# Introduction aux outils utilisés en Travaux Pratiques

Anthony Przybylski

Université de Nantes, L3 Informatique

# Plan

1. LibreOffice (lp_solve)
2. GLPK (GNU MathProg)
3. GLPK (bibliothèque)

# Plan

# LibreOffice

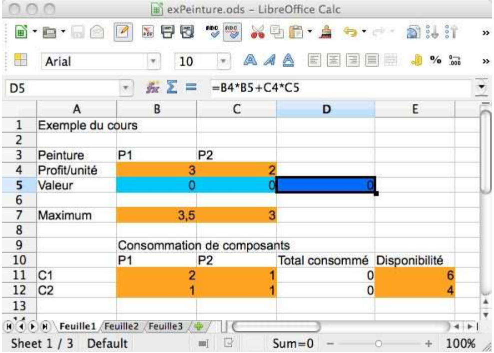
- Possibilité de résoudre des Programmes linéaires en variables mixtes, par l'intermédiaire d'une feuille de calcul
- Ergonomie du solveur très fortement inspirée de Microsoft Excel
- Le solveur appelé est lp_solve (sous licence GNU/LGPL)

# Exemple (1/5)

$$
\begin{array}{rrcl}
\max\ z = & 3x_1 + 2x_2 & & \\
s.c. & 2x_1 + x_2 & \leq & 6 \\
 & x_1 + x_2 & \leq & 4 \\
 & x_2 & \leq & 3 \\
 & x_1 & \leq & 3,5 \\
 & x_1, x_2 & \geq & 0
\end{array}
$$

à résoudre en utilisant lp_solve via LibreOffice

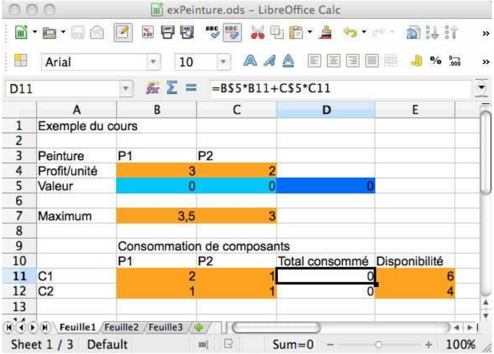
# Exemple (2/5)




- Orange : données
- Bleu clair : variables
- Bleu foncé : fonction objectif (formule!)

Informations à spécifier plus tard au solveur!

# Exemple (3/5)

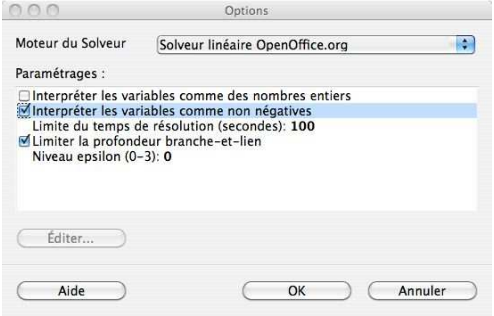



- Membre de gauche des contraintes précisés par des formules
- Alignement important et copier-coller utile!

# Exemple (4/5)


- Outils → Solveur, pour spécifier la fonction objectif (cellule cible), les variables (par modification de), et les contraintes (conditions de limitation)
- Options... pour spécifier le type des variables

# Exemple (5/5)




- Spécifier le type des variables est toujours important!
- Les variables ne sont pas supposées non-négatives par défaut (case à cocher absolument!)

