

TP- Utilisation GLPK et de son langage de modélisation MathProg

Avant de commencer

- le lien de téléchargement est:
 - https://sourceforge.net/projects/winglpk/?source=typ_redirect
- Télécharger winglpk-4.65
- On a un zip. Copier le zip sur C:\ puis déziper.
- Il est alors créé le répertoire : C:\winglpk-4.65
- L'exécutable se trouve dans w32 ou w64 selon votre ordinateur 32 ou 64 bits
- Modifier la variable d'environnement PATH
- Lui rajouter : ...; C:\winglpk-4.65\glpk-4.65\w64
- Ouvrir une fenêtre « invite de commande »

Vérifiez que le solveur GLPK est accessible en tapant : `glpsol --help`

1. Utilisation de GLPK

Exercice 1.a – Format LP

On veut modéliser et résoudre le problème :

$$\begin{cases} \max z = 3x_1 + 5x_2 \\ x_1 \leq 4 \\ x_2 \leq 6 \\ 3x_1 + 2x_2 \leq 18 \\ x_1 \geq 0 \quad x_2 \geq 0 \end{cases}$$

Pour ce faire, vous pouvez créer le fichier texte suivant, qui contient le problème à résoudre au format LP. Vous pouvez nommer ce fichier `ex1.lp`

```
\* exemple GLPK modèle au format LP *\nMaximize z: 3 x1 + 5 x2\nsubject to\n c1: x1<=4\n c2: x2<=6\n c3: 3 x1 + 2 x2 <= 18\n\nEnd
```

Ensuite , taper sur la ligne de commande :

```
glpsol --cpxlp ex1.lp
```

Si vous voulez plus d'information sur la solution , taper sur la ligne de commande :

```
glpsol --cpxlp ex1.lp --output ex1.sol
```

Le détail se trouve dans le fichier `.sol` .

Exercice 1.b – Format MathProg

Même exercice mais cette fois au format MathProg . On peut appeler le fichier ex1.mod.

```
# modèle ex1.mod
var x1 >=0;
var x2 >=0;

maximize z :
    3*x1 + 5*x2;

subject to
C1: x1 <= 4;
C2: x2 <= 6;
C3: 3*x1 + 2*x2 <= 18;
```

1- Créer ce fichier texte.

2- Ensuite, lancer la commande : `glpsol --model ex1.mod`

Maintenant, dans ce format, on peut aussi gérer ses propres éditions. Rajouter `solve` ; pour résoudre le problème puis faites les éditions voulues.

```
# modèle ex1.mod au format MathProg
var x1 >=0;
var x2 >=0;

maximize z :
    3*x1 + 5*x2;

subject to
C1: x1 <= 4;
C2: x2 <= 6;
C3: 3*x1 + 2*x2 <= 18;

printf "lancement du solve \n";
solve;
# display fonction objectif et solution;
printf "x1=%f\n", x1;
printf "x2=%f\n", x2;
printf "z=%f\n", z;
```

Vous pouvez ensuite résoudre le contenu de ce fichier par la même commande :

```
glpsol --model ex1.mod
```

Exercice 2 : Généralisons

On veut modéliser et résoudre le problème du sac à dos :

$$\begin{cases} \max u = \sum_{i=1}^n C_i x_i \\ \sum_{i=1}^n A_i x_i \leq B \\ x_i \in \{0,1\} \quad i = 1 \wedge n \end{cases}$$

1- Ecrire un fichier qui représente ce modèle (fichier texte de nom sac_a_dos.mod)

```
# Sac-à-dos en 0/1
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

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

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

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

printf "---- Avant résolution ----\n" ;

solve ;
display z ;
display x ;
```

2- Ecrire un fichier qui contient des données pour ce modèle (fichier texte de nom sac_a_dos.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- Taper la commande :

```
glpsol --model sac_a_dos.mod --data sac_a_dos.dat
```

4- Déterminer la valeur optimale de la relaxation continue du précédent problème :

```
glpsol --model sac_a_dos.mod --data sac_a_dos.dat --nomip
```

Comparer les valeurs obtenues.

