Méthodes de la recherche opérationnelle - Projet

Aurélie Charlou, Camille Lapoublade, Jean-Marie Roussel

3 mai 2015

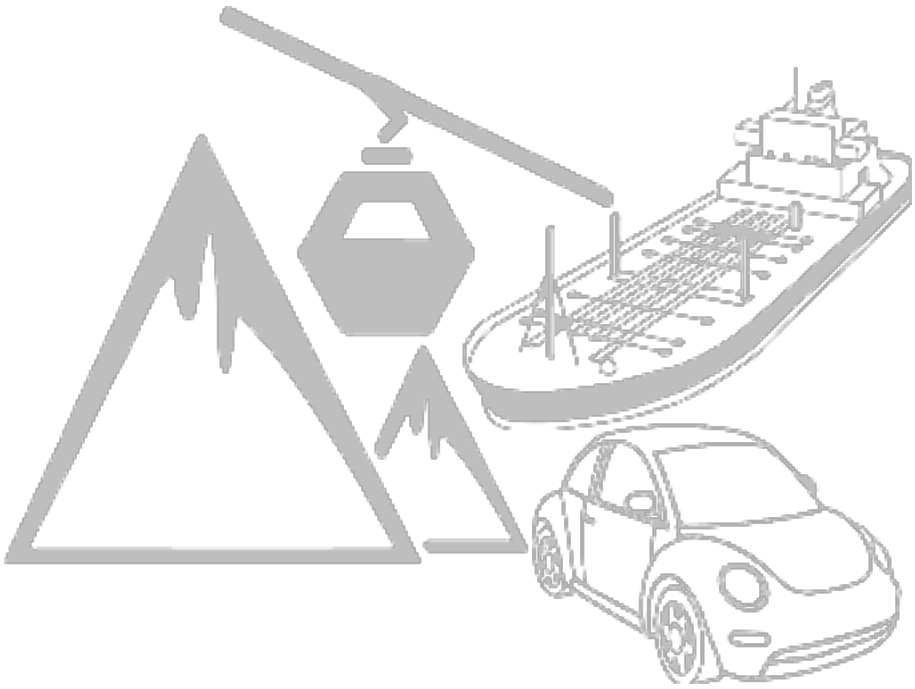

Table des matières

Introduction

La méthode de la recherche opérationnelle a pour but de chercher les solutions optimales afin de réaliser des travaux aux plus bas coûts par exemple. Ce projet est l'occasion pour nous de mobiliser nos connaissances en la matière au travers de trois exercices d'optimisation. Ce projet est par ailleurs l'occasion de résoudre des problèmes d'optimisation dynamique ou linéaire de façon informatique. On utilise le logiciel open source Scilab. Les scripts utilisés, ainsi que les résultats numériques obtenus, sont consignés en annexes.

I. Un problème de chargement de type sac à dos

1. Présentation du problème

Ce problème de chargement vise à trouver la méthode optimale afin de charger un cargo de 995 tonnes. Comme la méthode visant à considérer l'attractivité respective des marchandises n'est pas la meilleure, nous cherchons alors à appréhender le problème d'une manière différente.

2. Formalisation du problème

Montrons que le problème peut être formalisé de deux façons :

(K)

$$\min_{h_i, i=1:8} \quad \sum_{i=0}^{I} c_i x_i$$

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

(D)

$$\min_{h_i, i=1:8} \quad \sum_{i=0}^{I} c_i x_i$$

$$\left| \begin{array}{l} y_0 \in \{0, 1, ..., b\} \\ y_4 = b \\ y_i = y_{i-1} + a_i x_i \\ x_i \in N \end{array} \right.$$

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 :

$$(1) \qquad V_i(y_i) = max_{\begin{pmatrix} x_i \in N \\ x_i \leq y_i/a_i \end{pmatrix}} [V_{i-1}(y_i - a_i x_i) + c_i x_i]$$

$$\text{Initialisée par } V_0(y_0) = 0, \forall y_0 = 0, 1, ..., b$$

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. 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 a $x_i \leq y_i/a_i$ ce qui est équivalent à $y_i - a_i x_i$ est positif. Cette écriture est la conséquence de $y_i - a_i x_i = y_{i-1}$ et comme y_{i-1} est un chargement il se doit d'être positif. L'initialisation se justifie car au début le cargo est vide donc $V_0(y_0) = 0$ et $y_0 \in \{0, 1 \ldots, b\}$ d'après l'écriture de (D).