# Exercice (à faire à la maison)

$$
\begin{array}{rcll}
\max\ z & = & 15x_1 + 60x_2 + 4x_3 + 20x_4 & \\
& s.c. & 20x_1 + 20x_2 + 10x_3 + 40x_4 & \leq \ 21 \\
& & 10x_1 + 30x_2 + 20x_3 & \leq \ 6 \\
& & 20x_1 + 40x_2 + 30x_3 + 10x_4 & \leq \ 14 \\
& & x_1, x_2, x_3, x_4 & \geq \ 0
\end{array}
$$

Résoudre ce PL (premier exemple du cours) en utilisant LibreOffice

# Conclusion

- Possibilité de résoudre un PL en variable entières en utilisant LibreOffice (et la plupart des tableurs en général)
- Utilisation simple...
- ... mais solveur limité (surtout avec les variables entières)
- Fichier `exPeinture.ods` disponible sur madoc

# Plan

# GLPK et GNU MathProg

- GLPK (GNU Linear Programming Kit) est une bibliothèque de fonctions écrite en langage C, conçue pour la résolution de programmes linéaires en variables mixtes
- Il est possible d'appeler directement le solveur via le langage de modélisation GNU MathProg
- Très utile et rapide (à coder) si on souhaite juste résoudre un PL en variables mixtes
- Syntaxe inspirée du langage de modélisation AMPL (Bell laboratories)

# Exemple

$$
\begin{array}{lrlcl}
\max\ z & = & 15x_1 + 60x_2 + 4x_3 + 20x_4 & & \\
 & s.c. & 20x_1 + 20x_2 + 10x_3 + 40x_4 & \leq & 21 \\
 & & 10x_1 + 30x_2 + 20x_3 & \leq & 6 \\
 & & 20x_1 + 40x_2 + 30x_3 + 10x_4 & \leq & 14 \\
 & & x_1, x_2, x_3, x_4 & \geq & 0
\end{array}
$$

à saisir avec GNU MathProg

# Exemple : Modèle explicite (1/2)

- Saisie du modèle dans un fichier dont l'extension est `.mod` (ici `medoc1.mod`)
- Déclaration des variables

```
var x1 >= 0;
var x2 >= 0;
var x3 >= 0;
var x4 >= 0;
```

- Déclaration de la fonction objectif

```
maximize profit :  15*x1 + 60*x2 + 4*x3 + 20*x4;
```

- Déclaration des contraintes

```
s.t.  Toxine1 :  20*x1 + 20*x2 + 10*x3 + 40*x4 <= 21;
s.t.  Toxine2 :  10*x1 + 30*x2 + 20*x3 <= 6;
s.t.  Toxine3 :  20*x1 + 40*x2 + 30*x3 + 10*x4 <= 14;
```

# Exemple : Modèle explicite (2/2)

- Résolution

```
solve;
```

- Affichage des résultats

```
display : x1,x2,x3,x4;
display : 'z=',15*x1 + 60*x2 + 4*x3 + 20*x4;
```

- Fin

```
end;
```

- Lancement de l'exécution en tapant dans un terminal

```
glpsol --model medoc1.mod
```

# Remarques

- Les variables sont par défaut continues et libres
- On peut préciser des bornes supérieures et inférieures sur les variables (`>=0` n'est qu'un cas particulier)
- Un type de variable peut aussi être spécifié (comme `integer` ou `binary`)
  Exemple : `var x >= 0 integer;`
- Il est obligatoire de nommer les contraintes

# Vers un modèle implicite

- Si on a plusieurs instances numériques (souvent!), il ne faut pas faire un modèle explicite pour résoudre chaque instance!
  $\Rightarrow$ Nécessité de séparer le modèle (sous une forme générique) des données
- On parle de modèle implicite

# Exemple : vers un modèle implicite (1/3)

- Déclaration d'une donnée (ici, le nombre de médicaments)
  `param tailleM;`
- Déclaration d'un ensemble (ensemble des indices des médicaments)
  `set M := 1..tailleM;`
- Déclaration d'un tableau de variables
  `var x{M} >=0;`
  Indices d'un tableau toujours spécifiés par un ensemble (qui peut contenir autre chose que des entiers)

# Exemple : vers un modèle implicite (2/3)

- Déclaration de la fonction objectif

```
maximize profit : 15*x[1] + 60*x[2] + 4*x[3] + 20*x[4];
```

