Mini-projet de méthode de la recherche opérationnelle : Applications de la programmation linéaire

M. DUCHÊNE Alix - M. RIVOIRARD Lucas - M. WALKER Raphaël

4 mai 2013

FIGURE 1 – *http ://www.sxc.hu/photo/1159613*

Table des matières

Première partie

Exercice 1 : Problème de production

Introduction

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

$$s_i \;=\; s_{i-1} + p_i - d_i \;\forall \qquad i = 1, 2, ..., I$$

Ainsi le stock à l'instant t_i est égal au stock à l'instant précédant t_{i-1} auquel s'ajoute la production entre $[t_{i-1}; t_i]$ et en retirant la demande des clients à l'instant t_i.

Question 1

Soit K_i les capacités de production et d_i les demandes.

$$\forall\ i = 1, 2, ..., I \qquad s_i = s_{i-1} + p_i - d_i$$

Donc

$$\forall\ i = 1, 2, ..., I \qquad s_i = s_0 + \sum_{k=1}^{i} p_k - d_k$$

Or

$$s_0 = 0$$

Donc

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

De plus

$$\forall\ i = 1, 2, ..., I \qquad s_i \geq 0 \;\Rightarrow\; \sum_{k=1}^{i} p_k - d_k \geq 0 \;\Rightarrow\; \sum_{k=1}^{i} p_k \geq \sum_{k=1}^{i} d_k$$

Or

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

Donc

$$\forall\ i = 1, 2, ..., I \qquad \sum_{k=1}^{i} K_k \geq \sum_{k=1}^{i} p_k \geq \sum_{k=1}^{i} d_k \tag{1}$$

Ainsi on vérifie que les données sont en accord avec la condition (1) :

i	1	2	3	4	5	6	7
$\sum_{k=1}^{i} K_k$	140	240	350	450	570	670	760
$\sum_{k=1}^{i} d_k$	100	220	320	410	530	640	755

Question 2

- Les variables d'états sont des stocks s_i et les variables de commandes sont les productions p_i
- L'équation de Hamilton-Jacobi pour ce problème de production est :

$$V_{i+1}(s_{i+1}) \;=\; Min(p_i, s_i)[V_i(s_i) + \gamma_{i+1} * p_{i+1} + \theta_{i+1} * s_{i+1}]$$

$$\left|\begin{array}{ll} s_0 = 0 & \\ s_{i+1} = s_i + p_{i+1} - d_{i+1}, & i = 0, 1, ..., I-1 \\ \sum_{k=1}^{i} K_k \geq \sum_{k=1}^{i} d_k & \\ s_i \geq 0 & \end{array}\right.$$

- La méthode de résolution numérique s'énonce comme suit :
  - $s_0 = 0$ et $V_0(s_0) = 0$
  - Itération pour i=0 à i=I-1. On calcule pour les différentes valeurs possibles de $s_{i-1]}$ les valeurs possibles de V_{i+1} et après de longs calculs on trouve la solution optimale qui minimise V_I et on peut ainsi en déduire les s_i et $p_i \; \forall \; i = 1, 2, ..., I$.

Question 3

On a, d'après la question 2 :

$$Min(\gamma_i * p_i + \theta_i * s_i)$$

$$\left|\begin{array}{l} s_0 = 0 \\ s_{i+1} = s_i + p_{i+1} - d_{i+1} \\ K_i = p_i + y_i \geq 0 \\ s_i \geq 0 \end{array}\right.$$

On peut mettre le problème sous la forme canonique :

$$Max \; c.x$$

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

Avec

$b = (d_1, d_2, ..., d_I, K_1, K_2, ..., K_1)$ vecteur colonne de taille 2.I

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

$x = (s_1, s_2, ..., s_I, p_1, p_2, ...p_I, y_1, y_2, ..., y_I)$ vecteur colonne de taille 2.I