3. Résolution

a. Principe de résolution

On commence par créer une matrice M d'entiers de dimensions nombre d'objets*capacité + 1.

Chaque case $M_{i,j}$ de cette matrice représente le bénéfice maximal possible pour les i premiers objets avec un poids j.

On initialise avec le premier objet 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. 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. Et ainsi de suite...

b. Résultats - cas de 4 marchandises

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

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

On remarque alors que $\sum_{i=1}^{4} x_i a_i = 995$ donc le cargo est complet. L'ordinateur n'a pas eu besoin d'ajouter du vide pour atteindre la capacité totale.

c. Résultats - cas de 5 marchandises

i. Pour b=1800

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

ii. Pour b=1854

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

iii. Pour b=1858

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

II. Profil en long d'un système de transports guidés

1. Présentation du problème

Nous allons aborder ce problème de deux manières différentes. Dans un premier temps nous allons traiter le problème par la programmation dynamique, ensuite nous l'envisagerons par la programmation linéaire.

2. Résolution et résultats

a. Résolution par programmation dynamique

On considère le problème (2) :

$$\min_{h_i, i=1:8} \quad \sum_{i=1}^{8} \Gamma_i(h_i)$$

$$\left| \begin{array}{l} h_0 = 0 \\ h_i \geq 0 \ \forall i = 0, ..., 8 \\ |h_{i+1} - h_i| \ \leq 10 \ \forall i = 1, ..., 7 \end{array} \right.$$

Qui peut aussi s'écrire (4) :

$$\min_{l_i, i=1:8} \quad \sum_{i=1}^{8} \Gamma_i(10 l_i)$$

$$\left| \begin{array}{l} l_0 = 0 \\ l_i \in N \\ |l_{i+1} - l_i| \leq 1 \ \forall i = 1, ..., 7 \end{array} \right.$$

Nous remarquons que pour obtenir le problème sous la forme (2), il suffit de remplacer $10l_i$ par h_i dans le problème (4) car d'après l'équation (3), $h_i = 10l_i$

La formule de Hamilton-Jacobi-Bellman est donnée :

$$V_{i+1}(l_{i+1}) = \Gamma_{i+1}(10l_{i+1}) + \min_{(l_i \in N / |l_{i+1} - l_i| \leq 1)} [V_i(l_i)]$$

Le terme $V_{i+1}(h_{i+1})$ représente le coût des travaux et ce coût est égal au coût de l'étape $i+1$ plus le coût pour arriver à cette étape. Et comme on cherche à minimiser le coût des travaux, nous cherchons le minimum des coûts pour arriver à l'étape i. Comme il était possible de résoudre ce problème « à la main », nous avons choisi le faire. Nous avons alors obtenu le graphe suivant.