- Déclaration des contraintes

```
s.t. Toxine1 : 20*x[1] + 20*x[2] + 10*x[3] + 40*x[4] <= 21;
s.t. Toxine2 : 10*x[1] + 30*x[2] + 20*x[3] <= 6;
s.t. Toxine3 : 20*x[1] + 40*x[2] + 30*x[3] + 10*x[4] <= 14;
```

- Résolution

```
solve;
```

- Affichage des résultats

```
display : x;
display : 'z=',15*x[1] + 60*x[2] + 4*x[3] + 20*x[4];
```

# Exemple : vers un modèle implicite (3/3)

Les données sont instanciées

- soit à la suite...
  ```
  data;
  tailleM := 4;
  end;
  ```
  ...dans un bloc commençant par l'instruction `data;`
- soit dans un fichier différent (sans le mot-clé `data;`) dont l'extension est `.dat`, le lancement de l'exécution devient
  ```
  glpsol --model NomFichier1.mod --data NomFichier2.dat
  ```

# Exemple : modèle implicite (1/4)

Déclaration de toutes les données et ensembles d'indices

- Nombre et ensemble d'indices des médicaments
```
param tailleM;
set M := 1..tailleM;
```
- Nombre et ensemble d'indices des toxines
```
param tailleT;
set T := 1..tailleT;
```
- Tableau des coefficients de la fonction objectif
```
param obj{M};
```
- Matrice des contraintes
```
param coeffcontr{T,M};
```
- Tableau des membres de droite des contraintes
```
param mdroite{T};
```

# Exemple : modèle implicite (2/4)

Écriture d'un modèle réellement implicite

- Déclaration d'un tableau de variables

```
var x{M} >= 0;
```

- Déclaration de la fonction objectif

```
maximize profit :  sum{j in M} obj[j]*x[j];
```

- Déclaration d'un ensemble de contraintes

```
Toxine{i in T} :  sum{j in M} coeffcontr[i,j]*x[j] <= mdroite[i];
```

# Exemple : modèle implicite (3/4)

- Résolution

```
solve;
```

- Affichage des résultats

```
display : x;
display : sum{j in M} obj[j]*x[j];
```

# Exemple : modèle implicite (4/4)

Instanciation des données

```
data;
param tailleM := 4;
param tailleT := 3;
param obj := 1 15 2 60 3 4 4 20;
param coeffcontr :  1  2  3  4 :=
                  1 20 20 10 40
                  2 10 30 20 0
                  3 20 40 30 10;
param mdroite := 1 21
                 2 6
                 3 14;
end;
```

Les indices d'un tableau de données sont nécessairement rappelés avant d'en spécifier le contenu

# Modèle implicite : remarques

- Séparation complète entre le modèle et les données
- Possibilité de déclarer plusieurs ensembles de contraintes (de manière similaire à l'écriture "à la main")

# Exemple : utilité d'une matrice creuse (1/5)

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

À saisir sous une forme générique (ex TD 2.2)

# Exemple : utilité d'une matrice creuse (2/5)

Déclaration de toutes les données et ensembles d'indices

- Nombre et ensemble d'indices des salles

```
param maxSalle;
set S := 1..maxSalle;
```

- Ensemble d'indices des caméras

```
set indCam;
```

- Tableau des coefficients de la fonction objectif

```
param obj{indCam};
```

- Matrice des contraintes

```
param coeffcontr{S,indCam};
```

- Tableau des membres de droite des contraintes

```
param mdroite{S};
```

# Exemple : utilité d'une matrice creuse (3/5)

Écriture du modèle implicite (comme précédemment)

- Déclaration d'un tableau de variables

```
var x{indCam} binary;
```

- Déclaration de la fonction objectif

```
minimize cout : sum{j in indCam} obj[j]*x[j];
```

