RO – TP 3 : programmation linéaire

Stéphane Canu

Décembre 2016, ASI, INSA Rouen

Le but du TP est de résoudre un programme linéaire en Matlab et de comparer différents logiciels permettant de la faire.

Ex. 1 — programmation linéaire

1. Afin d'assurer une bonne santé de cobayes, il faut les nourrir en leur donnant un minimum de 24 grammes de lipides, 36 grammes de glucides et 4 grammes de protéines par jour. Il ne faut pas leur donner plus de 5 onces de nourriture. Nous disposions de deux sources d'alimentation. Les croquettes royal cobaye qui contiennent 8 grammes de lipides, 12 grammes de glucides et 2 grammes de protéines par once et qui coutent 3 euros les 10 onces, et les boulettes « vite se casse » qui contiennent 12 grammes de lipides, 12 grammes de glucides et 1 grammes de protéines par once et qui coutent 2 euros les 10 onces.

   a) Calculez, à l'aide de CVX le mélange le moins cher à donner à nos cobayes

```
cvx_begin
    variables x_1 x_2
    dual variables lip glu pro die p1 p2;
    minimize( 30*x_1 + 20*x_2 )
    subject to
        lip : 8 *x_1 + 12* x_2 >= 24;
        glu : 12 *x_1 + 12 *x_2 >= 36;
        pro : 2 *x_1 + x_2 >= 4;
        die : x_1 + x_2 <= 5;
        p1 : x_1 >=0;
        p2 : x_2 >=0;
cvx_end
```

   b) reformulez le problème comme un programme linéaire matriciel sous le forme suivante :

$$\begin{cases} \min\limits_{\mathrm{x}\in\mathbf{R}^m} & \mathrm{f}^\top \mathrm{z} \\ \text{avec} & A\mathrm{x} = \mathrm{b} \\ \text{et} & \mathrm{x} \geq 0 \end{cases}$$

```
f = [30 ; 20];
b = [24;36;4;-5];
A = [8 12 ; 12 12 ; 2 1 ; -1 -1];
```

   c) Ecrire un programme CVX permettant de résoudre le programme linéaire standard

```
cvx_begin
    variable x(2)
    dual variables a p;
    minimize( f'*x )
    subject to
        a : A*x >= b
        p : x >= 0;
cvx_end
```

d) Ecrire un programme GLPK permettant de résoudre le programme linéaire standard[1]

```
ctype = 'UUUU'; % contraintes dé'égalites
vartype = 'CC'; % variables continues
s = 1;          % 1 minimisation (-1 -> maximisation)

param.msglev = 1;
param.itlim = 100;

[xmin_g,fmin,status,extra]=glpk(f,-A,-b,0*f, [], ctype, vartype, s, param);
```

e) Ecrire un programme permettant de résoudre le programme linéaire standard à l'aide de linprog

```
[xl c e o p] = linprog(f,-A,-b,[],[],0*f);
```

f) vérifiez que les différents programmes donnent la même solution. Quel est le plus efficace et le plus facile à développer ?

```
[[x_1;x_2] x xl xmin_g]
```

2. Le gérant de la distribution de l'entreprise Cannes United (production de cannes pour aveugles), dispose de trois usines de fabrication (une à Amfreville la campagne (A), une à Bogota (B) et une dernière à Canberra (C)). Il souhaite minimiser ses couts de transports entre les sites de fabrication et ses quatre magasins (à Winipeg (W), Xanadu (X), Yerville (Y) et Zanzibar (Z)).

Usine	Production
A	400
B	1500
C	900
Total	2800

Magasins	Demande
W	700
X	600
Y	1000
Z	500
Total	2800

.

Nous avons aussi la matrice des couts de transports :

	W	X	Y	Z
A	20	40	70	50
B	100	60	90	80
C	10	110	30	200

Par exemple il en coute 70 euros de transporter à Yerville une canne fabriquée à Amfreville la campagne.

a) Reformulez le problème comme un programme linéaire et proposer un programme CVX permettant de le résoudre

```
a = [400; 1500; 900];
b = [700; 600; 1000; 500];
C = [20 40 70 50
    100 60 90 80
    10 110 30 200];
[n,p] = size(C);

cvx_begin
    variable X(n,p)
    dual variables D A pos;
    minimize( sum(sum(C.*X)) )
    subject to
        D : sum(X) == b';
        A : sum(X') == a';
        pos : X >= 0;
cvx_end
```

[1] https://www.gnu.org/software/glpk/

b) Reformulez le problème comme un programme linéaire standard et proposer un programme CVX permettant de le résoudre

```
e = ones(1,n);
z = zeros(1,n);
I = eye(n);
M = [e z z z; z e z z; z z e z; z z z e ;
repmat(I,1,p)];
Cl = reshape(C,1,n*p);
bb = [b ; a];

cvx_begin
    variable x(n*p)
    dual variables D pos;
    minimize( Cl*x )
    subject to
        D : M*x == bb;
        pos : x >= 0;
cvx_end
```

c) Ecrire un programme permettant de résoudre le programme linéaire standard à l'aide de linprog

```
[xl c e o p] = linprog(Cl',[],[],M,bb,0*Cl);
```

d) Ecrire un programme permettant de résoudre le programme linéaire standard à l'aide de GUROBI[2]

```
clear model;
model.obj = Cl;
model.A = sparse(M);
model.sense = ['='; '='; '='; '='; '='; '='; '='];
model.rhs = bb;
model.lb = zeros(size(x));
clear params;
params.Presolve = 2;
params.TimeLimit = 100;

result = gurobi(model, params);
disp(result.objval);
```

e) Ecrire un programme permettant de résoudre le programme linéaire standard à l'aide de CPLEX[3]

```
[xcl c e o p] = cplexlp(Cl',[],[],M,bb,0*Cl);
```

f) Ecrire un programme GLPK permettant de résoudre le programme linéaire standard[4]

```
ctype = 'SSSSSSS';        % contraintes d ' éégalites
vartype = 'CCCCCCCCCCCC'; % variables continues
s = 1;                    % 1 minimisation (-1 -> maximisation)
param.msglev = 1;
param.itlim = 100;

[xmin_g,fmin,status,extra]=glpk(Cl',M,bb,0*Cl, [], ctype, vartype, s, param);
```

g) vérifiez que les différents programmes donnent la même solution. Quel est le plus efficace et le plus facile à développer ?

```
[xl result.x xcl xmin_g]
```

3. Proposez un autre solveur permettant de résoudre un programme linéaire[5]

[2] http://www.gurobi.com/
[3] http://www-03.ibm.com/software/products/en/ibmilogcpleoptistud/
[4] https://www.gnu.org/software/glpk/
[5] http://plato.asu.edu/ftp/lpsimp.html