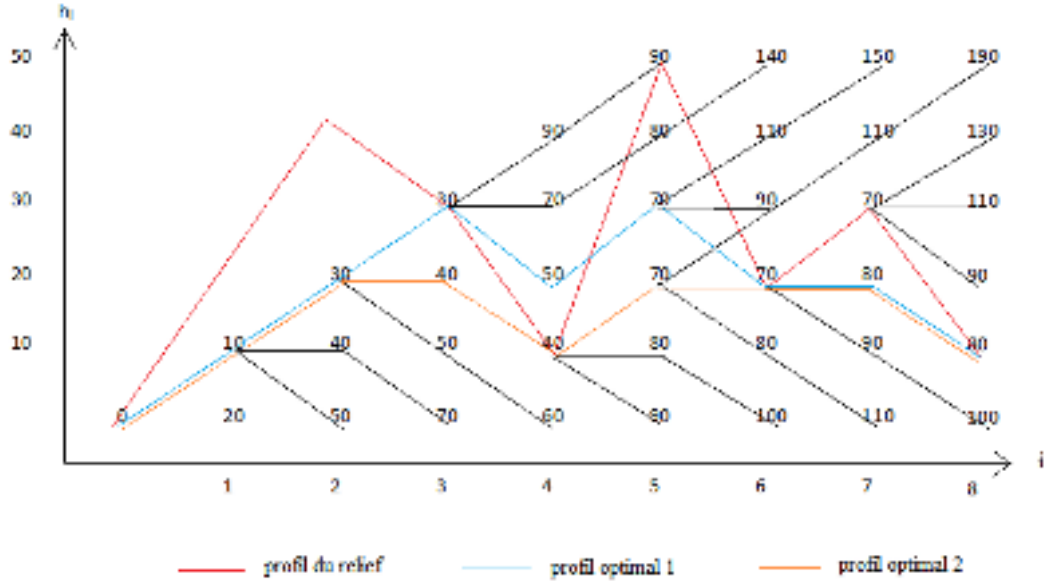

FIGURE 1 : Détermination du profil optimal par la formule de Hamilton-Jacobi-Bellman
Les états accessibles sont en noir

b. Résolution par programmation linéaire

Question 1

Une nouvelle fois, montrer l'équivalence est un jeu de formule. En effet, le coût $\Gamma_i(h_i)$ est donné par :

$$\Gamma_i(h_i) = C_d(k_i - h_i) \text{ si } k_i \geq h_i$$
$$\Gamma_i(h_i) = C_r(h_i - k_i) \text{ si } h_i \geq k_i$$

Ainsi, minimiser le coût $\Gamma_i(h_i)$ équivaut à introduire une nouvelle variable notée g_i telle que l'on ait :

$$g_i \geq C_d(k_i - h_i)$$
$$g_i \geq C_r(-k_i + h_i)$$

De même, en faisant appel aux propriétés de la valeur absolue, nous remarquons que les deux écritures sont équivalentes.

$$|h_{i+1} - h_i| \leq 10 \quad \Leftrightarrow \quad -10 \leq h_{i+1} - h_i \leq 10$$

Question 2

En réutilisant ce qui précède et en détaillant les écritures des contraintes imposées, on obtient bien la formulation (6).

- Le coût Γ_i devient g_i et pour minimiser celui-ci, les équations précédentes (celles juste au-dessus) sont des contraintes que l'on se doit de respecter afin de résoudre le problème
- La condition sur la valeur absolue de $h_{i+1} - h_i$ inférieure à 10 est reformulée selon ce que nous avons vu à la question 1.

Comme h_0 n'est pas une inconnue, cette donnée n'apparaît pas dans les contraintes à respecter.

Question 3

Introduction de variables d'écart positives : $\alpha_i, \beta_i, \gamma_i, \delta_i$

$$\max_{g_i, h_i, i=1,\dots,8} \quad -\sum_{i=1}^{8} g_i$$

$$\left| \begin{array}{l} g_i + C_d h_i - \alpha_i = C_d k_i \\ g_i - C_r h_i - \beta_i = -C_r k_i \\ h_i - h_{i_1} + \gamma_i = 10 \\ h_i - h_{i-1} - \delta_i = -10 \\ g_i, h_i, \alpha_i, \beta_i, \gamma_i, \delta_i \geq 0 \end{array} \right.$$