$A =$

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

Matrice de taille 2I*3.I

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

$$\begin{cases} s_1 = p_1 - 100 \\ s_2 = p_2 + d_1 - 120 \\ s_3 = p_3 + d_2 - 100 \\ s_4 = p_4 + d_3 - 90 \\ s_5 = p_5 + d_4 - 120 \\ s_6 = p_6 + d_5 - 110 \\ s_7 = p_7 + d_6 - 115 \end{cases}$$

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	60	70	80	80	70	45
p_i	140	100	110	100	120	100	90

$x = (40, 60, 70, 82, 80, 70, 45, 140, 100, 110, 100, 120, 100, 90)$ vecteur colonne de taille 2.I

Question 4

L'algorithme du simplexe nous donne les résultats suivants :

i	1	2	3	4	5	6	7
s_i	$3.5 * 10^8$	$5.4 * 10^4$	1.01	0.71	6.76	1.57	13.1
p_i	1652	1.48	1.72	$1.8 * 10^8$	2.56	311	2.73
y_i	1	1	1	1	1	1	1

Deuxième partie

Exercice 2 : Un problème de logistique urbaine

1 Analyse du problème

1.1 Question 1

On vérifie assez rapidement que les données D et $S_j \ \forall \ y = 1, ..., 8$ sont bien compatibles puisque :

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

D'où

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

De plus, la contrainte d'*allocation de la quantité de marchandises* est redondante puisqu'elle est combinaison linéaire des autres contraintes. En effet, 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 :

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

Or d'après la contrainte de *satisfaction des demandes des points de vente* :

$$S_j = \sum_{i=1}^{8} x_{ij} \qquad \forall \ y = 1, ..., 8$$

Ainsi en exprimant et sommant cette expression pour tous j compris entre 1 et 8 on a :

$$\sum_{j=1}^{8} S_j = \sum_{j=1}^{3} \sum_{i=1}^{8} x_{ij} \tag{3}$$

Ainsi en combinant les équations (2) - (3) on obtient :

$$0 = d_1 - d_2 - d_3 - \sum_{j=1}^{8} S_j \tag{4}$$

Or la contrainte d'*allocation de la quantité de marchandises* est :

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

Ce qui est bien redondant par rapport à l'équation (4)

1.2 Question 2

On peut mettre le problème sous la forme canonique :

$$Max\ c.x$$
$$\left|\begin{array}{r} A.x = b \\ x \geq 0 \end{array}\right.$$

Avec

$$x = (x_{11}, x_{12}, ..., x_{18}, x_{21}, x_{22}, ..., x_{28}, x_{31}, ..., x_{38}, d_1, d_2, d_3) \qquad \text{vecteur colonne de taille 27}$$
$$c = (C_{11}, C_{12}, ..., C_{18}, C_{21}, C_{22}, ..., C_{28}, C_{31}, ..., C_{38}, 0, 0, 0) \qquad \text{vecteur ligne de taille 27}$$

$$b = (S_1, S_2, ..., S_8, S_1, S_2, ..., S_8, S_1, S_2, ..., S_8, 0, 0, 0) \qquad \text{vecteur colonne de taille 27}$$

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

Remarque Nous avons fait ici le choix de prendre une matrice A **carrée** de taille 27 d'où la redondance d'informations (une matrice de taille 11*27 aurait été suffisante)

2 Résolution par la programmation linéaire

2.1 Résultats

La résolution par la programmation linéaire nous donne les valeurs optimales des $x_{i,j}$

$x_{i,j}$	1	2	3	4	5	6	7	8
1	150	250	0	0	180	140	240	0
2	0	0	0	0	0	0	0	0
3	0	0	400	200	0	0	0	320

Et pour les d_i :

$$d_1 = 960$$
$$d_2 = 0$$
$$d_3 = 920$$

On vérifie bien que l'on a :

$$\sum_{i=1}^{3} d_i = 690 + 920 = 1880 = D$$

2.1.1 Analyse

On constate que la plupart des $x_{i,j}$ sont nuls. De plus, pour chaque point de vente, un dépôt est privilégié face aux autres. Par exemple pour le point de vente 4, le dépôt 3 est favorisé($x_{4,3} = 200$). En effet cela est expliqué par le coût du transport vers le dépôt 3 jusqu'au point de vente 4 ($C_{4,3} = 9$) en comparaison au coût du transport des dépôts 1 et 2 jusqu'au point de vente 4 ($C_{4,1} = 14$ et $C_{4,2} = 15$).