L'intérêt d'avoir un modèle avec paramètres (et non des données en dur) 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 choisirez et exécutez .

2. Exercices de modélisation

Exercice 3 – Problème p-médian

Le problème p-médian est le suivant.

Etant donné :

- n : nombre de sites clients
- m : nombre de sites potentiels pour des dépôts
- p : nombre de dépôts à ouvrir
- D_{ij} : distance entre le site client i et le site dépôt j

On veut ouvrir p dépôts et affecter chaque client au dépôt ouvert qui lui est le plus proche, de façon à minimiser la somme des distances entre les clients et les dépôts auxquels ils sont affectés.

Proposez un modèle pour ce problème: deux familles de variables binaires 0-1

-variables pour décider de l'ouverture d'un dépôt

-variables pour modéliser l'affectation d'un client i à un dépôt j .

Instances disponibles à l'adresse suivante :

<http://perso.ensta-paristech.fr/~diam/ocro/distrib/pge06>

Exercice 4 – problème d'affectation

On a n tâches et n ouvriers. On doit affecter chaque tâche à un ouvrier. Une tâche j exécutée par l'ouvrier i coûte D_{ij} . La matrice D donne tous les coûts possibles. On veut minimiser le coût total d'affectation. Modéliser le problème. Penser à des variables de décision à 2 indices.

Ecrire le programme GLPK et tester le cas suivant :

D (coûts)	Tâche 1	Tâche 2	Tâche 3	Tâche 4
Ouvrier 1	8	3	1	5
Ouvrier 2	11	7	1	6
Ouvrier 3	7	8	6	8
Ouvrier 4	11	6	4	9

Exercice 5 – faire trier une liste par un programme linéaire

On dispose d'une liste D de n nombres entiers. On veut faire trier cette liste en ordre croissant par un programme linéaire. Ce programme aura (notamment) pour variables v_i $i=1$ à n où v_i est la valeur du nombre en position i dans la liste triée : $v_1 \leq v_2 \leq \dots \leq v_n$.

Exemple :

$$D = [6, 3, 4] \Rightarrow v = [3, 4, 6]$$

Une astuce recommandée est la suivante : on remarque que si on a une liste D et une liste C de coefficients décroissants pour minimiser $\sum_{i=1, \dots, n} D_i C_i$ il faut que D soit rangé dans l'ordre croissant.

Exemple :

$$C = [3, 2, 1] \quad D = [6, 3, 4] \quad 3 \times 6 + 2 \times 3 + 1 \times 4 = 28$$

$$C = [3, 2, 1] \quad D = [3, 4, 6] \quad 3 \times 3 + 2 \times 4 + 1 \times 6 = 23$$

Exercice 6 – Ordonnement de tâches sur une machine

n tâches doivent être exécutées sur une machine unique. Chaque tâche i , a une durée $d(i)$, une date d'échéance $e(i)$. Les tâches s'exécutent sur une unique machine. Les tâches ne sont pas sécables (une tâche commencée va jusqu'à son terme). Si une tâche se termine après sa date d'échéance, elle est en retard. Le retard d'une tâche i est $R(i)=\max\{0, t(i)-e(i)\}$, où $t(i)$ est la date de fin de la tâche i .

Il faut trouver un ordonnancement des tâches qui minimise la somme des retards des tâches.

Modéliser ce problème par un modèle mathématique en variables mixtes :

variables binaires $x_{ij}=1$ si tâche i avant j et $x_{ij}=0$ sinon,

variables continues t_i date de fin de la tâche i et R_i retard de la tâche i .

Résoudre l'exemple suivant.

Exemple :

$n=5$.

Tâches	1	2	3	4	5
Durée d	4	3	7	2	2
Echéances e	5	6	8	8	17

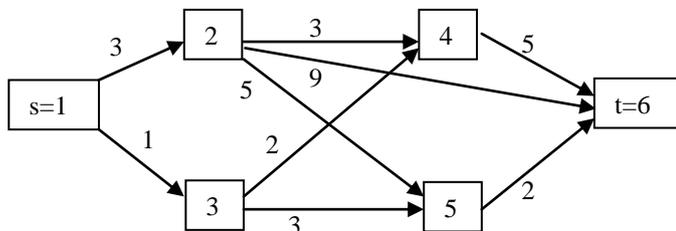
L'ordonnement optimal est 1, 2, 4, 3, 5 . Le retard total=11.

