TP - Utilisation de GLPK et de gmpl

ECE - Outils pour la logistique

2016-2017

Avant de commencer - Installation de GLPK sous windows

Télecharger les sources de `glpk` sur la page suivante :
`http://cedric.cnam.fr/~lamberta/ECE/OutLog/sources/`

et télécharger le fichier d'installation :
- `glpk-4.57.tar.gz` pour linux
- `winglpk-4.57.zip` pour windows

Une fois installé, le logiciel s'utilise ensuite sous l'éditeur de commande.

Si besoin, il faut une modification de la variable d'environnement `PATH` pour que le système sache où trouver l'exécutable `glpsol.exe` au moment de l'appel.

Exercice 1 — *Résolution du sac à dos*

Le problème du sac à dos fait partie des problèmes classiques de la Recherche Opérationnelle.
Etant donnés :
- un sac à dos de volume total b,
- n objets tels que chaque objet i possède :
  - un volume a_i,
  - une utilité c_i

On souhaite remplir le sac en maximisant l'utilité des objets qu'on y met. On fait ici l'hypothèse que les coefficients c_i , a_i et b sont positifs.
Formulation : En choisissant comme variables : x_i qui vaut 1 si l'objet i est mis dans le sac et 0 sinon, le problème peut être modélisé par le programme linéaire en nombres entiers suivant :

$$(SAD)\begin{cases} \max \quad \sum_{i=1}^{n} c_i x_i \\ \sum_{i=1}^{n} a_i x_i \leqslant b \\ x \in \{0,1\}^n \end{cases}$$

1. Après avoir compris le codage du sac à dos ci-desous, récupérez le fichier qui représente ce modèle `exo1.mod` sur la page du cours.

```
#modele de sac a dos

#donnees
param n ;           #nombre d'objets
```

```
param C{i in 1..n}; #utilité de l'objet i
param A{i in 1..n}; #poids de l'objet i
param B;            #capacité du sac


#variables
var x{1..n} binary;

#objectif
maximize f :sum {i in 1..n} C[i]*x[i] ;

#contraintes
subject to
capacite : sum{i in 1..n} A[i]*x[i] <= B ;

printf "------Debut de la resolution -----\n";
solve;

printf "------Fin de la resolution -----\n";
display x;

end;
```

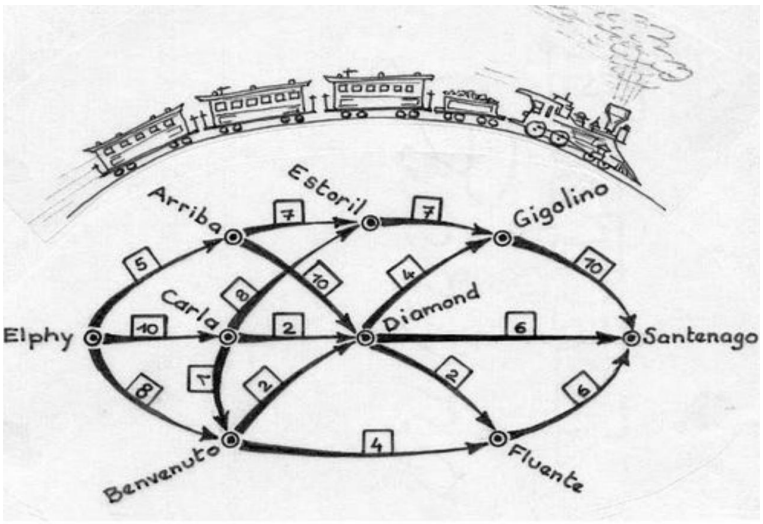
2. Récupérez le fichier de données associé `exo1.dat` :

```
#un fichier de donnees pour le probleme de sac a dos

data;
param n := 5;

param C := 1 12
           2 15
           3 5
           4 16
           5 17;

param A := 1 2
           2 6
           3 1
           4 7
           5 8;

param B := 20;

end;
```

3. Pour résoudre le programme mathématique obtenu par juxtaposition du modèle et des données, il faut exécuter la commande :
   `glpsol -m sac_a_dos.mod -d sac_a_dos.dat`
   on peut même demander que le programme mathématique obtenu soit écrit dans un fichier au format lp :
   `glpsol -m sac_a_dos.mod -d sac_a_dos.dat -wcpxlp sac.lp`
   Exécutez cette commande et regardez le contenu du fichier texte `sac.lp`
4. L'intérêt est qu'on peut changer les données. Ecrire un autre fichier de données `sac_a_dos2.dat` avec par exemple $n = 10$ et des données que vous choisissez et exécutez :
   `glpsol -m sac_a_dos.mod -d sac_a_dos2.dat`

Exercice 2 — ***Capacité journalière d'un réseau ferroviaire - problème de flot maximum***

Sur le réseau ferroviaire suivant, on a indiqué sur chaque tronçon entre 2 villes le nombre maximum de trains qui peuvent passer par jour dans le sens indiqué.



