Université Clermont Auvergne
Recherche opérationnelle
DUT Info 2A

Florent Foucaud

2020-2021

TP : Solveurs de programmes linéaires

Il existe beaucoup de solveurs de programmes linéaires. L'un des plus connus (car plus performant) est peut-être CPLEX, développé par Robert E. Bixby de l'université de Rice à Houston au Texas, en 1988. CPLEX est racheté en 1997 par l'entreprise française ILOG puis à son tour racheté en 2009 par IBM (mais nous n'avons pas la licence). R. E. Bixby a depuis créé un autre solveur commercial, Gurobi, en 2008 (Gurobi vient des noms de famille des fondateurs : Gu, Rothberg, Bixby).

Nous allons utiliser plusieurs autres logiciels dans ce TP.

Exercice 1 (Utilisation du tableur Calc de LibreOffice)**.**

Le tableur Calc intègre un solveur de programmes linéaires, lp_solve. Utilisons-le sur l'exemple des courgettes et des navets du TD1, en rentrant les données suivantes :

	A	B	C	D	E
1		courgettes	navets		
2					
3	profit	4	5	=B3*B$2+C3*C$2	
4	engrais A	2	1	=B4*B$2+C4*C$2	8
5	engrais B	1	2	=B5*B$2+C5*C$2	7
6	Anti-P		1	=B6*B$2+C6*C$2	3

Le but est d'obtenir les valeurs optimales des variables dans les cases B2 et C2. Le profit optimal s'affichera dans la case D3.

Pour lancer le solveur, il faut aller dans le menu `Outils` → `Solveur`, entrer la case correspondant à la fonction objectif dans le champ "cellule cible" (ici c'est D3), les cases contenant les variables dans le champ "par modification des cellules" (B2 et C2), et spécifier les contraintes dans les champs "conditions de limitation" (par exemple la première contrainte sera D4 $\leq$ E4).

Jouer avec le solveur.

Un solveur est également disponible dans Excel, mais il faut d'abord installer l'add-in gratuit "solveur". Le fonctionnement est similaire, le solveur est accessible dans le menu `Données` → `Analyse`*. On peut également y produire un "rapport de sensitivité", pour évaluer l'impact de changements de coefficients dans les contraintes, comme le "coût marginal" d'une contrainte.*

Exercice 2 (Utilisation de GLPK)**.**

Le GNU Linear Programming Kit (GLPK), dont la première version fut développée par Andrew O. Makhorin du Moscow Aviation Institute en 2000, est une suite logicielle pour la programmation linéaire sous GNU. On peut décrire un programme linéaire sous divers formats standards, par exemple, le format `lp`.

Voici le contenu d'un fichier `test.lp` :

```
Maximize
    nom_objectif: x1 + 2 x2 - x3
Subject To
    nom_contrainte_1: x1 + 3 x2 <= 12
    nom_contrainte_2: - x1 + 4 x3 >= 40
Bounds
    x2 <= 10
End
```

qui représente le progrmme linéaire suivant (noter que les variables sont non-négatives par défaut) :

$$\begin{array}{lrcrcrcr} \text{maximiser} & x_1 & + & 2x_2 & - & x_3 & & \\ \text{tel que} & x_1 & + & 3x_2 & & & \leq & 12 \\ & -x_1 & & & + & 4x_3 & \geq & 40 \\ & & & x_2 & & & \leq & 10 \\ & x_1 & & & & & \geq & 0 \\ & & & x_2 & & & \geq & 0 \\ & & & & & x_3 & \geq & 0 \end{array}$$

On peut ensuite exécuter le solveur avec la commande : `glpsol --lp test.lp -o solution.txt`, qui écrit la solution et un rapport dans le fichier `solution.txt`. La valeur optimale de la fonction objectif est dans la ligne "objective". Les valeurs des variables sont dans la colonne "activity" du deuxième tableau. (L'option `--ranges sensitivity.txt` écrit un rapport de sensitivité dans le fichier `sensitivity.txt`.)

A vous de jouer :

(a) Résoudre le problème du régime alimentaire à 5 variables présenté dans le cours magistral (voir ci-dessous).

