donc à appréhender le problème d'une manière différente.

2. Formalisation du problème

Le problème peut s'écrire sous la forme :

$$Max \sum_{i=0}^{I} c_i x_i$$

$$(K) \qquad \left| \begin{array}{l} \sum_{i=0}^{4} a_i x_i = b \\ x_i \in \mathbb{N} \quad \forall i = 0,1,2,3,4 \end{array} \right.$$

De même, il peut être formalisé de manière équivalente sous la forme :

$$Max \sum_{i=1}^{I} c_i x_i$$

$$(D) \qquad \left| \begin{array}{l} y_0 \in \{0,1,\dots,b\} \\ y_4 = b \\ \left. \begin{array}{c} y_i = y_{i-1} + a_i x_i \\ x_i \in \mathbb{N} \end{array} \right\} \forall i = 0,1,2,3,4 \end{array} \right.$$

Avec I=4, b=995 et :

i	0	1	2	3	4
a_i	1	125	51	18	13
c_i	0	126	51	16	12

Nous remarquons que ces deux problèmes sont équivalents. En effet, le système écrit sous la forme (K) traduit le chargement du cargo en une seule fois d'où la présence du signe Σ. A l'inverse, l'écriture de (D) montre qu'on charge le cargo par type de marchandises. Pour appliquer l'algorithme développé par le système (D), il faut connaître les états précédents car les y_i représentent l'état de chargement du cargo en tonne à l'instant i.

Par ailleurs, l'équation de Hamilton-Jacobi-Bellman du problème (D) s'écrit bien sous la forme :

$$(1) \qquad V_i(y_i) = Max_{\substack{x_i \in \mathbb{N} \\ x_i \leq {}^{y_i}/_{a_i}}} [V_{i-1}(y_i - a_i x_i) + c_i x_i]$$

Initialisée par $V_0(y_0) = 0 \quad \forall y_0 = 0, 1, \dots, b$

Les V_i représentent les bénéfices que nous cherchons à maximiser. Ensuite compte tenu de l'écriture du problème (D), l'équation $V_{i-1}(y_i - a_i x_i) + c_i x_i$ est cohérente puisque V_{i-1} dépend de $y_{i-1} = y_i - a_i x_i$ et ensuite pour avoir l'état en i il faut ajouter $c_i x_i$ pour avoir les bénéfices gagnés en i. Il en est de même pour les x_i car ils appartiennent bien à l'ensemble des entiers naturels comme cela est défini au problème (D). De plus, on remarque que $x_i \leq y_i/a_i$ est cohérent car $y_i = y_{i-1} + a_i x_i$ et $y_{i-1} \geq 0$ donc forcément $y_i \geq a_i x_i$. L'initialisation se justifie car au début le cargo est vide donc $V_0(y_0) = 0$ et $y_0 \in \{0, 1, \dots, b\}$ d'après l'écriture de (D).

3. Résolution

a. Principe de résolution

On crée une matrice M d'entiers de nombre de lignes égal au nombre d'objets et de nombre de colonnes égal à la capacité + 1. Chaque case Mi,j de la matrice représente le bénéfice maximal possible pour les i premiers objets avec un poids j. On initialise avec le premier objet ($V_0(y_0)=0$) et on réitère pour tous les autres.

Une fois la matrice remplie, le bénéfice maximal se trouve dans la case en bas à droite de la matrice. À partir de cette case, il faut remonter dans la matrice pour trouver le chemin de chargement optimal. On se déplace d'abord horizontalement vers la gauche tant que le bénéfice ne décroît pas pour minimiser le poids donnant ce bénéfice puis verticalement pour passer à l'objet suivant.

b. Résultats - cas de 4 marchandises

Pour une capacité de 995, la résolution par programmation dynamique donne :

i	0	1	2	3	4
x_i	0	7	2	1	0
y_i	0	875	977	995	995
V_i	0	882	984	1000	1000

Tableau 1. x_{Opt} pour 4 marchandises

On remarque alors que $\sum_{i=1}^{4} a_i x_i = 995$ donc le cargo est complet et on n'a pas eu besoin d'ajouter du vide pour atteindre la capacité totale. Le bénéfice est de $\sum_{i=0}^{4} c_i x_i = 1000$.

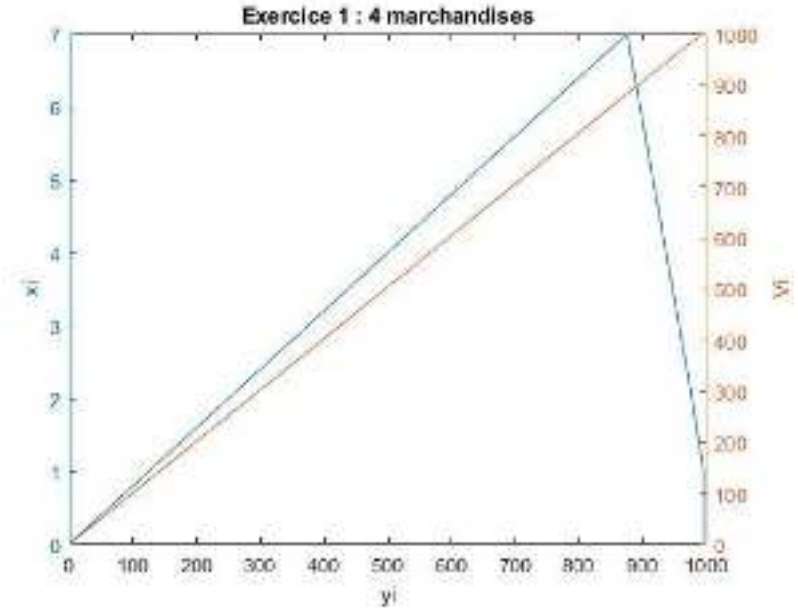

Figure 1. V_i et $x_{Opt,i}$ en fonction de y_i

Si on avait appliqué la règle heuristique d'allocation par attractivité décroissante, on aurait obtenu :

i	0	1	2	3	4
Attractivité$_i$	0	1,008	1	0,889	0,923

Tableau 2. Attractivité pour 4 marchandises

L'attractivité est calculée comme c_i/a_i. L'ordre de priorité serait donc : 1, 2, 4, 3, 0. Si à partir de ce résultat on avait rempli le cargo en chargeant le plus de marchandises 1 possible, ensuite dans la place restante le plus de marchandises 2 possible et ainsi de suite, les quantités de chaque marchandise auraient été :

i	0	1	2	3	4
x_i	0	7	2	1	0

Tableau 3. x_{Opt} pour 4 marchandises avec l'autre méthode

On remarque que le chargement aurait été le même.