De plus le dépôt 2 n'est jamais utilisé puisque le coût de transport de ce dépôt vers tous les points de vente est toujours supérieur aux autres dépôts.

3 Compléments

3.1 Question 1

3.1.1 Résultats

On procède au changement $L_6 = 280$ et on obtient :

$x_{i,j}$	1	2	3	4	5	6	7	8
1	150	250	0	0	180	**280**	240	0
2	0	0	0	0	0	0	0	0
3	0	0	400	200	0	0	0	320

Et pour les d_i :

$$d_1 = 1100$$
$$d_2 = 0$$
$$d_3 = 920$$

On vérifie bien que l'on a :

$$\sum_{i=1}^{3} d_i = 110 + 920 = 1880 + 140 = D + Ajout_{L6}$$

3.1.2 Analyse

On observe que le dépôt 1 contient 140 unités de marchandises supplémentaires, correspondant à l'augmentation de la demande au point de vente 6, puisque c'est le dépôt qui possède un coup de transport vers le point de vente 6 le moins onéreux.

3.2 Question 2

3.2.1 Résultats

On procède au changement $C_{2,6} = 4$ et on obtient :

$x_{i,j}$	1	2	3	4	5	6	7	8
1	150	250	0	0	180	0	240	0
2	0	0	0	0	0	**280**	0	0
3	0	0	400	200	0	0	0	320

Et pour les d_i :

$$d_1 = 1100$$

$$d_2 = 0$$

$$d_3 = 920$$

On vérifie bien que l'on a :

$$\sum_{i=1}^{3} d_i = 110 + 920 = 1880 + 140 = D + Ajout_{L6}$$

3.2.2 Analyse

On observe que le dépôt 2 contient 280 unités de marchandises supplémentaires, correspondant à la demande au point de vente 6, puisque c'est le dépôt qui possède maintenant un coup de transport vers le point de vente 6 le moins onéreux.

Troisième partie

Exercice 3 : Un problème d'affectation optimale de trafic

Question 1

On a

$$Z_1(d_1) = d_1.C_1$$

On peut donc calculer

$$\frac{dZ_1}{\mathrm{d}d1}(d_1)$$

Or

$$\alpha_1^l + \beta_1^l.d_1 = \frac{dZ_1}{\mathrm{d}d1}(d_1)(d_1 - d_1^l) + Z_1(d_1^l)$$

Ainsi

$$\begin{cases} \alpha_1^l = -d_1^l.\frac{dZ_1}{\mathrm{d}d1}(d_1^l) + Z_1(d_1^l) \\ \beta_1^l = \frac{dZ_1}{\mathrm{d}d1}(d_1^l) \end{cases} \tag{5}$$

On obtient les résultats suivants :

l	d_1^l	$Z_1(d_1^l)$	$\frac{dZ_1}{\mathrm{d}d1}(d_1^l)$	α_1^l	β_1^l
0	0	0	60	0	60
1	1167	105000	120	-35000	120
2	2334	280000	180	-140000	180
3	3500	525000	240	-315000	240

En remplaçant l par h et 1 par 2 dans la formule (5) on obtient les résultats :

h	d_2^h	$Z_2(d_2^h)$	$\frac{dZ_2}{dd2}(d_2^h)$	α_2^h	β_2^h
0	0	0	33.4	0	33
1	389	18420	61	-5479	61
2	778	47964	91	-22633	90
3	1167	89320	122	-53550	122
4	1556	143828	159	-103558	160
5	1944	214485	207	-188816	207
6	2333	309671	293	-374356	293
7	2722	465050	582	-1119471	582
8	3111	1340370	11164	-33392592	11164
9	3500	174453	1214	-4076953	1215