Le problème se met alors sous forme canonique de la façon suivante :

$$\max \ cx$$

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

avec :

$$c = \left(\begin{array}{cccccccc|ccc} -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & \dots & 0 \end{array} \right)_{(1,48)}$$

$$b = \begin{pmatrix} C_d k_1 \\ \vdots \\ C_d k_8 \\ \hline -C_r k_1 \\ \vdots \\ -C_r k_8 \\ \hline 10 \\ \vdots \\ 10 \\ \hline -10 \\ \vdots \\ -10 \end{pmatrix}_{(32,1)} \quad , \quad x = \begin{pmatrix} g_1 \\ \vdots \\ g_8 \\ \hline h_1 \\ \vdots \\ h_8 \\ \hline \alpha_1 \\ \vdots \\ \alpha_8 \\ \hline \beta_1 \\ \vdots \\ \beta_8 \\ \hline \gamma_1 \\ \vdots \\ \gamma_8 \\ \hline \delta_1 \\ \vdots \\ \delta_8 \end{pmatrix}_{(48,1)}$$

et $A = \left(\begin{array}{cccccc} I & I & -I & 0 & 0 & 0 \\ \hline I & -2I & 0 & -I & 0 & 0 \\ \hline 0 & B & 0 & 0 & I & 0 \\ \hline 0 & B & 0 & 0 & 0 & -I \end{array}\right)_{(32,48)}$ avec $B = \begin{pmatrix} 1 & 0 & \dots & \dots & 0 \\ -1 & \ddots & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & -1 & 1 \end{pmatrix}_{(8,8)}$

et I la matrice identité (8,8)

La résolution par programmation linéaire nous donne :

i	0	1	2	3	4	5	6	7	8
h_i	0	10	20	26	16	26	20	20	10

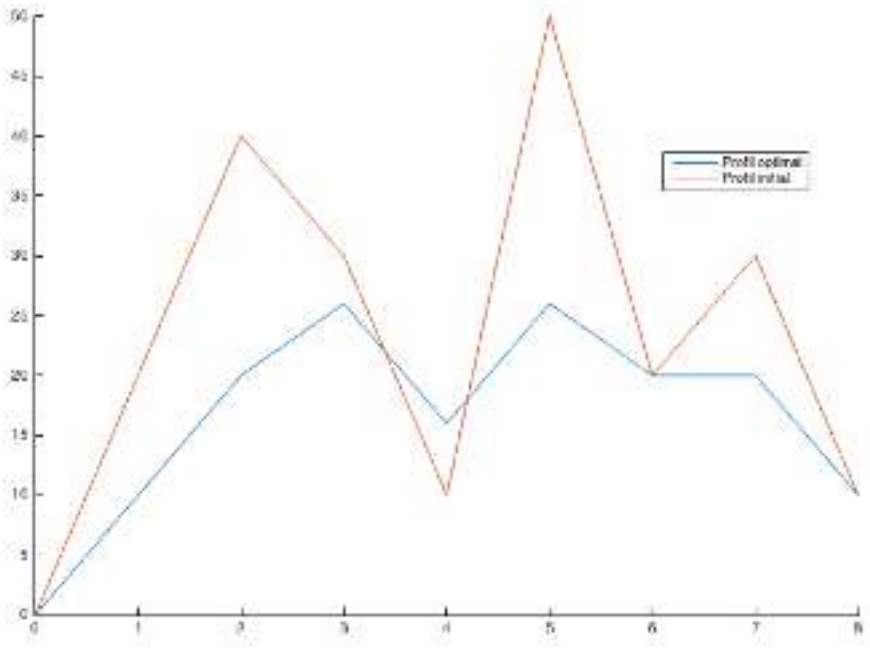

FIGURE 2 : Tracé optimal obtenu par programmation linéaire

c. Comparaison des deux méthodes

Nous remarquons que les résultats que nous avons obtenu pour les h_i ne sont pas des multiples de 10. En effet, nous n'avons pas fait intervenir cette condition lorsque nous avons élaboré le programme. Toutefois, lorsque nous avons résolu l'équation de Hamilton-Jacobi-Bellman « à la main », deux chemins permettaient d'obtenir un coût minimum. Lors de la troisième itération, la résolution de l'équation de Hamilton-Jacobi-Bellman donne $h_i = 20$ ou $h_i = 30$ alors que l'algorithme du simplexe donne $h_i = 26$. Ainsi, nous remarquons que 26 appartient à l'intervalle $[20; 30]$ et que 26 est la valeur optimale pour obtenir un coût minimal. Nous pouvons mener la même réflexion aux itérations 4 et 5.