c. Résultats - cas de 5 marchandises

i. Pour b=1800

Pour une capacité de 1800, la résolution par programmation dynamique donne :

i	0	1	2	3	4	5
x_i	2	17	1	1	0	0
y_i	2	1719	1770	1800	1800	1800
V_i	0	1751	1802	1831	1831	1831

Tableau 4. x_{Opt} pour 5 marchandises

On remarque alors que $\sum_{i=1}^{4} a_i x_i = 1798$ donc on a dû laisser 2 du vide pour atteindre la capacité totale. Le bénéfice est de $\sum_{i=0}^{4} c_i x_i = 1831$.

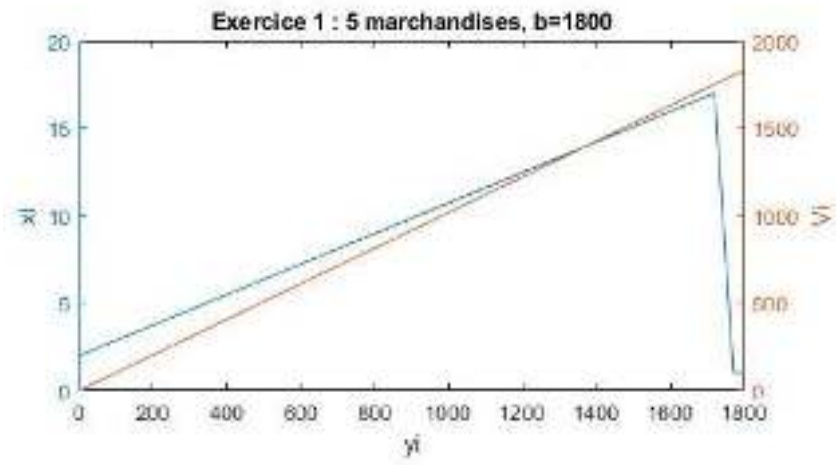

Figure 2. V_i et $x_{Opt,i}$ en fonction de y_i

Si on avait appliqué la règle heuristique d'allocation par attractivité décroissante, on aurait obtenu :

i	0	1	2	3	4	5
Attractivité$_i$	0	1,02	1	0,967	0,889	0,923

Tableau 5. Attractivité pour 5 marchandises

L'ordre de priorité serait donc : 1, 2, 3, 5, 4, 0. Si à partir de ce résultat on avait rempli le cargo avec la méthode par ordre d'attractivité, les quantités de chaque marchandise auraient été :

i	0	1	2	3	4	5
x_i	2	17	1	1	0	0

Tableau 6. x_{Opt} pour 5 marchandises avec l'autre méthode

On remarque que le chargement aurait été le même.

ii. Pour b=1854

Pour une capacité de 1854, la résolution par programmation dynamique donne :

i	0	1	2	3	4	5
x_i	0	18	0	0	2	0
y_i	0	1818	1818	1818	1854	1854
V_i	0	1854	1854	1854	1886	1886

Tableau 7. x_{Opt} pour 5 marchandises

On remarque alors que $\sum_{i=1}^{4} a_i x_i = 1854$ donc on n'a pas eu besoin d'ajouter du vide pour atteindre la capacité totale. Le bénéfice est de $\sum_{i=0}^{4} c_i x_i = 1886$.

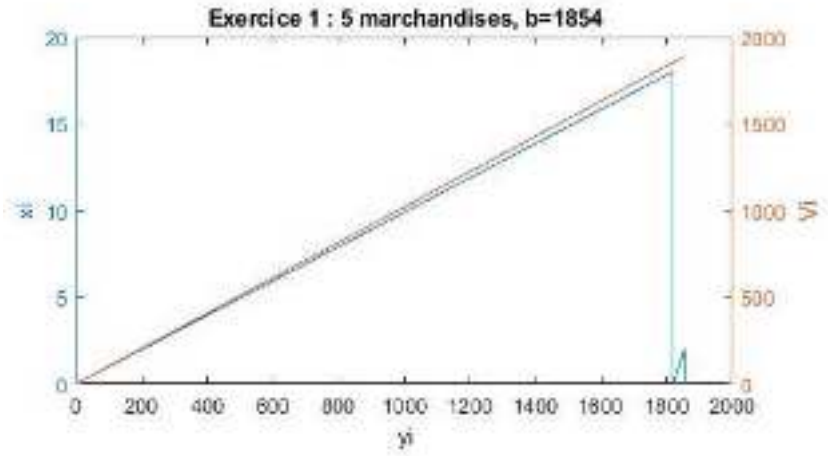

Figure 3. V_i et $x_{Opt,i}$ en fonction de y_i

Si à partir du tableau 5 on avait rempli le cargo avec la méthode par ordre d'attractivité, les quantités de chaque marchandise auraient été :

i	0	1	2	3	4	5
x_i	6	18	0	1	0	0

Tableau 8. x_{Opt} pour 5 marchandises avec l'autre méthode

On remarque que le chargement aurait été différent donc dans ce cas les marchandises ne sont pas choisies par attractivité décroissante.

iii. Pour b=1858

Pour une capacité de 1858, la résolution par programmation dynamique donne :

i	0	1	2	3	4	5

x_i	1	18	0	0	0	3
y_i	1	1819	1819	1819	1819	1858
V_i	0	1854	1854	1854	1854	1890

Tableau 9. x_{Opt} pour 5 marchandises

On remarque alors que $\sum_{i=1}^{4} a_i x_i = 1857$ donc on a laissé 1 du vide pour atteindre la capacité totale. Le bénéfice est de $\sum_{i=0}^{4} c_i x_i = 1890$.

