Feuille de travaux pratiques n° 3
Utilisation de GLPK en tant que bibliothèque de fonctions

Dans les TPs précédents, nous avons fait appel au solveur GLPK en utilisant le langage de modélisation GNU MathProg. Il s'agit en effet de la manière la plus simple et rapide pour "juste" résoudre un problème de programmation linéaire en variables mixtes. Cependant, si nous voulons intégrer l'utilisation du solveur dans un logiciel, ou écrire un algorithme utilisant comme routine le solveur GLPK, il peut être préférable d'appeler le solveur via la bibliothèque de fonction associée.

1 Exemple d'utilisation

Deux exemples de code en C faisant appel à GLPK sont donnés sur madoc (`musee.c` et `tp3.c`), l'utilisation de la bibliothèque y est abondamment commentée. Ici, l'utilisation d'une matrice creuse est obligatoire. Il s'agit de l'exercice 2.2 des feuilles de TD, qui a déjà été utilisé pour illustrer l'utilisation de matrice creuse avec GNUMathProg. Pour rappel, sa modélisation est :

$$\begin{array}{rcll}
\min z & = & \sum_{j=B}^{R} x_j & \\
 s.c. & & x_B + x_F + x_E & \geq & 1 \\
 & & x_B + x_C + x_D & \geq & 1 \\
 & & x_D + x_H + x_I & \geq & 1 \\
 & & x_E + x_G + x_L + x_M & \geq & 2 \\
 & & x_C + x_F + x_G + x_H + x_J & \geq & 1 \\
 & & x_I + x_J + x_P & \geq & 1 \\
 & & x_M + x_N & \geq & 1 \\
 & & x_K + x_L + x_N + x_O + x_R & \geq & 1 \\
 & & x_O + x_P + x_Q & \geq & 1 \\
 & & x_Q + x_R & \geq & 1 \\
 & & x_B, \ldots, x_R & \in & \{0,1\}
\end{array}$$

Dans le fichier `musee.c`, toutes les données sont saisies "en dur" dans le code. La modélisation saisie dans le code n'est donc pas réutilisable. Cependant, le code a l'avantage d'être bcp plus facile à lire. Cela permet en particulier de bien observer le remplissage de la matrice creuse. On pourra revenir sur le TP2 pour relire cette matrice.

Pour compiler :

```
gcc -c musee.c
gcc musee.o -lglpk -lm
```

Dans le deuxième fichier `tp3.c`, les données sont absentes (on les trouve dans le fichier `DonneesEx22`). La modélisation saisie est :

$$\begin{array}{rcl}
\min z & = & \sum_{j=1}^{n} c_j x_j \\
 & & \sum_{j=1}^{n} a_{ij} x_j \geq b_i \quad i = 1, \ldots, m \\
 & & x_j \in \{0,1\}
\end{array}$$

où le nombre de variables n, le nombre de contraintes m, les coûts c_j, les membres de droites b_i, les coefficients $a_{ij} \in \{0, 1\}$ de la matrice des contraintes sont des paramètres à lire dans le fichier de données. Le format ici choisi est :

- Première ligne composée de deux entiers indiquant le nombre de variables n et le nombre de contraintes m,
- Deuxième ligne composée d'entiers indiquant les coefficients c_j,
- Paquets de 3 lignes décrivant chaque contrainte, en indiquant
  - Le nombre de variables intervenant dans la contrainte,
  - Les indices des variables intervenant dans la contrainte,
  - Le coefficient du membre de droite b_i correspondant à la contrainte.

Des allocations dynamiques sont ici indispensables, et le remplissage de la matrice creuse des contraintes est ici moins explicite. Cela correspond cependant à une partie du futur travail vous attendant pour le projet. Il y aura en effet plusieurs instances numériques d'un même problème à résoudre.

2 Travail à faire

Après avoir bien compris cet exemple, on fera une résolution du problème de l'exercice 2.4 des TDs en appelant la bibliothèque de fonctions de GLPK via un code en C. Pour rappel, il s'agit d'un problème d'affectation (ici en maximisation), dont la modélisation est :

$$
\begin{aligned}
\max z = & \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \\
& \sum_{j=1}^{n} x_{ij} = 1 \qquad i = 1, \ldots, n \\
& \sum_{i=1}^{n} x_{ij} = 1 \qquad j = 1, \ldots, n \\
& x_{ij} \in \{0, 1\}
\end{aligned}
$$

Avec GNUMathProg, nous n'aurions pas recours à une matrice creuse pour saisir les contraintes, car on saisirait la modélisation comme "sur papier". Il faut de plus préciser que les variables du problème dans GLPK n'ont qu'un seul indice (qui commence de plus à 1). Une correspondance entre les x_{ij} de la modélisation sur papier et les x_i de GLPK est donc nécessaire. Par exemple si $n = 3$, on a ici une représentation de la matrice des contraintes avec une correspondance entre les double-indices des variables du problème et les indices de GLPK.

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
x_{11}	x_{12}	x_{13}	x_{21}	x_{22}	x_{23}	x_{31}	x_{32}	x_{33}
1	1	1						
			1	1	1			
						1	1	1
1			1			1		
	1			1			1	
		1			1			1

Un motif que l'on retrouvera quelque soit la taille du problème est ici apparent !

La marche à suivre est donc :

- Établir une correspondance entre les double-indices et les indices des variables du problème et les indices des variables dans GLPK,
- Bien identifier la façon de remplir la matrice creuse des contraintes,
- Passer à l'implémentation.

Le code réalisé devra être déposé sur madoc à une date définie par votre encadrant de TP.