- Déclaration de l'ensemble des contraintes

```
s.t. Salle{i in S} : sum{j in indCam}
coeffcontr[i,j] * x[j] >= mdroite[i];
```

# Exemple : utilité d'une matrice creuse (4/5)

- Résolution

```
solve;
```

- Affichage des résultats

```
display : x;
display{j in indCam : x[j] = 1} : j;
display : 'objectif : ', sum{j in indCam}
obj[j]*x[j];
```

# Exemple : utilité d'une matrice creuse (5/5)

## Instanciation des données

```
data;
param maxSalle := 10;
set indCam := B C D E F G H I J K L M N O P Q R;
param obj := B 1 C 1 D 1 E 1 F 1 G 1 H 1 I 1 J 1 K 1 L 1 M 1 N 1 O 1 P 1
Q 1 R 1;
param coeffcontr :  B C D E F G H I J K L M N O P Q R :=
                   1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0
                   2 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                   3 0 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0
                   4 0 0 0 1 0 1 0 0 0 0 1 1 0 0 0 0 0
                   5 0 1 0 0 1 1 1 0 1 1 0 0 0 0 0 0 0
                   6 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 0 0
                   7 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0
                   8 0 0 0 0 0 0 0 0 0 1 1 0 1 1 0 0 1
                   9 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0
                   10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1;
param mdroite := 1 1 2 1 3 1 4 2 5 1 6 1 7 1 8 1 9 1 10 1;
```

# Remarques

- Données peu lisibles, malgré la petite taille du problème
- Énormément de 1 dans les données
  $\Rightarrow$ Préférable dans ce cas d'utiliser des boucles
- Matrice des contraintes composée essentiellement de 0
  (Phénomène courant en optimisation combinatoire)
  $\Rightarrow$ Utilisation d'une matrice creuse

# Exemple : utilisation d'une matrice creuse (1/6)

Déclaration de toutes les données et ensembles d'indices

- Nombre et ensemble d'indices des salles
  ```
  param maxSalle;
  set S := 1..maxSalle;
  ```
- Ensemble d'indices des caméras
  ```
  set indCam;
  ```
- Ensemble des (double-)indices de la partie non-creuse de la matrice des contraintes (sous-ensemble de `S` $\times$ `indCam`)
  ```
  set SCam within S cross indCam;
  ```
- Tableau des coefficients de la fonction objectif
  ```
  param obj{indCam};
  ```
- Tableau des coefficients de la matrice des contraintes
  ```
  param coeffcontr{(i,j) in SCam};
  ```
- Tableau des membres de droites des contraintes
  ```
  param mdroite{S};
  ```

# Exemple : utilisation d'une matrice creuse (2/6)

Écriture du modèle implicite

- Déclaration du tableau de variables
```
var x{indCam} binary;
```
- Déclaration de la fonction objectif
```
minimize cout : sum{j in indCam} obj[j]*x[j];
```
- Déclaration de l'ensemble des contraintes
```
s.t. Salle{i in S} : sum{(i,j) in SCam}
coeffcontr[i,j] * x[j] >= mdroite[i];
```

# Exemple : utilisation d'une matrice creuse (3/6)

- Résolution

```
solve;
```

- Affichage des résultats

```
display{j in indCam : x[j] = 1} : j;
display : 'objectif : ', sum{j in indCam}
obj[j]*x[j];
```