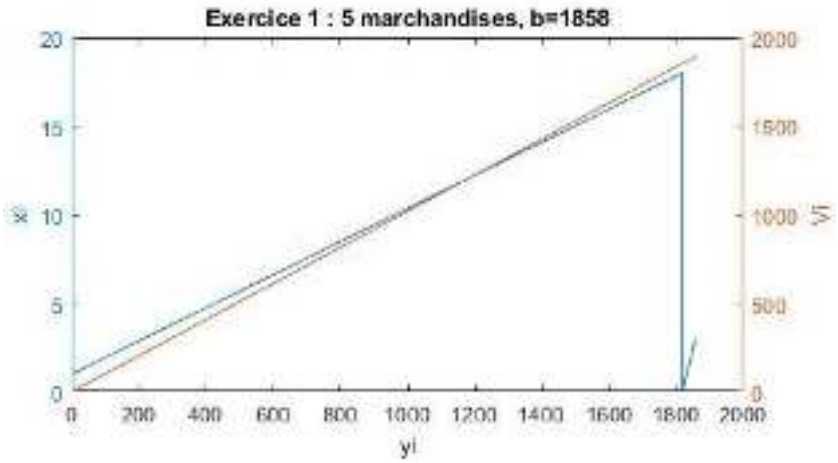

Figure 4. V_i et $x_{Opt,i}$ en fonction de y_i

Si à partir du tableau 5 on avait rempli le cargo avec la méthode par ordre d'attractivité, les quantités de chaque marchandise auraient été :

i	0	1	2	3	4	5
x_i	10	18	0	1	0	0

Tableau 8. x_{Opt} pour 5 marchandises avec l'autre méthode

On remarque que le chargement aurait été différent donc dans ce cas les marchandises ne sont pas non plus choisies par attractivité décroissante.

II. Problème de production

1. Présentation du problème

Ce problème de production a pour but de trouver la méthode optimale afin de satisfaire la demande des clients avec un coût minimal. Dans un premier temps nous allons traiter le problème par la programmation dynamique, ensuite nous l'envisagerons par la programmation linéaire.

2. Réponse aux questions

Le processus de production se traduit par l'évolution du stock suivante :

$$S_i = S_{i-1} + p_i - d_i \qquad \forall i = 1, 2, \dots, I$$

Cette relation montre que le stock à l'instant i dépend du stock précédent ainsi que de la production faite entre [i-1,i] et la demande en i. Le stock en i serait donc le stock en i-1 plus la production réalisée entre i-1 et i moins les besoins des clients en i.

Question 1

Soient Ki les capacités de production et di les demandes :

$$S_i = S_{i-1} + p_i - d_i \qquad \forall i = 1, 2, \dots, I$$

Donc

$$S_i = S_0 + \sum_{k=1}^{i} p_k - d_k \qquad \forall i = 1, 2, \dots, I \; et \; S_0 = 0$$

$$S_i = \sum_{k=1}^{i} p_k - d_k \qquad \forall i = 1, 2, \dots, I$$

Or

$$S_i \geq 0 \qquad \forall i = 1, 2, \dots, I$$

Donc

$$\sum_{k=1}^{i} p_k - d_k \geq 0$$

$$\sum_{k=1}^{i} p_k \geq \sum_{k=1}^{i} d_k$$

Or

$$K_k \geq p_k \qquad \forall k = 1, 2, \dots, I$$

Donc

$$\sum_{k=1}^{i} K_k \geq \sum_{k=1}^{i} p_k \geq \sum_{k=1}^{i} d_k \quad \forall i = 1, 2, \dots, I$$

Ainsi on vérifie que les données satisfont la condition.

i	1	2	3	4	5	6	7
$\sum_{k=1}^{i} K_k$	140	240	360	460	580	680	770
$\sum_{k=1}^{i} d_k$	100	220	320	410	530	640	750

Tableau 9. Vérification des conditions de K_i et d_i

Question 2

- Variables d'état : les stocks S_i
- Variables de commande : les productions p_i
- Équation de Hamilton-Jacobi :

$$V_{i+1}(S_{i+1}) = Min_{(p_i,S_i)}[V_i(S_i) + \gamma_{i+1}p_{i+1} + \sigma_{i+1}S_{i+1}]$$

$$\left| \begin{array}{c} S_0 = 0 \\ S_{i+1} = S_i + p_{i+1} - d_{i+1} \\ S_i \geq 0 \\ \sum_{k=1}^{i} K_k \geq \sum_{k=1}^{i} d_k \end{array} \right. \qquad \forall i = 1, 2, \dots, I$$

- Pour résoudre le problème numériquement :
  - On initialise avec $S_0 = 0$ et $V_0(S_0) = 0$
  - On itère pour i = 0 jusqu'à i = I-1. On calcule pour les différentes valeurs possibles de S_{i+1}, les valeurs possibles de V_{i+1} selon S_{i+1} et on trouve la solution optimale qui minimise V_I et on peut ainsi en déduire les S_i et p_i ∀ i = 1, 2, ..., I.
- Résultats :
  - Valeurs en divisant par 10 :

i	S_{i+1}	S_i	P_{i+1}	Vi+1
6	0	2	9	485.65
5	2	3	10	413.65
4	3	3	12	352.85
3	3	2	10	267.95
2	2	0	12	207.20
1	0	4	8	134.80
0	4	0	14	70.800

Tableau 10. Résultats divisés par 10 avec programmation dynamique

  - Valeurs sans diviser par 10 :

i	S_{i+1}	S_i	P_{i+1}	Vi+1
6	0	20	90	4856.50
5	20	30	100	4136.50
4	30	30	120	3528.50
3	30	20	100	2679.50
2	20	0	120	2072
1	0	40	80	1348
0	40	0	140	708

Tableau 11. Résultats réels avec programmation dynamique

Question 3

D'après la question 2 :

$$Min(\gamma_i p_i + \theta_i S_i)$$

$$\begin{vmatrix} S_0 = 0 \\ S_{i+1} = S_i + p_{i+1} - d_{i+1} \\ K_i = p_i + y_i \geq 0 \\ S_i \geq 0 \end{vmatrix}$$