En sachant qu'aller de Elphy à Santenago prend moins d'un jour et que chaque jour, il peut partir au plus 23 trains d'Elphy, le problème posé est le suivant : combien de ces trains, au maximum, peuvent parvenir dans la journée à Santenago ?

On se propose de formaliser ce problème par un problème de flot maximum. Pour cela, nous allons construire un réseau de transport, dans lequel nous calculerons la quantité maximum de flot que l'on peut acheminer.

Rappelons tout d'abord qu'un réseau de transport est un graphe ayant une source `s`, un puits `p`, des capacités sur les arcs `c(i,j)`, et où pour tout sommet intermédaire `s`, il exite un chemin reliant `s` à `p` qui passe par `s`.

Finalement, le problème du flot maximum consiste, dans ce réseau de transport, à maximiser le flot total émis par la source `s`, tel qu'il y ait conservation du flot en chaque sommet intermédiaire `s`, tout en respectant les capacités des arcs.

1. Modélisation du problème par un problème de flot maximum.

   (a) Construction du réseau de transport : quels seront les sommets, arcs et capacités des arcs pour ce problème ?

   (b) Quelle seront la source, le puits, et les sommets intermédiaires ?

   (c) Exprimer le lien entre notre problème et un problème de flot maximum.

2. Résolution par un programme linéaire.
   Nous allons maintenant modéliser le problème du flot maximum par un programme linéaire. Pour cela il faut considérer les données, les variables, les contraintes et la fonction objectif du problème.

   — *Les données :*
   $G = (S, A)$: le graphe qui est un réseau de transport,
   $s \in S$: le sommet source du réseau de transport,
   $p \in S$: le sommet puits du réseau de transport,
   La matrice C représentant les capacités des arcs, où $c(i, j)$ est la capacité de l'arc (i, j). Si l'arc n'exixte pas, nous considérons sa capacité à 0.

— *Les variables :*
$x(i,j)$: valeur du flot de l'arc (i,j),
v : valeur du flot maximum.

— *Contraintes de flot*
Elles représentent le fait que pour chaque sommet, la somme du flot entrant est égale à la somme du flot sortant. Les sommets source et puits doivent être traités de façon particulière.

$$\forall i \in S\backslash\{s,p\}, \quad \sum_{j\in S} x(i,j) - \sum_{j\in S} x(j,i) = 0$$

$$\sum_{j\in S} x(s,j) - \sum_{j\in S} x(j,s) = v$$

$$\sum_{j\in S} x(p,j) - \sum_{j\in S} x(j,p) = -v$$

— *Contraintes de capacité*

$$\forall (i,j) \in A, \quad 0 \leqslant x(i,j) \leqslant c(i,j)$$

— *Fonction objectif*
On cherche ici à maximiser la valeur du flot : v.

On obtient le modèle linéaire suivant qui résoud le problème de flot maximum :

$$\text{(Flot max)} \begin{cases} \max v \\ \text{s.c.} \\ \quad \displaystyle\sum_{j\in S} x(i,j) - \sum_{j\in S} x(j,i) = 0 \quad \forall i \in S\backslash\{s,p\} \\ \quad \displaystyle\sum_{j\in S} x(s,j) - \sum_{j\in S} x(j,s) = v \\ \quad \displaystyle\sum_{j\in S} x(p,j) - \sum_{j\in S} x(j,p) = -v \\ \quad 0 \leqslant x(i,j) \leqslant c(i,j) \quad \forall (i,j) \in A \end{cases}$$

Coder le modèle ci-dessus. Tester le ensuite sur l'instance proposée en utilisant le fichier de données `exo2.dat`.

Exercice 3 — *Ordonnancement d'un chantier d'une mine - Graphe Potentiel-Tâche*

La mise en exploitation d'un nouveau gisement minier demande la réalisation d'un certain nombre de tâches. Le tableau suivant représente ces différentes tâches avec leurs relations d'antériorité.

Tâche	Description	Durée (en jours)	Tâches antérieures
A	Obtention d'un permis d'exploitation	120	-
B	Etablissement d'une piste de 6 km	180	A
C	Transport et installation de 2 sondeuses	3	B
D	Création de batiments provisoires pour le bureau des plans	30	B
E	Goudronnage de la piste	60	B
F	Adduction d'eau	90	D
G	Campagne de sondage	240	C,D
H	Forage et équipement de trois puits	180	E,F,G
I	Construction de bureaux et logements, ouvriers et ingénieurs	240	E,F,G

Quelles sont les dates au plus tôt de chaque tâche et le temps minimum de réalisation de l'ensemble du projet ?