III. Contrôle d'accès

1. Présentation du problème

Nous considérons ici une voie d'insertion sur une autoroute. Sur cette voie d'insertion est disposé un feu tricolore afin de réguler l'entrée des véhicules sur l'autoroute. Pour contrôler l'accès à cette voie et gérer le flux de voitures de manière idéale, nous devons prendre en compte la demande, l'offre et le débit du convergent.

2. Formalisation du problème

Question 1

Afin de rendre linéaire ce problème et pour pouvoir utiliser l'algorithme du simplexe, nous introduisons une variable y_i à laquelle nous imposons deux contraintes :

$$y_i \geq s_{ref} - s_{i+1} \text{ et } y_i \geq 0$$

Nous cherchons à montrer que ces contraintes satisfont toujours à $y_i = |s_{ref} - s_i|^+$ c'est-à-dire que nous souhaitons démontrer l'équivalence suivante

$$s_{ref} - s_{i+1} \geq |s_{ref} - s_i|^+$$

Distinguons deux cas :

- **Premier cas :** On suppose $s_{ref} \geq s_i$ ce qui implique que $|s_{ref} - s_i|^+ = s_{ref} - s_i$
D'où $s_{ref} - s_{i+1} = s_{ref} - s_i - d_i - q_i$
Cela revient à montrer que $q_i - d_i \geq 0$

- **Second cas :** $s_{ref} \leq s_i$ ce qui implique $|s_{ref} - s_i|^+ = 0$
D'où $s_{ref} - s_{i+1} \geq 0$
Qui est équivalent à $s_{ref} - s_i - d_i - q_i \geq 0$ or $s_{ref} - s_i = 0$
Cela revient à montrer que $q_i - d_i \geq 0$

Dans les deux cas, nous devons démontrer que $q_i - d_i \geq 0$ mais nous n'y sommes pas parvenus.

3. Résolution

Question 2

Introduction de variables d'écart positives : α_i, β_i

$$\max \quad -\sum_{i=1}^{I}(2s_i + 5y_i)$$

$$\left| \begin{array}{l} s_0 = 25 \\ s_{i+1} - s_i + q_i = d_i \\ q_i + \alpha_i = O_i \\ y_i + s_{i+1} + \beta_i = s_{ref} \\ y_i, s_i, \alpha_i, \beta_i \geq 0 \end{array} \right.$$

Question 3

Le problème se met alors sous forme canonique de la façon suivante :

$$\max \ cx$$

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

$$c = \Big(\ 0 \ \big| \ -2 \ \ \ldots \ \ -2 \ \big| \ -5 \ \ \ldots \ \ -5 \ \big| \ 0 \ \ \ldots \ \ 0 \ \Big)_{(1,41)}$$

$$\text{b=}\begin{pmatrix} s_0 \\ \hline d_1 \\ \vdots \\ d_7 \\ \hline O_1 \\ \vdots \\ O_7 \\ \hline s_{ref} \\ \vdots \\ s_{ref} \end{pmatrix}_{(25,1)} \quad , \quad x = \begin{pmatrix} s_0 \\ \vdots \\ s_8 \\ \hline y_0 \\ \vdots \\ y_7 \\ \hline q_0 \\ \vdots \\ q_7 \\ \hline \alpha_0 \\ \vdots \\ \alpha_7 \\ \hline \beta_0 \\ \vdots \\ \beta_7 \end{pmatrix}_{(41,1)}$$