Donc sous la forme canonique :

$$Max\ c.x$$

$$\begin{vmatrix} A.x = b \\ x \geq 0 \end{vmatrix}$$

Avec

$$b = (d_1, d_2, \dots, d_I, K_1, K_2, \dots, K_I) \quad \rightarrow vecteur\ colonne\ de\ taille\ 2I$$

$$c = (-\sigma_1, -\sigma_2, \dots, -\sigma_I, -\gamma_1, -\gamma_2, \dots, -\gamma_I, 0, \dots 0) \quad \rightarrow vecteur\ ligne\ de\ taille\ 3I$$

$$x = (S_1, S_2, \dots, S_I, p_1, p_2, \dots, p_I, y_1, y_2, \dots, y_I) \quad \rightarrow vecteur\ colonne\ de\ taille\ 3I$$

$$A = \begin{matrix} -1 & 0 & \cdots & \cdots & 0 & 1 & 0 & \cdots & \cdots & 0 & 0 & \cdots & \cdots & \cdots & 0 \\ 1 & \ddots & \ddots & \ddots & \vdots & 0 & \ddots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 & \vdots & \ddots & \ddots & \ddots & 0 & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & -1 & 0 & \cdots & \cdots & 0 & 1 & 0 & \cdots & \cdots & \cdots & 0 \\ \\ 0 & \cdots & \cdots & \cdots & 0 & 1 & 0 & \cdots & \cdots & 0 & 1 & 0 & \cdots & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots & 0 & \ddots & \ddots & \ddots & \vdots & 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots & \ddots & 0 & \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & \cdots & 0 & 0 & \cdots & \cdots & 0 & 1 & 0 & \cdots & \cdots & 0 & 1 \end{matrix} \rightarrow matrice\ de\ taille\ 2I \times 3I$$

On introduit les variables d'écarts car on a $K_i \geq p_i$. On pose donc y tel que $K_i = p_i + y_i$. La variable d'écart est donc y, qui représente l'écart entre la production maximale et la production réelle.

On assemble la matrice A en regroupant les contraintes (lignes) par équations d'état et contraintes de capacité de production et en classant les variables (colonnes) par variables de stock, variables de production et variables d'écart. Pour assembler x par blocs, on prend la matrice A et on regarde en ordre la variable qui représente chaque colonne de chaque bloc et pour assembler b, on prend la matrice A et on regarde en ordre à quoi correspond chaque ligne de la matrice (dans ce cas il s'agit des demandes et des capacités de production).

On cherche une solution réalisable simple pour :

$$\begin{cases} S_1 = p_1 - 100 \\ S_2 = p_2 + S_1 - 120 \\ S_3 = p_3 + S_2 - 100 \\ S_4 = p_4 + S_3 - 90 \\ S_5 = p_5 + S_4 - 120 \\ S_6 = p_6 + S_5 - 110 \\ S_7 = p_7 + S_6 - 110 \end{cases}$$

Une solution de x simple consiste à prendre $p_i = K_i$. On a alors :

i	1	2	3	4	5	6	7
S_i	40	20	40	50	50	40	20
p_i	140	100	120	100	120	100	90

Tableau 12. Résultats solution simple problème de production

$x = (40, 20, 40, 50, 50, 40, 20, 140, 100, 120, 100, 120, 100, 90) \rightarrow vecteur\ colonne\ de\ taille\ 2I$

Question 4

Pour résoudre le problème avec l'algorithme du simplexe, on a utilisé une fonction déjà definie sur matlab, linprog, et les résultats obtenus sont :

i	S_i	p_i	y_i	Vi
1	40	140	0	708
2	0	80	20	1348
3	20	120	0	2072
4	30	100	0	2679.50
5	30	120	0	3528.50
6	20	100	0	4136.50
7	0	90	0	4856.50

Tableau 13. Résultats avec l'algorithme du simplexe

Si on compare avec les résultats obtenus avec la programmation dynamique, on remarque que ce sont les mêmes. De ce fait, la programmation linéaire n'est ni plus ni moins efficace que la programmation dynamique.

III. Affectation des véhicules à la demande

1. Présentation du problème

Ce problème a pour but de d'optimiser le cout de transport de véhicule pour satisfaire l'offre et la demande. La somme sur i des véhicules déplacés des points i a un point j correspond au nombre de véhicule manquant au point j. On a donc :

$$\sum_{i \in \varepsilon} z_{ij} = m_j$$

De même, la somme sur j des véhicules déplacer d'un point i aux points j correspond au nombre de véhicule en excès au point i.

$$\sum_{j \in M} z_{ij} = -m_i$$

De plus, le déplacement d'un véhicule d'un point i à j est compté positivement sachant que ε correspond à l'ensemble des points possédants des véhicules en excès et M l'ensemble des points en manque de véhicule.

$$z_{ij} \geq 0\,, \qquad \forall i \in \varepsilon\,, \forall j \in M$$

2. Analyse du problème

Question 1

Le programme linéaire est un problème de transport.

On a :

$$\sum_{j \in M} z_{ij} = -m_i$$

Or

$$z_{ij} \geq 0\,, \qquad \forall i \in \varepsilon$$

Donc on obtient la relation suivante :

$$\sum_{j \in M} z_{ij} = -m_i \geq 0$$

Et

$$m_i \leq 0$$

On note m_i le nombre de véhicules manquant au dépôt i et m_j le nombre de véhicules en excès donc on sait que :

$$m_i + mj = 0$$

Et

$$\sum_{i \in \varepsilon} z_{ij} - \sum_{j \in M} z_{ij} = 0$$

Question 2

On peut mettre le problème sous la forme canonique :

$$Max\ c.x$$

$$\left| \begin{matrix} A.x = b \\ x \geq 0 \end{matrix} \right.$$

Avec :

$$c = -\sum_{i=1}^{10} \sum_{j=i}^{10} d_{ij}$$

$$b = (m_i) \rightarrow \ vecteur\ colonne\ de\ taille\ 10$$

$$x = \left(x_{1,2}, \dots, x_{1,10}, x_{3,1}, \dots, x_{3,10}, x_{4,1}, \dots, x_{4,10}, \dots, x_{10,9}\right)$$
$$\rightarrow vecteur\ colonne\ de\ taille\ 54$$