1. Modélisation par un graphe Potentiel-Tâches.
   (a) Construction du graphe Potentiel-Tâches : quels seront les sommets et les arcs ?
   (b) Quelle seront les valuations des arcs ?
   (c) Construisez le graphe Potentiel-Tâches associé à l'instance proposée.
   (d) Précisez le lien entre ce graphe et notre problème.

2. Résolution par un programme linéaire.

   — *Les données :*
   T : l'ensemble des n tâches du problème qui comprend aussi les tâches `debut` et `fin` du projet,
   A : la matrice d'adjacence du graphe : $A \in n \times n$ et $a(i,j) = 1$ si (i,j) est un arc dans le graphe potentiel tâche, sinon $a(i,j) = 0$
   $d \in \mathbb{R}^n$ tel que $d(i)$ est la durée de la tâche i.
   Pour les tâches `debut` et `fin`, nous considérons comme durée 0.
   — *Les variables :*
   $t \in \mathbb{R}^n$, tel que $t(i)$ est la date au plus tôt de la tâche i.

   — *Les contraintes d'antériorité :*
   Elles représentent le fait qu'une tâche ne peut commencer que si les tâches la précédant sont finies :
   $$\forall (i,j) : a(i,j) = 1, \quad t(i) + d(i) \leqslant t(j)$$
   — *Les contraintes de positivité :*
   $$\forall i \in T, \quad t(i) \geqslant 0$$
   — *Fonction objectif*
   On cherche ici à minimiser le temps total d'ordonnancement, et donc la valeur de la date au plus tôt de la tâche fin : $t(\text{fin})$.

   On obtient le modèle linéaire suivant qui résoud notre problème d'ordonnancement :

$$\text{(MPM)} \begin{cases} \min t(\text{fin}) \\ \text{s.c.} \\ \quad t(i) + d(i) \leqslant t(j) \quad \forall a(i,j) = 1 \\ \quad t(i) \geqslant 0 \quad \forall i \in T \end{cases}$$

   Coder le modèle ci-dessus. Tester le ensuite sur l'instance proposée en utilisant le fichier de données `exo3.dat`.

Exercice 4 — *Problème de découpe* Une entreprise fabrique des rouleaux d'étoffe de 210 cm de large et de 100 m de long. Ses clients lui commandent des rouleaux de 100 m de long, mais dont les largeurs sont variables. Les commandes immédiates sont les suivantes :

Nb de rouleaux	Largeur en cm
1	110
3	90
2	60

Comment découper les rouleaux de 210 cm de large de manière à minimiser les chutes ?

Résolution par un programme linéaire.

— *Les données :*
P : le nombre total de rouleaux de taille 210×100 disponibles. On suppose que le nombre de rouleaux disponible est suffisant pour satisfaire la demande (i.e. $P = \sum_{i=1} b_i$).
L : la largeur d'un rouleau à découper
n : le nombre de types de rouleaux commandés
$l(i)$: la largeur du rouleau de type i
$b(i)$: la quantité de rouleaux de type i demandés

— *Les variables :*
$x(i,p) \in \mathbb{N}$: le nombre de rouleaux de type i découpé dans le rouleau p
$y(p) \in \{0,1\}$: qui vaut 1 si le rouleau p est découpé.
— *Les contraintes de taille de rouleaux :*
Elles représentent le fait qu'on ne peut découper plus que la largeur d'un rouleau p

$$\forall p \in 1,\dots,P, \quad \sum_{i=1}^{n} l(i)x(i,p) \leqslant Ly(p)$$

— *Les contraintes de satisfaction de la demande :*

$$\forall i \in 1,\dots,n, \quad \sum_{p=1}^{P} x(i,p) \geqslant b(i)$$

— *Fonction objectif*
On cherche ici à minimiser le nombre de rouleaux découpés

$$\sum_{p=1}^{P} y(p)$$

.

On obtient le modèle linéaire suivant qui résoud notre problème de découpe :

$$(\mathrm{D}) \begin{cases} \min \sum_{p=1}^{P} y(p) \\ \text{s.c.} \sum_{i=1}^{n} l(i)x(i,p) \leqslant Ly(p) \quad \forall p \in 1,\dots,P \\ \sum_{p=1}^{P} x(i,p) \geqslant b(i) \quad \forall i \in 1,\dots,n \\ x(i,p) \in \mathbb{N} \quad \forall (i,p) \in (1,\dots,P) \times (1,\dots,n) \\ y(p) \in \{0,1\} \quad \forall p \in 1,\dots,P \end{cases}$$

Coder le modèle ci-dessus. Tester le ensuite sur l'instance proposée en utilisant le fichier de données `exo4.dat`.