Question 2

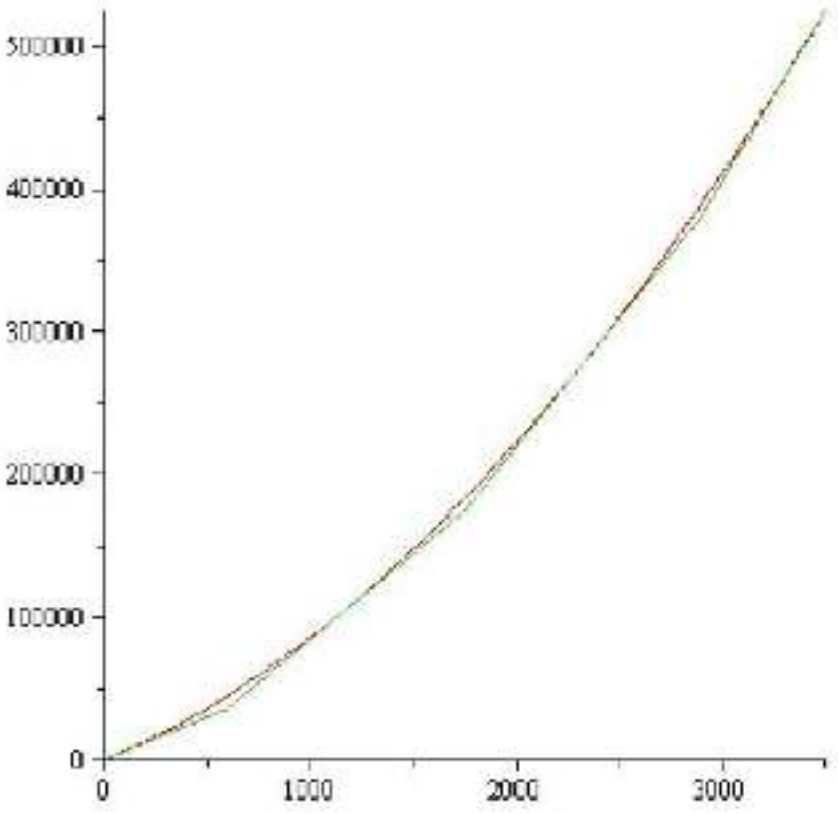

FIGURE 2 – Graphique des fonctions $d_1 \rightarrow Z_1(d_1)$ et $d_1 \rightarrow max_{l=0,\ldots,L}[\alpha_1^l + \beta_1^l.d_1]$