# Exemple : utilisation d'une matrice creuse (4/6)

Instanciation des données (version lourde)

```
data;
param maxSalle := 10;
set indCam := B C D E F G H I J K L M N O P Q R;
set SCam := (1,B) (1,E) (1,F) (2,B) (2,C) (2,D) (3,D)
(3,H) (3,I) (4,E) (4,G) (4,L) (4,M) (5,C) (5,F) (5,G)
(5,H) (5,J) (5,K) (6,I) (6,J) (6,P) (7,M) (7,N) (8,K)
(8,L) (8,N) (8,O) (8,R) (9,O) (9,P) (9,Q) (10,Q) (10,R);
param obj := B 1 C 1 D 1 E 1 F 1 G 1 H 1 I 1 J 1 K 1 L 1 M
1 N 1 O 1 P 1 Q 1 R 1;
param coeffcontr := [1,B] 1 [1,E] 1 [1,F] 1 [2,B] 1 [2,C]
1 [2,D] 1 [3,D] 1 [3,H] 1 [3,I] 1 [4,E] 1 [4,G] 1 [4,L] 1
[4,M] 1 [5,C] 1 [5,F] 1 [5,G] 1 [5,H] 1 [5,J] 1 [5,K] 1
[6,I] 1 [6,J] 1 [6,P] 1 [7,M] 1 [7,N] 1 [8,K] 1 [8,L] 1
[8,N] 1 [8,O] 1 [8,R] 1 [9,O] 1 [9,P] 1 [9,Q] 1 [10,Q] 1
[10,R] 1;
param mdroite := 1 1 2 1 3 1 4 2 5 1 6 1 7 1 8 1 9 1 10 1;
```

# Exemple : utilisation d'une matrice creuse (5/6)

Instanciation des données (plus lisible)

```
data;
param maxSalle := 10;
set indCam := B C D E F G H I J K L M N O P Q R;
set SCam := (1,B) (1,E) (1,F) (2,B) (2,C) (2,D) (3,D)
(3,H) (3,I) (4,E) (4,G) (4,L) (4,M) (5,C) (5,F) (5,G)
(5,H) (5,J) (5,K) (6,I) (6,J) (6,P) (7,M) (7,N) (8,K)
(8,L) (8,N) (8,O) (8,R) (9,O) (9,P) (9,Q) (10,Q) (10,R);
param obj default 1;
param coeffcontr default 1;
param mdroite default 1:= 4 2;
```

`default` indique la valeur par défaut de toutes les cases d'un tableau

# Exemple : utilisation d'une matrice creuse (6/6)

Plusieurs possibilités équivalentes pour la déclaration de la partie non-creuse d'une matrice

- `set SCam := (1,B) (1,E) (1,F) (2,B) (2,C) (2,D) (3,D) (3,H) (3,I) (4,E) (4,G) (4,L) (4,M) (5,C) (5,F) (5,G) (5,H) (5,J) (5,K) (6,I) (6,J) (6,P) (7,M) (7,N) (8,K) (8,L) (8,N) (8,O) (8,R) (9,O) (9,P) (9,Q) (10,Q) (10,R);`
- `set SCam := (1,*) B E F (2,*) B C D (3,*) D H I (4,*) E G L M (5,*) C F G H J K (6,*) I J P (7,*) M N (8,*) K L N O R (9,*) O P Q (10,*) Q R;`

# Conclusion

- Archives `medoc.zip` et `camera.zip` contenant l'ensemble des exemples disponibles sur madoc
- GNU MathProg permet de saisir et résoudre facilement des programmes linéaires en variables mixtes
- Saisie du modèle se rapprochant de l'écriture "à la main"
- GLPK plus performant que lp_solve (mais moins que COIN_OR ou SCIP, et beaucoup moins que IBM CPLEX)
- Et si on veut faire plus avec GLPK?