Les $x_{i,i}$ étant nuls et sans inclure les x qui partent d'un dépôt qui manque de véhicules

$$A = \begin{matrix} 1 & 1 & 0 & \cdots & \cdots & 0 \\ 1 & 0 & 1 & 0 & \vdots & \vdots \\ \vdots & \vdots & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \ddots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & 1 \\ 1 & 0 & \cdots & \cdots & 0 & 0 \end{matrix}$$

$$\begin{matrix} 0 & 1 & 1 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 0 & \vdots & \vdots \\ \vdots & \vdots & 0 & \ddots & \ddots & 0 \\ \vdots & 1 & \vdots & \ddots & \ddots & 1 \\ \vdots & 0 & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & \cdots & 0 & 0 \end{matrix}$$

$$\vdots$$

$$\begin{matrix} 0 & \cdots & \cdots & \cdots & 0 & 1 \end{matrix}$$

Une matrice de 54 lignes et 10 colonnes

IV. Un problème de logistique urbaine

1. Présentation du problème

Ce problème de logistique urbaine a pour but d'optimiser l'utilisation des dépôts afin de satisfaire la demande des clients avec un coût minimal. Le problème sera traité avec la programmation linéaire.

2. Analyse du problème

Question 1

On vérifie que les données D et S_j ∀j = 1, ..., 8 sont bien compatibles puisque :

$$\sum_{j=1}^{8} S_j = 150 + 250 + 400 + 200 + 180 + 140 + 240 + 320 = 1880$$

Donc

$$\sum_{j=1}^{8} S_j = D$$

De plus, on remarque que la contrainte d'allocation de la quantité de marchandises est redondante puisqu'en partant de la contrainte de distribution du stock de chaque dépôt entre les points de vente et en l'exprimant et en sommant pour tous i compris entre 1 et 3 on trouve qu'elle est combinaison linéaire des autres contrainte :

$$0 = -d_1 - d_2 - d_3 + \sum_{j=1}^{8} x_{1j} + \sum_{j=1}^{8} x_{2j} + \sum_{j=1}^{8} x_{3j}$$

Or

$$S_j = \sum_{i=1,2,3} x_{ij} \quad \forall j = 1, 2, \dots, 8$$

$$\sum_{j=1}^{8} S_j = \sum_{i=1}^{3} \sum_{j=1}^{8} x_{ij} \qquad (1)$$

Donc

$$0 = -d_1 - d_2 - d_3 + \sum_{j=1}^{8} S_j \qquad (2)$$

Or

$$\sum_{j=1}^{8} S_j = D = d_1 + d_2 + d_3$$

Ce qui est redondant par rapport à (2).

L'allocation de la quantité totale de marchandise est combinaison linéaire de la satisfaction des demandes des tous les points de vente comme montré dans l'équation (1).

Question 2

On peut mettre le problème sous la forme canonique :

$$Max\ c.x$$

$$\begin{cases} A.x = b \\ x \geq 0 \end{cases}$$

Avec

$$b = (0,0,0,S_1,S_2,\dots,S_8) \quad \rightarrow vecteur\ colonne\ de\ taille\ 11$$

$$c = (C_{11},C_{12},\dots,C_{18},C_{21},C_{22},\dots,C_{28},C_{31},C_{32},\dots C_{38},0,0,0)$$
$$\rightarrow vecteur\ ligne\ de\ taille\ 27$$

$$x = (x_{11},x_{12},\dots,x_{18},x_{21},x_{22},\dots,x_{28},x_{31},x_{32},\dots,x_{38},d_1,d_2,d_3)$$
$$\rightarrow vecteur\ colonne\ de\ taille\ 27$$

$$A = \begin{matrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & -1 \end{matrix}$$

$$\begin{matrix} 1 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 & 0 & \cdots & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots & 0 & \ddots & \ddots & \vdots & 0 & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 & \vdots & \ddots & \ddots & 0 & \vdots & \ddots & \ddots & 0 & \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 & 1 & 0 & \cdots & \cdots & 0 \end{matrix} \rightarrow matrice\ de\ taille\ 11 \times 27$$

De même, on observe que A est de rang maximum et que le vecteur x se décompose en quatre parties : la première ce sont les quantités livrées depuis le dépôt 1, la deuxième celles livrées depuis le dépôt 2, la troisième celles livrées depuis le dépôt 3 et la quatrième ce sont les quantités de marchandise à allouer à chaque dépôt.

3. Résolution par la programmation linéaire

On obtient :

	d_i	L_1	L_2	L_3	L_4	L_5	L_6	L_7	L_8
D_1	960	150	250	0	0	180	140	240	0
D_2	0	0	0	0	0	0	0	0	0
D_3	920	0	0	400	200	0	0	0	320

Tableau . Résultats problème de logistique urbaine

On constate bien que $\sum_{i=1}^{3} d_i = 960 + 920 = D = 1880$.

De même, on constate que le dépôt 2 (D_2) n'est pas utile car on ne livre aucune marchandise depuis ce dépôt et que la plupart des $x_{i,j}$ sont nuls. De plus, on remarque qu'on livre à chaque point de vente la marchandise dont il a besoin depuis le dépôt d'où le coût de livraison est le moins cher. En effet, un dépôt est toujours privilégié face aux autres.

4. Compléments

Question 1

Avec $L_6 = 280$, on obtient :

	d_i	L_1	L_2	L_3	L_4	L_5	L_6	L_7	L_8
D_1	1100	150	250	0	0	180	280	240	0
D_2	0	0	0	0	0	0	0	0	0
D_3	920	0	0	400	200	0	0	0	320

Tableau . Résultats problème de logistique urbaine avec $L_6 = 280$

On constate que $\sum_{i=1}^{3} d_i = 1100 + 920 = D + ajout_{L6} = 1880 + 140 = 2020$.

Les résultats étaient prévisibles car on observe que le dépôt 1 contient 140 unités de marchandises supplémentaires, correspondant à l'augmentation de la demande au point de vente 6, puisqu'il est celui qui possède le coût de transport le plus faible vers le point de vente 6.

Question 2

Avec $C_{2,6} = 4$, on obtient :

	d_i	L_1	L_2	L_3	L_4	L_5	L_6	L_7	L_8
D_1	820	150	250	0	0	180	0	240	0
D_2	280	0	0	0	0	0	280	0	0
D_3	920	0	0	400	200	0	0	0	320

Tableau . Résultats problème de logistique urbaine avec $C_{2,6} = 4$

On constate que $\sum_{i=1}^{3} d_i = 820 + 280 + 920 = D + ajout_{L6} = 1880 + 140 = 2020$.