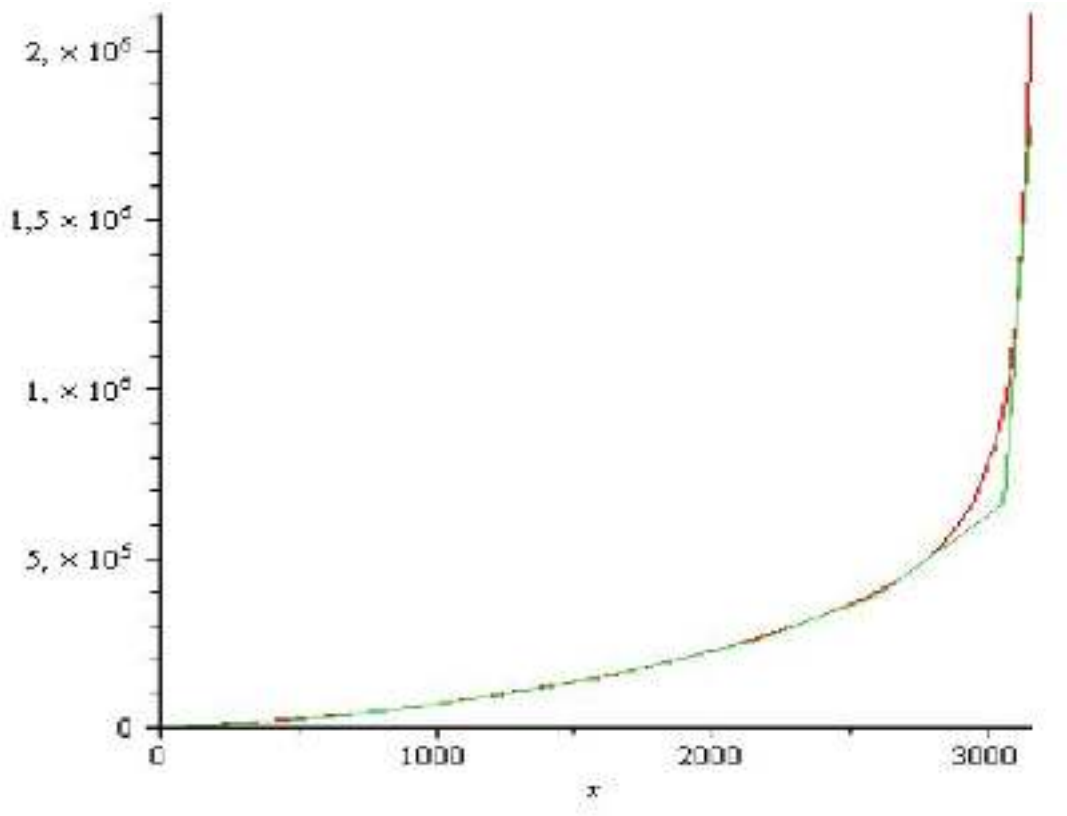

FIGURE 3 – Graphique des fonctions $d_2 \to Z_2(d_2)$ et $d_2 \to max_{h=0,\ldots,H}[\alpha_2^h + \beta_1^h.d_2]$

Question 3

Comme on a des termes linéaires, le maximum de la somme est égal à la somme des maximums. Ainsi (7) et (8) sont équivalentes

Question 4

On a d'après (8) :

$$Min\ z = max_{l=0,...,L;\ h=0,...,H}[\alpha_1^l + \alpha_2^h + \beta_1^l.d_1 + \beta_2^h.d_2$$

$$\left|\begin{array}{l} d_2 \leq D_{2,max} \\ d_1 + d_2 = D \\ d_1 \geq 0,\ d_2 \geq 0 \end{array}\right.$$

Or

$$Min\ z = max_{l=0,...,L;\ h=0,...,H}[\alpha_1^l + \alpha_2^h + \beta_1^l.d_1 + \beta_2^h.d_2$$

Donc si z est égal au maximum alors z est supérieur à chaque terme ainsi :

$$\left|\begin{array}{l} Min\ z \\ z \geq \alpha_1^l + \alpha_2^h + \beta_1^l.d_1 + \beta_2^h.d_2 \quad \forall l = 0, ..., L \quad \forall h = 0, ..., H \\ d_2 \leq D_{2,max} \\ d_1 + d_2 = D \\ d_1 \geq 0,\ d_2 \geq 0 \end{array}\right.$$

Question 5

On peut mettre le problème sous la forme canonique :

$$\left|\begin{array}{l} Max\ c.x \\ A.x = b \\ x \geq 0 \end{array}\right.$$

Avec :

$b = (D, D_{2,max})$ vecteur colonne de taille 2

$x = (d_1, d_2, 1, 1, r)$ vecteur colonne de taille 5 avec r une variable d'écart