# Plan

# GLPK (bibliothèque)

- Utilité : ne pas se limiter à seulement la résolution d'un programme linéaire en variables mixtes, intégrer l'utilisation du solveur pour un usage plus large
- Inconvénients :
  - Obligation de remplir une (unique) matrice creuse pour les contraintes
    $\Rightarrow$ Éloignement par rapport à l'écriture "à la main"
  - Coder en C (facile pour les informaticiens, mais pour les autres...)
  - Les indices commencent à 1 dans les structures de données de GLPK

# Exemple : modèle explicite (1/9)

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

À résoudre en utilisant la bibliothèque de fonctions GLPK

# Exemple : modèle explicite (2/9)

- Nécessaire inclusion de bibliothèques usuelles, plus GLPK

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <glpk.h>
```

- Premier exemple avec un modèle explicite, et des allocations statiques

```
#define NBVAR 17
#define NBCONTR 10
#define NBCREUX 34
```

# Exemple : modèle explicite (3/9)

Création d'un problème (initialement vide)

- Déclaration d'un pointeur sur le problème
  ```
  glp_prob *prob;
  ```
- Allocation mémoire
  ```
  prob = glp_create_prob();
  ```
- Affectation d'un nom (on peut mettre NULL)
  ```
  glp_set_prob_name(prob, "musee");
  ```
- Précision du sens d'optimisation (GLP_MIN = min, GLP_MAX = max)
  ```
  glp_set_obj_dir(prob, GLP_MIN);
  ```

# Exemple : modèle explicite (4/9)

Déclaration des contraintes (initialement vides)

- On a `NBCONTR` contraintes
  ```
  glp_add_rows(prob, NBCONTR);
  ```
- Pour chaque contrainte
  ```
  for(i = 1;i <= NBCONTR;i++) {
  ```
  - Contraintes vues sous la forme
    $$(l_i \leq) \sum_{j=1}^{n} a_{ij}x_j (\leq u_i)$$
    où $\sum_{j=1}^{n} a_{ij}x_j$ n'est pas encore déclaré
  - Précision du (ou des) membre(s) de droite
    ```
    if (i == 4) glp_set_row_bnds(prob, i, GLP_LO, 2.0, 0.0);
    else glp_set_row_bnds(prob, i, GLP_LO, 1.0, 0.0);
    ```
  ```
  }
  ```

# Exemple : modèle explicite (5/9)

Dans la fonction `glp_set_row_bnds`

- `GLP_LO` pour déclarer uniquement $l_i$ (paramètre 4)
- `GLP_UP` pour déclarer uniquement $u_i$ (paramètre 5)
- `GLP_FX` pour une contrainte d'égalité ($l_i = u_i$)
- `GLP_DB` pour déclarer $l_i$ et $u_i$ (soit deux contraintes)

# Exemple : modèle explicite (6/9)

Déclaration des variables

- On a `NBVAR` variables
  `glp_add_cols(prob, NBVAR);`
- Pour chaque variable
  `for(i = 1;i <= NBVAR;i++) {`
  - Bornes sur les variables $l_i \leq x_i \leq u_i$
    `glp_set_col_bnds(prob,i,GLP_DB,0.0,1.0);`
    Même principe que `glp_set_row_bnds`
  - Type des variables (par défaut continues)
    `glp_set_col_kind(prob, i, GLP_BV);`
    (`GLP_BV` = binaire, `GLP_IV` = entière)
  - Coefficients des variables dans la fonction objectif
    `glp_set_obj_coef(prob,i,1.0);`

  `}`

# Exemple : modèle explicite (7/9)

Remplissage de la matrice creuse des contraintes

- Déclaration des trois tableaux correspondants
  ```
  int ia[1 + NBCREUX];
  int ja[1 + NBCREUX];
  double ar[1 + NBCREUX];
  ```
- Remplissage (ici, pour la première contrainte)
  `ia[1] = 1;ja[1] = 1;ar[1] = 1.0;// + 1.0`$x_B$
  `ia[2] = 1;ja[2] = 4;ar[2] = 1.0;// + 1.0`$x_E$
  `ia[3] = 1;ja[3] = 5;ar[3] = 1.0;// + 1.0`$x_F$
  `...`
- `ia[1] = 1` indique la ligne 1 de la matrice
  `ja[1] = 1` indique la colonne 1 de la matrice
  `ar[1] = 1.0` indique le coefficient à la position (`ia[1],ja[1]`) de la matrice

# Exemple : modèle explicite (8/9)

- Chargement de la matrice dans le problème
  `glp_load_matrix(prob,NBCREUX,ia,ja,ar);`
- Résolution
  `glp_simplex(prob,NULL); // toujours`
  `glp_intopt(prob,NULL); // en cas de variables entières/binaires`

# Exemple : modèle explicite (9/9)

- Récupération du résultat
```
double z;
double x[NBVAR];
z = glp_mip_obj_val(prob);
for(i = 0;i < NBVAR; i++) x[i] = glp_mip_col_val(prob,i+1);
```
- Dans un PL en variables continues,
  `glp_mip_obj_val` est remplacé par `glp_get_obj_val`
  `glp_mip_col_val` est remplacé par `glp_get_col_prim`
- Libération mémoire
```
glp_delete_prob(prob);
```

# Exemple (modèle explicite) : Conclusion

- Ensemble simple, mais risque d'erreur important
- Pour debugger : utile de faire une écriture explicite du modèle dans un fichier
  `glp_write_lp(prob,NULL,"musee.lp");`
- Pour plus de lisibilité, on peut nommer les contraintes et les variables (dans leur boucle de déclaration)
  ```
  glp_set_row_name(prob,i,"NomContrainte i");
  glp_set_col_name(prob,i, "NomVariable i");
  ```
- Exemple complet (abondamment commenté) : `musee.c` sur madoc
- Compilation
  ```
  gcc -c musee.c
  gcc musee.o -lglpk -lm
  ```

# Exemple : modèle implicite (1/11)

$$
\begin{array}{rcll}
\min\ z & = & \sum_{j=1}^{n} x_j & \\
s.c. \ \sum_{j=1}^{n} a_{ij} x_j & \geq & b_i, & i \in \{1, \ldots, m\} \\
x_j & \in & \{0, 1\}, & j \in \{1, \ldots, n\}
\end{array}
$$

où $n$, $m$, $a_{ij} \in \{0, 1\}$, $b_i$ sont des données

Modèle générique à instancier et résoudre en utilisant la bibliothèque de fonctions GLPK

# Exemple : modèle implicite (2/11)

Données à lire dans un fichier texte formaté

- Première ligne : deux entiers $n$ et $m$
- Deuxième ligne : $n$ entiers définissant les coefficients $c_j$
- Paquets de 3 lignes décrivant chaque contrainte :
  - 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

# Exemple : modèle implicite (3/11)

Fichier `DonneesEx22.txt` correspondant à l'exemple 1