L'impact de l'aménagement sur l'organisation des livraisons est l'utilisation du dépôt 2 pour livrer le point de vente 6 car maintenant son coût de transporte est le plus faible comparé aux autres dépôts. On observe donc que le dépôt 2 contient 280 unités de marchandises supplémentaires, correspondant à la demande au point de vente 6.

Annexes

I. Résolution numérique TP1

```
%CAS DE 4 MARCHANDISES
capacite=995;
donnes=[1 125 51 18 13
        0 126 51 16 12];
n=10; %on suppose une cantitè de 10 pour chaque marchandise

[M_4_marchandises,benefices_4_marchandises,sol_4_marchandises,iteratio
ns_4_marchandises,chargement_4_marchandises,v_4] =
resultat_programmation_dynamique(capacite,donnes(1,1:end),donnes(2,1:e
nd),n);

disp(benefices_4_marchandises)
disp(sol_4_marchandises)
disp(iterations_4_marchandises)

%on crée les graphes demandés
figure(1)
yyaxis left
plot(v_4(2,1:end),v_4(1,1:end))
yyaxis right
plot(v_4(2,1:end),v_4(3,1:end))
title('Exercice 1 : 4 marchandises')
xlabel('yi')
yyaxis left
ylabel('xi')
yyaxis right
ylabel('Vi')

%CAS DE 5 MARCHANDISES
donnes_5=[1 101 51 30 18 13
        0 103 51 29 16 12];
n=20; %on suppose une cantitè de 20 pour chaque marchandise

%cas de b=1800
b1=1800;
[M_5_m_b1,benefices_5_mar_b1,sol_5_mar_b1,iterations_5_mar_b1,chargeme
nt_5_mar_b1,v_5_b1] =
resultat_programmation_dynamique(b1,donnes_5(1,1:end),donnes_5(2,1:end
),n);

disp(benefices_5_mar_b1)
disp(sol_5_mar_b1)
disp(iterations_5_mar_b1)

%cas de b=1854
b2=1854;
[M_5_m_b2,benefices_5_mar_b2,sol_5_mar_b2,iterations_5_mar_b2,chargeme
nt_5_mar_b2,v_5_b2] =
resultat_programmation_dynamique(b2,donnes_5(1,1:end),donnes_5(2,1:end
),n);

disp(benefices_5_mar_b2)
disp(sol_5_mar_b2)
disp(iterations_5_mar_b2)
```

```
%cas de b=1858
b3=1858;
[M_5_m_b3,benefices_5_mar_b3,sol_5_mar_b3,iterations_5_mar_b3,chargeme
nt_5_mar_b3,v_5_b3] =
resultat_programmation_dynamique(b3,donnes_5(1,1:end),donnes_5(2,1:end
),n);

disp(benefices_5_mar_b3)
disp(sol_5_mar_b3)
disp(iterations_5_mar_b3)

%on crée les graphes demandés
figure(2)
subplot(2,2,1);
yyaxis left
plot(v_5_b1(2,1:end),v_5_b1(1,1:end))
yyaxis right
plot(v_5_b1(2,1:end),v_5_b1(3,1:end))
title('Exercice 1 : 5 marchandises, b=1800')
xlabel('yi')
yyaxis left
ylabel('xi')
yyaxis right
ylabel('Vi')

subplot(2,2,2);
yyaxis left
plot(v_5_b2(2,1:end),v_5_b2(1,1:end))
yyaxis right
plot(v_5_b2(2,1:end),v_5_b2(3,1:end))
title('Exercice 1 : 5 marchandises, b=1854')
xlabel('yi')
yyaxis left
ylabel('xi')
yyaxis right
ylabel('Vi')

subplot(2,2,3);
yyaxis left
plot(v_5_b3(2,1:end),v_5_b3(1,1:end))
yyaxis right
plot(v_5_b3(2,1:end),v_5_b3(3,1:end))
title('Exercice 1 : 5 marchandises, b=1858')
xlabel('yi')
yyaxis left
ylabel('xi')
yyaxis right
ylabel('Vi')


function [M,benefices,sol,etats_iterations,chargement_optimal,valeurs]
= resultat_programmation_dynamique(capacite,a,c,n)

%on agrandit les matrices des poids et des coûts en supposant qu'il y
a 10
%unités de chaque marchandise
a2=nouveau_vecteur(a,n);
c2=nouveau_vecteur(c,n);
```

```
[M,benefices,c_o] = programmation_dynamique(capacite,a2,c2);

%réduction des numéros des produits obtenus
c_o=floor(c_o./n);
%comptage des quantités
x(length(a2)/n)=0;%matrice des quantités
for i=1:length(c_o)
    for k=1:length(a2)
        if c_o(i)==k
        x(k)=x(k)+1;
        end
    end
end

%on vérifie combien de place vide on laissera
somme=0;
for i=1:length(a)-1
    somme=somme+x(i)*a(i+1);
    if i==length(a)-1
       if somme==capacite
           x=[x(length(a)) x(1:(length(a)-1))];
       else
           vide=capacite-somme;
           x=[vide x(1:(length(a)-1))];
       end
    end
end

%on renvoie un tableau avec les x optimaux pour chaque i
rowNames={'i','xi'};
sol=array2table([0:(length(a)-1);x],'RowNames',rowNames);

%on contruit un tableau qui contient les Vi, xi et yi selon les
résultats obtenus
valeurs=zeros(2,length(x));
valeurs=[x;valeurs];

for j=1:length(a)
   if j==1
      valeurs(2,j)=a(j)*x(j);
      valeurs(3,j)=0;
   else
      valeurs(2,j)=valeurs(2,j-1)+a(j)*x(j);
      valeurs(3,j)=valeurs(3,j-1)+c(j)*x(j);
   end
end

%on renvoie un tableau avec les valeurs pour chaque i
etats_iterations=array2table([0:(length(a)-
1);valeurs],'RowNames',{'i','xi','yi','Vi'});

chargement_optimal=0;
for j=1:5
   chargement_optimal=chargement_optimal+a(j)*x(j);
end

end


function a2=nouveau_vecteur(a,n)
```

```
%a2=nouveau_vecteur(a,n)
%on agrandit les matrices des poids et des coûts en supposant qu'il y
a n
%unités de chaque marchandise
%a2 : nouveau vecteur qui tient en compte les n unités de chaque
%marchandise
a2=[];
    for i=0:length(a)-1
    a2(i*n+1:(i+1)*n)=a(i+1);
    end
end


function [M,benefices,c_o] = programmation_dynamique(b,a,c)
%[M] = programmation_dynamique(b,objets,a,c)
%on déroule le programme pour le problème de type sac à dos
%b : capacité
%a : poids
%c : coûts

%on initialise la matrice qui contiendra les valeurs qui nous donne
l'algorithme
M=zeros(size(a,1),b+1);

for j=1:b+1 %initialisation de l'équation de Hamilton-Jacobi-Bellman
    M(1,j)=0;
end

for i=2:length(a) %on remplit le reste des valeurs
    for j=1:b+1
        if j==1
            M(i,j)=0; %la première colonne correspond au chargement 0
donc tous les valeurs de cette colonne sont 0
        else
            if j<=a(i) %si la capacité du chargement j est inférieure
au poids de l'élément, on ne l'ajoute pas
                M(i,j)=M(i-1,j);
            else
                M(i,j)=max(M(i-1,j),M(i-1,j-a(i))+c(i)); %si la
capacité du chargement j est supérieure au poids de l'élément,
                                                        %on choisit
de le garder que si sa valeur ajoutée est supérieure à
                                                        %la valeur
calculée pour une capacité de même taille avec i-1 objets
            end
        end
    end
end

benefices=M(size(M,1),size(M,2));

%on cherche le chemin optimal dans la matrice M
c_o=[];%chemin optimal (=liste des marchandises)

%on récupère dans la dernière ligne le poids minimal pour faire un
bénéfice maximal
j=size(M,2);
while M(size(M,1),j)==M(i,j-1)
j=j-1;
end
```

```
i=size(M,1);
while j>0 && i>0
    while i>1 && M(i-1,j)==M(i,j)
        i=i-1;
    end
    j=j-a(i);
    if j>0
        c_o=[c_o i];
    end
i=i-1;
end

end
```