$$\text{et } A = \begin{pmatrix} 1 & 0 & \dots & \dots & 0 \\ \hline B_1 & 0_{8,8} & I_8 & 0_{8,8} & 0_{8,8} \\ 0_{8,9} & 0_{8,8} & I_8 & I_8 & 0_{8,8} \\ B_2 & I_8 & 0_{8,8} & 0_{8,8} & -I_8 \end{pmatrix}_{(25,41)}$$

$$\text{avec } B_1 = \begin{pmatrix} -1 & 1 & 0 & \dots & \dots & 0 \\ 0 & \ddots & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & \dots & 0 & -1 & 1 \end{pmatrix}_{(8,9)} \quad \text{et } B_2 = \begin{pmatrix} 0 & 1 & 0 & \dots & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & \ddots & 0 \\ 0 & \dots & \dots & \dots & 0 & 1 \end{pmatrix}_{(8,9)}$$

La résolution par programmation linéaire nous donne :

i	0	1	2	3	4	5	6	7	8
s_i	25	19	15	10	11	19	34	46	41

Conclusion

Ce projet nous a permis d'appliquer les concepts étudiés en cours à des cas concrets. Nous nous sommes intéressés à trois problèmes. Le premier consistait à déterminer la composition optimale d'un chargement. Le deuxième proposait de construire une route le long d'un relief en minimisant les coûts de travaux. Quant au dernier, il s'agissait de gérer un flux de voitures sur une voie d'insertion d'autoroute munie d'un feu tricolore. Pour cela, nous avons formalisé, modélisé ces problèmes et utilisé deux outils vus en cours afin de les résoudre, à savoir Hamilton-Jacobi-Bellman et la programmation linéaire. Qui plus est, grâce au logiciel de programmation Scilab, nous avons pu traiter les applications informatiques.

Annexes

Exercice 1

Programmation dynamique sous SCILAB en utilisant la formule de Hamilton-Jacobi-Bellman.

```
function x=main(a,b,c,n)
%multiplication des stocks (en espÈrant qu'elle soit suffisante)
a=multi(a,n)
c=multi(c,n)

%remplissage du sac ‡ dos
O=hjb3(a,b,c)

%rÈduction des numÈros des produits obtenus
O=floor(O./n)

%comptage des quantitÈs
x(length(a)/n)=0%matrice des quantitÈs
for i=1:length(O)
    for k=1:length(a)
        if O(i)==k then
            x(k)=x(k)+1
        end
    end
end

end
```

```
function a2=multi(a,n)
a2=[]
for i=0:length(a)-1
    a2(i*n+1:(i+1)*n)=a(i+1)
end
end
```

```
function O=hjb3(a,b,c)
%RÈsolution d'un problËme type sac ‡ dos
%a est le tableau des poids
%c est le tableau des valeurs
%b est la contenance du sac ‡ dos

%ETABLISSEMENT D'UN TABLEAU CAPACITE/VALEURS CUMULEES
M(1,b+1)=0

%initialisation - rÈalisation de la premiËre ligne
j=1
while j<=b+1
    if a(1)>=j then
        M(1,j)=0
    else
        M(1,j)=c(1)
    end
    j=j+1
end

%lignes suivantes
for i=2:length(a)
    for j=1:b+1
        if a(i)>=j then
            M(i,j)=M(i-1,j)
        else
            M(i,j)=max(M(i-1,j),M(i-1,j-a(i))+c(i))
        end
    end
end

disp(M)

%RECHERCHE DU CHEMIN OPTIMAL DANS LE TABLEAU ETABLI

O=[]%O sera le chemin (=liste des marchandises) optimal
%initialisation - on rÈcupËre dans la derniËre ligne le poids minimal pour
%faire un bÈnÈfice maximal

j=size(M,2)
while M(size(M,1),j)==M(i,j-1)
    j=j-1
end


%itÈration
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 then
        O=[O i]
    end
    i=i-1
end
end
```