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$ A matrice de taille 2*5

$$C = \begin{pmatrix} \beta_1^0 & \beta_2^0 & \alpha_1^0 & \alpha_2^0 & 0 \\ \beta_1^0 & \beta_2^1 & \alpha_1^0 & \alpha_2^1 & 0 \\ \beta_1^0 & \beta_2^2 & \alpha_1^0 & \alpha_2^2 & 0 \\ \beta_1^0 & \vdots & \alpha_1^0 & \vdots & 0 \\ \beta_1^0 & \beta_2^9 & \alpha_1^0 & \alpha_2^9 & 0 \\ \beta_1^1 & \beta_2^0 & \alpha_1^1 & \alpha_2^0 & 0 \\ \beta_1^1 & \vdots & \alpha_1^0 & \vdots & 0 \\ \beta_1^1 & \beta_2^9 & \alpha_1^1 & \alpha_2^9 & 0 \\ \beta_1^2 & \beta_2^0 & \alpha_1^2 & \alpha_2^0 & 0 \\ \beta_1^2 & \vdots & \alpha_1^2 & \vdots & 0 \\ \beta_1^2 & \beta_2^9 & \alpha_1^2 & \alpha_2^9 & 0 \\ \beta_1^3 & \beta_2^0 & \alpha_1^3 & \alpha_2^0 & 0 \\ \beta_1^3 & \vdots & \alpha_1^3 & \vdots & 0 \\ \beta_1^3 & \beta_2^9 & \alpha_1^3 & \alpha_2^9 & 0 \end{pmatrix} \qquad \text{C matrice de taille 40*5}$$

$$\forall i = 1, ..., 5 \qquad c_i = \sum_{j=1}^{40} -C_{i,j} \qquad \text{c est un vecteur ligne de taille 5}$$

Résolution par la programmation linéaire

A l'optimum on obtient $d_1 = 857$ et $d_2 = 2643$ On calcule ensuite :

$$C_1(d_1) = 60 + \frac{180}{7000}.d_1 = 82$$

et

$$C_2(d_2) = 25 + \frac{105}{3200}.d_1 + \frac{27000}{3200 - d_2} = 160$$

A l'optimum, on obtient un rapport d2/d1 fixé. Cependant, le deuxième trajet étant plus rapide, pour un usager individuel, il est plus intéressant de prendre le second trajet, au niveau du temps de parcours. Ainsi, si les usagers veulent satisfaire leurs intérêts propres, ils auront tendance à aller vers l'itinéraire 2, le rapport d2/d1 sera donc supérieur à la valeur nominale.

L'autorité devrait ainsi mettre en place des mesures pour dissuader les utilisateurs d'aller sur l'itinéraire 2. Elles peuvent être de natures incitatives, comme des messages sur des panneaux en bord de route ou à la radio lors de congestions. Par ailleurs, il est aussi possible de réduire les usagers sur le second trajet en interdisant l'accès à certains utilisateurs à certains horaires ou en mettant en place une forme de péage pour dissuader les utilisateurs d'aller sur le second trajet.

Quatrième partie

Annexes

A Codes utilisés

Pour une meilleur lisibilité, notre code est coloré syntaxiquement. Néanmoins, en contre-partie et pour des raisons d'encodage, nous n'avons pas pu mettre d'accent (é, è, à ...).

A.1 TP1

A.1.1 Résolution par la programmation linéaire

```
% Code pour scilab v5.3 utilisant la fonction karmarkar
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% On definit la matrice A par bloc grace a la fonction eye qui definit une matrice identite ,
     de la fonction zeros et ones qui definissent respectivement des matrices de zeros et de
     uns.
A=[[-1 0 0 0 0 0 0;1 -1 0 0 0 0 0;0 1 -1 0 0 0 0; 0 0 1 -1 0 0 0; 0 0 0 1 -1 0 0; 0 0 0 0 1
    -1 0 ; 0 0 0 0 0 1 -1] eye(7,7) zeros(7,7); zeros(7,7) eye(7,7) zeros(7,7)];
% On definit les vecteurs b et c
b=[100;120;100;90;120;110;115;140;100;110;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]
% On utilise la fonction karmarkar
[xopt,fopt,exitflag,iter,yopt]=karmarkar(A,b,c)

% Le resultat obtenu :
 yopt  =

   ineqlin: [0x0 constant]
   eqlin: [14x1 constant]
   lower: [21x1 constant]
   upper: [21x1 constant]
 iter  =

    167.
 exitflag  =

  - 2.
 fopt  =

    1.159D+09
 xopt  =

    3.481D+08
    53975.657
    1.0185276
    0.7142040
    6.7625149
    1.5719777
    13.179735
    1652.264
    1.4817562
    1.7270806
    1.815D+08
    2.5613712
    311.29936
    2.7313039
    1.
    1.
    1.
    1.
    1.
    1.
    1.
```