II. Résolution numérique TP2

Programmation dynamique

```
%matrice des valeurs données
M=[14 10 12 10 12 10 9
    10 12 10 9 12 11 11
    5 8 6 6 7 6 8
    0.2 0.3 0.2 0.25 0.3 0.4 0.45];
%stock initial
so=0;

[resultat,valeurs_optimaux,valeurs_optimaux_reels] =
programmation_dyn_production(M,so);


function
[recapitulatif_valeurs,valeurs_optimaux,valeurs_optimaux_reels] =
programmation_dyn_production(M,so)
%[recapitulatif_valeurs,valeurs_optimaux] =
programmation_dyn_production(M,so)
% Cet algorithme trouve la production et le stock optimal pour chaque
% période afin de répondre aux besoins des clients

%on trouve la quantité des fois à itérer selon les temps donnés
for i=0:(size(M,2)-1)
   %on initialise avec le stock donné en t=0
   if i==0
     M_ite1=zeros(M(1,1)+1,4);
     for j=1:size(M_ite1,1)
         M_ite1(j,3)=j-1;
         M_ite1(j,2)=so;
         M_ite1(j,1)=M_ite1(j,2)+M_ite1(j,3)-M(2,1);
         if M_ite1(j,1)<0
             M_ite1(j,4)=inf;
         else
            M_ite1(j,4)=M_ite1(j,3)*M(3,1)+M_ite1(j,1)*M(4,1);
         end
     end
     resultat=[];
     %on cherche à garder les résultats qui sont possibles en gardeant
     %seulement les stocks supérieures ou égales à 0
     for j=1:size(M_ite1,1)
         if isinf(M_ite1(j,4))==0
```

```
            resultat=[resultat;i M_ite1(j,:)];
        end
    end
  else
    %on itère pour les autres i
    ite=[];
    %on trouve les valeurs des stocks pour chaque quantité de
prduction
    %selon la capacité de production maximale en i
    for j=0:M(1,i+1)
        it=[];
        c=sum(resultat(:,1)==i-1);
        it(:,2)=resultat((size(resultat,1)-c+1):size(resultat,1),2);
        it(:,3)=j;
        for z=1:c
            it(z,1)=it(z,2)+it(z,3)-M(2,i+1);
            if it(z,1)<0
               it(z,4)=inf;
            else
               for t=1:size(resultat,1)
                  if resultat(t,1)==i-1 && resultat(t,2)==it(z,2)

it(z,4)=resultat(t,5)+it(z,3)*M(3,i+1)+it(z,1)*M(4,i+1);
                  end
               end
            end
        end
        %on ne garde que les stocks possibles qui sont supérieures
ou
        %égales à 0
        for r=1:size(it,1)
            if it(r,1)>=0
               ite=[ite;it(r,:)];
            end
        end
    end
    stock_possible=unique(ite(:,1));
    %on compare les fonctions valeurs des stocks qui sont égaux et
on
    %choisi la plus petite
    for p=1:size(stock_possible,1)
        vo=inf;
        for k=1:size(ite,1)
            if ite(k,1)==stock_possible(p,1) && vo<ite(k,4)
               vo=vo;
            elseif ite(k,1)==stock_possible(p,1) && vo>ite(k,4)
                    vo=ite(k,4);
            end
        end
        [l c]=find(ite==vo);
        resultat=[resultat;i ite(l,:)];
    end
  end
end
recapitulatif_valeurs=array2table(resultat,'VariableNames',{'i','Si_pl
us_un','Si','Pi_plus_un','Vi_plus_un'});
valeurs_opt=[];
%à partir de la valeurs dans le dernier temps on remonte pour trouver
les valeurs optimaux pour chaque période
for h=(size(M,2)-1):-1:0
    if h==(size(M,2)-1)
```

```
        op=inf;
        for n=1:size(resultat,1)
            if resultat(n,1)==h && op<resultat(n,5)
               op=op;
            elseif resultat(n,1)==h && op>resultat(n,5)
               op=resultat(n,5);
            end
        end
        [lig col]=find(resultat==op);
        valeurs_opt=[valeurs_opt;resultat(lig,:)];
    else
        g=size(valeurs_opt);
        for n=1:size(resultat,1)
            if resultat(n,1)==h && resultat(n,2)==valeurs_opt(g(1),3)
               valeurs_opt=[valeurs_opt;resultat(n,:)];
            end
        end
    end
end
%on mutiplie par 10 les stocks, les production et les fonctions
valeurs car
%on les avait divisé par 10 pour rendre les calculs numériques plus
rapides
valeurs_reels=valeurs_opt(:,2:end).*10;
valeurs_reels=[valeurs_opt(:,1) valeurs_reels];
valeurs_optimaux=array2table(valeurs_opt,'VariableNames',{'i','Si_plus
_un','Si','Pi_plus_un','Vi_plus_un'});
valeurs_optimaux_reels=array2table(valeurs_reels,'VariableNames',{'i',
'Si_plus_un','Si','Pi_plus_un','Vi_plus_un'});
end
```