Exercice 7 – Plus court chemin dans un graphe

Soit un graphe orienté dont les arcs sont munis de poids positifs et 2 sommets s et t de ce graphe. On veut trouver le plus court chemin de s à t .

Le graphe a n sommets et est défini par sa matrice d'adjacence E de n lignes et n colonnes et la matrice C de n lignes et n colonnes des coûts des arcs.

Ecrire le modèle (linéaire) permettant de résoudre ce problème. Tester votre modèle sur l'instance suivante :



Exercice 8 – Bipartition de graphe et linéarisation

Un graphe complet d'ordre n avec des poids positifs ou nuls sur les arêtes. On veut faire une partition des sommets du graphe en 2 paquets. Le paquet 1 contient k sommets et le paquet 2 $n-k$ sommets. Le coût d'une partition est la somme des poids des arêtes « coupées » c'est-à-dire avec une extrémité dans le paquet 1 et l'autre dans le paquet 2. On veut faire une partition de coût minimum.

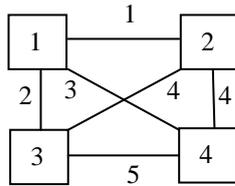
On définit, pour chaque sommet i , la variable x_i binaire telle que $x_i = 1$ si le sommet i est dans le paquet 1 et 0 sinon.

1-Montrer que l'arête $[i,j]$ est « coupée » si et seulement si $x_i(1-x_j)+x_j(1-x_i) = 1$.

2-En déduire la fonction objectif du problème à minimiser.

3-Cette fonction objectif contient des produits $x_i x_j$ ($i < j$). Pour linéariser chaque produit, on introduit une variable y_{ij} qui vaudra $x_i x_j$. Proposer les contraintes linéaires liant la variable y_{ij} et les variables x_i et x_j et qui feront que l'égalité $y_{ij} = x_i x_j$ sera bien réalisée quand x_i et x_j sont binaires.

4-Ecrire le modèle linéarisé et tester l'exemple suivant avec $k=2$:



Exercice 9 – Voyageur de commerce

Un voyageur de commerce doit visiter n villes et revenir à son point de départ. Les villes sont numérotées de 1 à n . Le point de départ est la ville 1. Les distances entre les villes sont données dans une matrice D . Le voyageur de commerce désire faire un tour de longueur minimale.

Proposer plusieurs modèles (voir ci-dessous) et résoudre l'instance suivante avec GLPK. Comparer les modèles au niveau de la relaxation continue et du temps de calcul.

```

data;
param n := 9;

param D : 1 2 3 4 5 6 7 8 9 :=
  1      0      796    482    640    583    449    801    244    876
  2      796    0      685    995    212    562    11     900    79
  3      482    685    0      309    463    761    701    538    763
  4      640    995    309    0      766    921    1011   405    1074
  5      583    212    463    766    0      338    229    676    292
  6      449    562    761    921    338    0      579    699    569
  7      801    11     701    1011  229    579    0      908    91
  8      244    900    538    405    676    699    908    0      969
  9      876    79     763    1074  292    569    91     969    0;

end;
```

- modèle 1 : dans le tour, chaque ville occupe une position : la 1^{ère} ou la 2^{ème} ou la 3^{ème} etc Contraintes linéaires mais l'objectif quadratique sera à linéariser. Proposer une linéarisation basique et des améliorations de cette linéarisation.
- modèle 2 : basé sur les arcs utilisés (parcours). Ce modèle entraîne le problème des sous-tours qu'il faudra éliminer. Comment éliminer les sous-tours ?

Exercice 10 – Sudoku

Proposez un modèle et un fichier de données pour trouver la solution de la grille de Sudoku ci-dessous. Affichez le résultat.

7			2		9			1
4	9	1		3		8	6	2
	8						9	
5			6		3			8
3	6	2				9	5	4
8			9		4			3
	5						1	
6	7	8		4		2	3	9
1			3		6			5