A.2 TP2

A.2.1 Résolution par la programmation linéaire

```
% Code pour scilab v5.3 utilisant la fonction karmarkar
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% On definit la matrice A par bloc grace a la fonction eye qui definit une matrice identite , de la fonction zeros et ones qui definissent respectivement des matrices de zeros et de uns.
A=[eye(8,8) eye(8,8) eye(8,8) zeros(8,3); eye(8,8) eye(8,8) eye(8,8) zeros(8,3); eye(8,8) eye(8,8) eye(8,8) zeros(8,3);  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]
% On definit les vecteurs b et c
b=[150;250;400;200;180;140;240;320;150;250;400;200;180;
      140;240;320;150;250;400;200;180;140;240;320;0;0;0]
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]
% On utilise la fonction karmarkar
[xopt,fopt,exitflag,iter,yopt]=karmarkar(A,b,c)

% Le resultat obtenu :
 yopt  =

   ineqlin: [0x0 constant]
   eqlin: [27x1 constant]
   lower: [27x1 constant]
   upper: [27x1 constant]
 iter  =

    70.
 exitflag  =

    1.
 fopt  =

    15500.129
 xopt  =

    149.99365
    249.98844
    0.0037950
    0.0015180
    179.99715
    139.99635
    239.99697
    0.0018975
    0.0044573
    0.0107994
    0.0008721
    0.0013309
    0.0013309
    0.0028082
    0.0011324
    0.0028082
    0.0018975
    0.0007590
    399.99533
    199.99715
    0.0015180
    0.0008433
    0.0018975
    319.99529
    959.97977
    0.0255393
    919.99469
```

A.2.2 Complément 1

```
% Code pour scilab v5.3 utilisant la fonction karmarkar
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% On definit la matrice A par bloc grace a la fonction eye qui definit une matrice identite,
      de la fonction zeros et ones qui definissent respectivement des matrices de zeros et de
      uns.
A=[eye(8,8) eye(8,8) eye(8,8) zeros(8,3); eye(8,8) eye(8,8) eye(8,8) zeros(8,3); eye(8,8)
     eye(8,8) eye(8,8) zeros(8,3);  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]
% On definit les vecteurs b et c (b6=280)
b=[150;250;400;200;180;2800;240;320;150;250;400;200;180;2800;240;320;150;250;400;200;
      180;2800;240;320;0;0;0]
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]
% On utilise la fonction karmarkar
[xopt,fopt,exitflag,iter,yopt]=karmarkar(A,b,c)

% Le resultat obtenu :
yopt  =

   ineqlin: [0x0 constant]
   eqlin: [27x1 constant]
   lower: [27x1 constant]
   upper: [27x1 constant]
 iter  =

    70.
 exitflag  =

    1.
 fopt  =

    16200.129
 xopt  =

    149.99364
    249.98843
    0.0037981
    0.0015193
    179.99715
    279.99635
    239.99697
    0.0018991
    0.0044610
    0.0108084
    0.0008729
    0.0013320
    0.0013320
    0.0028105
    0.0011333
    0.0028105
    0.0018991
    0.0007596
    399.99533
    199.99715
    0.0015193
    0.0008440
    0.0018991
    319.99529
    1099.9798
    0.0255605
    919.99469
```

