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

Buts de la séance :
– Comprendre l'utilisation de GLPK en tant que bibliothèque de fonctions,
– Être capable de l'utiliser.

1 Compréhension et prise en main

Les exemples présentés en CM sont disponibles sur madoc dans l'archive `biblio.zip`. Cette archive contient 3 fichiers permettant de résoudre le modèle de l'exercice 2.2 des TDs : `musee.c` (contenant le modèle avec toutes les données saisies "en dur" dans le code), `generic.c` (contenant le modèle générique présenté en CM) et `DonneesEx22.txt` (données de l'exercice 2.2 pour compléter le modèle générique).

Il sera important de bien comprendre ces exemples avant de commencer. On pourra aussi observer le résultat obtenu par la résolution de ces problèmes.

2 Exercice

On résoudra le problème de modélisation de l'examen 2014, en utilisant la bibliothèque de fonctions de GLPK via un code en C. L'énoncé du problème et la modélisation associée sont posés ci-dessous.

2.1 Problème

Trois wagons de chemin de fer de charge utile limitée à 100 quintaux sont réservés pour transporter seize caisses. Les caisses et leurs poids en quintaux sont donnés dans le tableau ci-dessous. La question qu'on se pose est : "Comment affecter les caisses aux wagons de façon à respecter les charges utiles maximales et à minimiser la charge du wagon le plus chargé ?"

Numéro caisse	1	2	3	4	5	6	7	8
Poids (en quintal)	34	6	8	17	16	5	13	21
Numéro caisse	9	10	11	12	13	14	15	16
Poids (en quintal)	25	31	14	13	33	9	25	25

Afin d'écrire le problème sous une forme générique, nous noterons les données :
– n le nombre de caisses,
– m le nombre de wagons,
– p_i le poids de la caisse i ($i \in \{1, \ldots, n\}$).

Dans ce problème, les décisions à prendre consistent à choisir pour chaque caisse, le wagon de chargement. Nous obtenons donc les variables de décision suivantes :

$$x_{ij} = \begin{cases} 1 \text{ si on affecte la caisse } i \text{ au wagon } j, \\ 0 \text{ sinon} \end{cases}$$

où $(i, j) \in \{1, \ldots, n\} \times \{1, \ldots, m\}$.

Les contraintes exprimant le fait que chaque caisse doit être chargée dans un wagon s'exprime immédiatement par

$$\sum_{j=1}^{m} x_{ij} = 1, \forall i \in \{1, \ldots, n\}.$$

La charge de chaque wagon $j \in \{1, \ldots, m\}$ nous intéresse ensuite particulièrement, celle ci est exprimée par

$$\sum_{i=1}^{n} p_i x_{ij}.$$

Cette charge est inférieure ou égale à 100 quintaux pour chaque wagon, et nous souhaitons également minimiser la charge du wagon le plus chargé. Autrement dit, nous souhaitons minimiser

$$\max_{j=1,\ldots,m} \sum_{i=1}^{n} p_i x_{ij},$$

ce qui n'est pas (immédiatement) linéaire. Nous ajoutons donc une variable de décision $C_{\max}$ qui représentera (une fois bien définie) la charge maximale des wagons. Chaque wagon doit avoir une charge inférieure ou égale à la charge maximale

$$\sum_{i=1}^{n} p_i x_{ij} \leq C_{\max}, \forall j \in \{1, \ldots, m\},$$

et nous souhaitons également que $C_{\max}$ soit le plus petit possible, ce qui nous amème à minimiser la fonction objectif

$$z = C_{\max}.$$

La minimisation de $C_{\max}$ va faire qu'à l'optimum, $C_{\max}$ sera exactement égal à la charge du wagon le plus chargé.

En résumé, nous obtenons la modélisation suivante

$$\begin{array}{rlcll}
\min z = & C_{\max} & & & \\
\text{s.c.} & \sum_{j=1}^{m} x_{ij} & = & 1, & \forall i \in \{1, \ldots, n\}, \\
& \sum_{i=1}^{n} p_i x_{ij} & \leq & C_{\max}, & \forall j \in \{1, \ldots, m\}, \\
& x_{ij} & \in & \{0, 1\}, & \forall (i, j) \in \{1, \ldots, n\} \times \{1, \ldots, m\}, \\
& C_{\max} & \geq & 0 &
\end{array}$$

On pourrait éventuellement ajouter la contrainte $C_{\max} \leq 100$, mais cette contrainte sera redondante si le problème admet une solution admissible. Dans le cas contraire, la fonction objectif nous indiquera une valeur $C_{\max}$ plus grande que 100.

2.2 Vers l'implémentation

Nous avons ici un ensemble de variables (x_{ij}) et la variable $C_{\max}$, que l'on déclarerait séparément avec GNU MathProg. Cela n'est pas (directement) possible en utilisant la bibliothèque de fonctions de GLPK. Nous ne pouvons que déclarer un seul tableau (unidimensionnel) de variables (avec des indices entiers partant de **1**). Il est donc nécessaire de définir des indices pour les variables initialement appelées $C_{\max}$ et x_{ij}, où $(i, j) \in \{1, \ldots, n\} \times \{1, \ldots, m\}$. Le tableau ci-dessous donne une possibilité.

Variable	x_{11}	x_{12}	x_{13}	x_{21}	x_{22}	x_{23}	$\ldots$	x_{ij}	$\ldots$	C_{max}
Indice	1	2	3	4	5	6	$\ldots$	$(i-1) \times n + j$	$\ldots$	$m \times n + 1$

Suivant ces indices, le premier ensemble de contraintes peut donc s'écrire (en notant les variables simplement par X suivi des indices proposés ci-dessus)

$$\sum_{j=1}^{m} X_{(i-1)\times m+j} = 1, \forall i \in \{1, \ldots, n\}.$$

De même, le deuxième ensemble de contraintes peut donc s'écrire

$$\sum_{i=1}^{n} p_i X_{(i-1)\times m+j} - X_{m\times n+1} \leq 0, \forall j \in \{1,\ldots,m\}.$$

La matrice des contraintes (ici écrite dans le cas $m = 3$) a donc l'allure suivante.

$$\begin{pmatrix}
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & \ldots & 0 & 0 & 0 & 0 \\
\vdots & & & & & & & & & & & & & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots & 1 & 1 & 1 & 0 \\
p_1 & 0 & 0 & p_2 & 0 & 0 & p_3 & 0 & 0 & \ldots & p_n & 0 & 0 & -1 \\
0 & p_1 & 0 & 0 & p_2 & 0 & 0 & p_3 & 0 & \ldots & 0 & p_n & 0 & -1 \\
0 & 0 & p_1 & 0 & 0 & p_2 & 0 & 0 & p_3 & \ldots & 0 & 0 & p_n & -1
\end{pmatrix}$$

Vous êtes maintenant sur la bonne voie, à vous de finir...

Les fichiers (`.c` et éventuellement `.dat`) seront à déposer sur madoc dans l'espace correspondant à votre groupe, au plus tard à la date limite fixée par votre enseignant de TP.