Exercice 2

Résolution par programmation linéaire sous SCILAB grâce à la fonction *karmarkar*.

```
%EXERCICE 2 : profil en long d un systeme de transports guides

%definition d une matrice identite 8x8
I=eye(8,8)

%definition de la matrice B
function B=matriceA2()
for i=1:8
    A2(i,i)=1
    if i<8
        A2(i+1,i)=-1
    end
end
endfunction

%definition de la matrice b
b=[20.0;40.0;30.0;10.0;50.0;20.0;30.0;10.0;-40.0;-80.0;-60.0;-20.0;-100.0;
    -40.0;-60.0;-20.0;10.0;10.0;10.0;10.0;10.0;10.0;10.0;10.0;-10.0;-10.0;
    -10.0;-10.0;-10.0;-10.0;-10.0;-10.0]

%definition de la matrice c
%ici c est positive car la fonction karmarkar optimise deja dans le sens
%d une minimisation
c=[1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;
    0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;
    0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0]

%definition d une matrice vide
V(8,8)=0

%definition de la matrice A
A=[I I -I V V V; I -2*I V -I V V; V B V V I V; V B V V V -I]

%optimisation par l'algorithme de karmarkar
[xopt,fopt,exitflag,iter,yopt] = karmarkar(A,b,c)

%resultat de l optimisation
yopt  =

ineqlin: [0x0 constant]
eqlin: [32x1 constant]
lower: [48x1 constant]
upper: [48x1 constant]
```

```
iter  =
71.
exitflag  =
1.
fopt  =
80.
xopt  =
%liste des gi
10.
20.
3.8
12.
24.
0.0
10.
0.0
%liste des hi
10.
20.
26.
16.
26.
20.
20.
10.
%liste des alphai
0.0
0.0
0.0
19.
0.0
0.0
0.0
0.0
%liste des betai
30.
60.
11.
0.0
71.
0.0
30.
0.0
%liste des gammai
0.0
0.0
3.8
20.
```

```
0.0
16.
10.
20.
%liste des deltai
20.
20.
16.
0.0
20.
3.8
10.
0.0
```

Exercice 3

Résolution par programmation linéaire sous SCILAB grâce à la fonction *karmarkar*.

```
%EXERCICE 3 : controle d acces

%definition des matrices
function B3=matriceB3()
for i=1:8
    B3(i,i)=-1
    B3(i,i+1)=1
end
endfunction

function B4=matriceB4()
for i=1:8
    B4(i,i+1)=1
end
endfunction

V(8,8)=0
V2(8,9)=0

A=[[1 zeros(40,1)]; B3 V I V V; V2 V I I V; B4 I V V -I]

b=[25 9 5 10 15 20 25 20 15 15 22 15 14 12 10 8 20 15 15 15 15 15 15 15 15]


%c est definie positive car la fonction karmarkar resout les problemes
lineaires en minimisant cx, c(1)=0 car s0 n intervient pas
dans la somme minimisee
c=[0 2 2 2 2 2 2 2 2 5 5 5 5 5 5 5 5]
c(41)=0


%resolution par la fonction karmarkar
[xopt,fopt,exitflag,iter,yopt] = karmarkar(A,b,c)

%resultat
yopt  =

ineqlin: [0x0 constant]
eqlin: [25x1 constant]
lower: [41x1 constant]
upper: [41x1 constant]
iter  =
90.
```

```
exitflag  =
1.%cela signifie que l algorithme converge
fopt  =
435.
xopt  =
%liste des si (9)
25.
19.
15.
10.
11.
19.
34.
46.
41.
%liste des yi (8)
0.0
0.0
5.
4.
0.0
0.0
0.0
0.0
%liste des qi (8)
15.
9.
15.
14.
12.
10.
8.
20.
%liste des alphai (8)
0.0
13.
0.0
0.0
0.0
0.0
0.0
0.0
%liste des betai (8)
4.
0.0
0.0
0.0
4.
```

```
19.
31.
26.
```