A.2.3 Complément 2

```
% Code pour scilab v5.3 utilisant la fonction karmarkar
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% On definit la matrice A par bloc grace a la fonction eye qui definit une matrice identite , de la fonction zeros et ones qui definissent respectivement des matrices de zeros et de uns.
A=[eye(8,8) eye(8,8) eye(8,8) zeros(8,3); eye(8,8) eye(8,8) eye(8,8) zeros(8,3); eye(8,8) eye(8,8) eye(8,8) zeros(8,3);  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]
% On definit les vecteurs b et c (C 2,6=4)
b=[150;250;400;200;180;2800;240;320;150;250;400;200;180;2800;240;320;150;250;400;200;
    180;2800;240;320;0;0;0]
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 karmarkar
[xopt,fopt,exitflag,iter,yopt]=karmarkar(A,b,c)

% Le resultat obtenu :
 yopt  =

   ineqlin: [0x0 constant]
   eqlin: [27x1 constant]
   lower: [27x1 constant]
   upper: [27x1 constant]
 iter  =

    70.
 exitflag  =

    1.
 fopt  =

    15920.123
 xopt  =

    149.99425
    249.99156
    0.0038343
    0.0015337
    179.99719
    0.0076687
    239.99699
    0.0019172
    0.0038343
    0.0076687
    0.0008521
    0.0012781
    0.0012781
    279.99156
    0.0010955
    0.0025562
    0.0019172
    0.0007669
    399.99531
    199.99719
    0.0015337
    0.0007669
    0.0019172
    319.99553
    819.99494
    280.01013
    919.99493
```

A.3 TP3

A.3.1 Graphique

```
x=0:1:3500;
y=x.*(60+(x.*180)/7000);
z=
xset("window",1);
plot2d(x,y);plot2d(x,z)
```

A.3.2 Résolution par la programmation linéaire

```
% Code pour scilab v5.3 utilisant la fonction karmarkar
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% On definit la matrice A.
A=[1 1 0 0 0 ; 0 1 0 0 1]
% On definit le vecteur b.
b=[3500; 3200]
% On definit la matrice c par deux matrices sinon on a une commande qui depasse 512 caracteres, la limite de scilab.
c1=[60 33 0 0 ; 60 61 0 -5479 ; 60 90 0 -22633 ; 60 122 0 -53550 ; 60 160 0 -103558 ; 60 207 0 -188816 ; 60 293 0 -374356 ; 60 582 0 -1119471 ; 60 1164 0 -33392592 ; 60 1215 0 -4076953 ; 120 33 -35000 0 ; 120 61 -35000 -5479 ; 120 90 -35000 -22633 ; 120 122 -35000 -53550 ; 120 1120 -35000 -103558 ; 120 207 -35000 -188816 ; 120 293 -35000 -374356 ; 120 582 -35000 -1119471 ; 120 1164 -35000 -33392592 ; 120 1215 -35000 -4076953 ; 180 33 -140000 0 ; 180 61 -140000 -5479 ; 180 90 -140000 -22633 ]

c2=[180 122 -140000 -53550 ; 180 1180 -140000 -103558 ; 180 207 -140000 -188816 ; 180 293 -140000 -374356 ; 180 582 -140000 -1119471 ; 180 1164 -140000 -33392592 ; 180 1215 -140000 -4076953 ; 240 33 -1315000 0 ; 240 61 -1315000 -5479 ; 240 90 -1315000 -22633 ; 240 122 -1315000 -53550 ; 240 1240 -1315000 -103558 ; 240 207 -1315000 -188816 ; 240 293 -1315000 -374356 ; 240 582 -1315000 -1119471 ; 240 1164 -1315000 -33392592 ; 240 1215 -1315000 -4076953 ]
 % On definit la matrice C
 c3=[c1 zeros(23,1);c2 zeros(17,1)]
% On definit le vecteur c comme la somme des colonnes de C
c=c3'*[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]'
% On utilise la fonction karmarkar
[xopt,fopt,exitflag,iter,yopt]=karmarkar(A,b,c)

% Le resultat obtenu :
yopt =

   ineqlin: [0x0 constant]
   eqlin: [0x0 constant]
   lower: [5x1 constant]
   upper: [0x0 constant]
 iter =

    52.
 exitflag =

    1.
 fopt =

  - 1.175D+08
 xopt =

    857.3665
    2642.6335
    1.
    1.
    557.3665
```