Exercice 5 — *Affectation linéaire de coût minimum* Dans la commune de Montvert, il y a 6 quartiers et 3 maternelles. Les coûts de transport par élève des 6 zones aux différentes écoles sont donnés dans le tableau ci-dessous. Le "0" indique que le transport n'est pas assuré par la commune alors que le signe "-" indique une affectation impossible.

Zone	Nombre d'élèves	Ecole 1	Ecole 2	Ecole 3
1	450	300	0	700
2	600	-	400	500
3	550	600	300	200
4	350	200	500	-
5	500	0	-	400
6	450	500	300	0
Capacité d'acceuil des écoles		900	1100	1000

Quelle est l'affectation des élèves dans les trois écoles, qui minimise le coût global de transport pris en charge par la commune ?

Pour résoudre ce problème, nous proposons une formulation par un programmation linéaire.

— *Les données :*
E : l'ensemble des écoles,
Z : l'ensemble des zones,
$Cap(k)$: Capacité de l'école k,
$Nb(i)$: nombre d'élèves de la zone i,
$c(i,k)$: coût de transport d'un élève de la zone i vers l'école k.

— *Les variables :*
$x(i,k)$: nombre d'enfants de la zone i affectés à l'école k.

— *Les contraintes d'affectation :*
Elle représente le fait que tous les élèves doivent être affféctés à une école :

$$\forall i \in Z, \quad \sum_{k \in E} x(i,k) = Nb(i)$$

— *Les contraintes de capacité :*

$$\forall k \in E, \quad \sum_{i \in Z} x(i,k) \leqslant Cap(k)$$

— *Les contraintes de positivité :*

$$\forall i \in Z,\, k \in E \quad x(i,k) \geqslant 0$$

— *Fonction objectif*
On cherche ici à minimiser le coût total de prise en charge par la commune :

$$\sum_{i \in Z} \sum_{k \in E} c(i,k) * x(i,k)$$

On obtient le modèle linéaire suivant qui résoud le problème de l'affectation linéaire :

$$(\text{Aff}) \begin{cases} \min \displaystyle\sum_{i \in Z} \sum_{k \in E} c(i,k) * x(i,k) \\ \text{s.c.} \\ \quad \displaystyle\sum_{k \in E} x(i,k) = Nb(i) \quad \forall i \in Z \\ \quad \displaystyle\sum_{i \in Z} x(i,k) \leqslant Cap(k) \quad \forall k \in E \\ \quad x(i,k) \geqslant 0 \quad \forall i \in Z,\, k \in E \end{cases}$$

Coder le modèle ci-dessus. Tester le ensuite sur l'instance proposée en utilisant le fichier de données `exo5.dat`.

Exercice 6 — *Problème de sous-traitance* Une entreprise à la possibilité de vendre deux produits p_1 et p_2 qu'elle se charge de fabriquer elle même ou d'acheter à un sous-traitant. Il s'agit pour cette entreprise, de savoir dans quelle mesure il est préférable de cumuler les rôles de producteur et de vendeur. Les produits p_1 et p_2 rapportent respectivement 40 et 20 euros s'ils sont fabriqués, 30 et 10 euros s'ils sont achetés au sous-traitant puis revendus. De plus, ils nécessitent respectivement 2 et 4 heures de machine pour être fabriqués. La disponibilité totale des équipements est de 400 heures. Une étude de marché montre que l'on peut espérer vendre 250 p_1 et 200 p_2. Quelles quantités de p_1 et de p_2 décidez-vous de fabriquer et d'acheter ?

1. Modéliser ce problème par un programme linéaire en nombres entiers.
2. Coder le et tester le ensuite sur l'instance proposée en utilisant le fichier de données `exo6.dat`.

Exercice 7 — *Un problème quadratique*

On considère le problème quadratique suivant :

$$(QP)\begin{cases} \min \quad x_1^2 - x_3^2 + 2x_1x_2 - 4x_1x_3 \\ \quad x_1 + x_2 \leqslant 4 \\ \quad x_1 + x_3 \leqslant 12 \\ \quad x \in \{0,1\}^3 \end{cases}$$

1. Reformuler (QP) en un problème équivalent qui est linéaire.
2. Ecrirele modèle générique de linéarisation d'un programme quadratique ayant des contraintes linéaires (fichier `lin.mod`)
3. Avec le fichier `exo7.dat`, tester votre modèle.