Programmation linéaire

```
%on définit les matrices et les vecteurs du problème
Aeq = [-eye(7) eye(7) zeros(7,7)
     zeros(7,7) eye(7) eye(7)];
for i=1:6
    Aeq(i+1,i)=1;
end
beq=[100;120;100;90;120;110;110;140;100;120;100;120;100;90];
c=[0.2;0.3;0.2;0.25;0.3;0.4;0.45;5;8;6;6;7;6;8;0;0;0;0;0;0;0];

lb=zeros(21,1);
ub=[];
A=[];
b=[];
%on utilise la fonction linprog pour résoudre le problème
x=linprog(c,A,b,Aeq,beq,lb,ub);

%on trouve les valeurs de v pour chaque itération
vi=zeros(7,1);
for i=1:7
    if i==1
        vi(1)=c(8)*x(8)+c(1)*x(1);
    else
        vi(i)=vi(i-1)+c(i+7)*x(i+7)+c(i)*x(i);
    end
end
```

```
valeurs_optimaux=array2table([transpose(1:7) x(1:7) x(8:14) x(15:21)
vi],'VariableNames',{'i','Si','Pi','yi','Vi'});
```

III. Résolution numérique TP3

```
%on définit les matrices et les vecteurs du problème
coord=[[0,0];[4,4];[4,2];[6,2];[7,0];[7,5];[1,7];[1,2];[2,0];[0,3];
m=[-2;7;-4;-3;3;-5;-1;4;6;-5];
A=[];
b=[];
c=[];
Aeq=[];
lb=zeros(54,1);
ub=[];
%on crée la matrice A
beq=[2;7;4;3;3;5;1;4;6;5];
Aeq=[];

for i=1:10
    int=zeros(10,10);
    if m(i)<0
        for j=1:10
                if i==j
                    for h=1:10
                        if m(h)>0
                           int(j,h)=1;
                        end
                    end
                else
                     if m(j)>0
                        int(j,j)=1;
                     end
                end
        end
    end
    if isempty(int)==0
       Aeq=[Aeq int];
    end
end

Aeq=[Aeq(:,1:10) Aeq(:,21:40) Aeq(:,51:70) Aeq(:,91:100)];
Aeq=[Aeq(:,2:12) Aeq(:,14:23) Aeq(:,25:35) Aeq(:,37:46) Aeq(:,48:59)];

%on créer la matrice C
c=[];
for i = 1:10
    if i==1 || i==3 || i==4 || i==6 || i==7 || i==10
        for j = 1:10
            if j~=i
                d=sqrt((coord(i,1)-coord(j,1))^2+(coord(i,2)-
coord(j,2))^2);
                c=[c, d];
            end
        end
    end
end

%on utilise la fonction linprog pour résoudre le problème
x=linprog(c,A,b,Aeq,beq,lb,ub);
```

IV. Résolution numérique TP4

```
%on définit les matrices et les vecteurs du problème
Aeq = [ones(1,8) zeros(1,16) -1 0 0
     zeros(1,8) ones(1,8) zeros(1,8) 0 -1 0
     zeros(1,16) ones(1,8) 0 0 -1
     eye(8) eye(8) eye(8) zeros(8,3)];
beq= [0;0;0;150;250;400;200;180;140;240;320];
c= [10;8;12;14;7;5;7;12;12;9;19;15;13;8;14;11;14;18;10;
    9;12;14;11;8;0;0;0];

lb=zeros(27,1);
ub=[];
A=[];
b=[];
%on utilise la fonction linprog pour résoudre le problème
x = linprog(c,A,b,Aeq,beq,lb,ub);

valeurs_optimaux=array2table([x((size(x)-2):end)
[transpose(x(1:8));transpose(x(9:16));transpose(x(17:24))]],'Variab
leNames',{'di','L1','L2','L3','L4','L5','L6','L7','L8'},'RowNames',
{'D1','D2','D3'});

%Compléments
%Question 1

%on change la valeur de L6
beq= [0;0;0;150;250;400;200;180;280;240;320];
%on utilise la fonction linprog pour résoudre le problème
x = linprog(c,A,b,Aeq,beq,lb,ub);

valeurs_optimaux_queston1=array2table([x((size(x)-2):end)
[transpose(x(1:8));transpose(x(9:16));transpose(x(17:24))]],'Variab
leNames',{'di','L1','L2','L3','L4','L5','L6','L7','L8'},'RowNames',
{'D1','D2','D3'});

%Question 2
%on change la valeur de C26
c= [10;8;12;14;7;5;7;12;12;9;19;15;13;4;14;11;14;18;10;
    9;12;14;11;8;0;0;0];
%on utilise la fonction linprog pour résoudre le problème
x = linprog(c,A,b,Aeq,beq,lb,ub);

valeurs_optimaux_queston2=array2table([x((size(x)-2):end)
[transpose(x(1:8));transpose(x(9:16));transpose(x(17:24))]],'Variab
leNames',{'di','L1','L2','L3','L4','L5','L6','L7','L8'},'RowNames',
{'D1','D2','D3'});
```