(b) Jouer avec les contraintes et les prix des denrées, pour obtenir un régime qui vous parait intéressant et/ou cohérent. Observer l'influence des différentes contraintes sur la solution.

aliment	prix (€/kg)	protéines (g/kg)	vitamine C (mg/kg)	fer (mg/kg)
Ananas	3.1	5	478	3
Banane	2.1	10	70	12
Carotte	1.6	7.8	20	2.4
Datte	8.7	25	4	10
Endive	3.8	13	65	8

Soient a, b, c, d, e les quantités d'ananas, bananes, carottes, dattes, endives (en kg). On obtient le programme linéaire suivant :

$$\begin{array}{lrcrcrcrcrcr} \text{minimiser :} & 3.1a & + & 2.1b & + & 1.6c & + & 8.7d & + & 3.8e & & \\ \text{tel que :} & 5a & + & 10b & + & 7.8c & + & 25d & + & 13e & \geq & 56 \\ & 478a & + & 70b & + & 20c & + & 4d & + & 65e & \geq & 110 \\ & 3a & + & 12b & + & 2.4c & + & 10d & + & 8e & \geq & 2 \\ & a & & & & & & & & & \geq & 0 \\ & & & b & & & & & & & \geq & 0 \\ & & & & & c & & & & & \geq & 0 \\ & & & & & & & d & & & \geq & 0 \\ & & & & & & & & & e & \geq & 0 \end{array}$$

Exercice 3 (Utilisation de Sagemath)**.**

Sagemath est très pratique pour manipuler des graphes, et s'interface avec des solveurs de PL. Pour le lancer, écrire la commande `sage`, puis, dans la console de sage, écrire `notebook()`. Cela lance une interface (basée sur `jupyter notebook`) dans votre navigateur. Vous pouvez ensuite créer une nouvelle feuille de calcul en cliquant sur `New worksheet`. Ensuite on code en Python.

On peut définir et résoudre un PL dans Sagemath de la façon suivante :

```
p = MixedIntegerLinearProgram()
v = p.new_variable(real=True, nonnegative=True)              #v est un dictionnaire qui va
                                                              contenir les variables du PL
x, y, z = v[0], v[1], v[2]                                   #on initialise des variables
                                                              dans le dictionnaire
p.set_objective( x + y + 3*z )                               #fonction objectif
p.add_constraint( x + 2*y <= 4 )                             #contraintes
p.add_constraint( 5*z - y <= 8 )
opt = p.solve()                                              #résoudre le PL
print(opt, p.get_values(x), p.get_values(y), p.get_values(z))  #afficher la valeur optimale
                                                              et les valeurs des variables
```

Cela correspond au programme linéaire suivant :

$$
\begin{array}{lrcrcrcl}
\text{maximiser} & x & + & y & + & 3z & & \\
\text{tel que} & x & + & 2y & & & \leq & 4 \\
 & & & -y & + & 5z & \leq & 8 \\
 & x & & & & & \geq & 0 \\
 & & & y & & & \geq & 0 \\
 & & & & & z & \geq & 0
\end{array}
$$

Beaucoup de fonctions de base sur les graphes sont implémentées dans Sagemath (voir `https://doc.sagemath.org/html/en/reference/graphs/index.html`). Il existe aussi beacuoup d'algorithmes avancés qui y sont implémentés. (D'ailleurs, souvent, la méthode utilisée est la PL.)

Pour créer et manipuler un graphe étiqueté dans Sagemath on peut écrire : [1]

```
G = DiGraph(3)                 #Crée un graphe orienté à 3 sommets (nommés 0,1,2 par défaut)
G.add_vertex(name="toto")      #ajoute un sommet nommé "toto"
G.add_edge(1,2,6)              #Ajoute l'arc 1->2 avec l'étiquette (label) "6"
for e in G.edges():            #boucle sur la liste des arcs de G
    print(e[0],e[1],e[2])      #affiche l'origine, la destination, et l'étiquette de l'arc e
for v in G.vertices():         #boucle sur la liste des sommets de G
    print(G.neighbors_out(v))  #affiche la liste des voisins sortants de v
    print(G.neighbors_in(v))   #affiche la liste des voisins entrants de v
    print(G.neighbors(v))      #affiche la liste de tous les voisins de v
    print(G.incoming_edges(v)) #affiche la liste des arcs entrant dans v
    print(G.outgoing_edges(v)) #affiche la liste des arcs sortant de v
G.show(edge_labels=True)       #dessine le graphe en affichant les étiquettes
```

A vous de jouer :

(a) Dans Sagemath, créer le graphe orienté étiqueté qui correspond au réseau de la figure ci-dessous.

(b) Créer le programme linéaire qui résoud le problème de flot pour aller d'Elphy à Santenago (voir la vidéo 2 du cours du 2 février). Le résoudre et afficher la solution.

1. Voir `https://doc.sagemath.org/html/en/reference/graphs/index.html` pour plus de fonctions, si nécessaire.

(c) Écrire une fonction générique `def flot(G,s,t)` qui prend en entrée un graphe étiqueté G (les étiquettes sont les capacités initiales), le sommet source `s` et le sommet destination `t`. Dans cette fonction, on définit et on résout le PL associé au problème du flot dans G. La fonction renvoie la solution optimale sous forme d'un couple (graphe étiqueté, valeur optimale du flot), où les étiquettes des arcs représentent les valeurs de flot pour ces arcs.

(d) Tester votre fonction sur le graphe de la question (a).