```
17 10
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
3
1 4 5
1
3
1 2 3
1
3
3 7 8
1
4
4 6 11 12
2
6
2 5 6 7 9 10
1
3
...
```

# Exemple : modèle implicite (4/11)

- Toutes les instances numériques ne sont pas de la même taille
  $\Rightarrow$ nécessité d'avoir recours à des allocations dynamiques!
- Données du problème à ranger dans une structure

```
typedef struct {
int nbvar; // n
int nbcontr; // m
int *couts; // Tableau des c_j
int **contr; // Tableau de tableaux indiquant les
indices des variables dans chaque contrainte
int *sizeContr; // Tableau des nombres de variables
dans chacune des contraintes
int *droite; // Tableau des b_i
} donnees;
```

# Exemple : modèle implicite (5/11)

- Allocations dynamiques et remplissage d'une variable `p` de type `donnees` effectuées de manière habituelle (voir fonction `lecture_data` dans le fichier `generic.c`)
- Il reste ensuite à créer le problème, le compléter, et le résoudre

# Exemple : modèle implicite (6/11)

Création d'un problème (initialement vide)

- Déclaration d'un pointeur sur le problème
```
glp_prob *prob;
```
- Allocation mémoire
```
prob = glp_create_prob();
```
- Affectation d'un nom (on peut mettre NULL)
```
glp_set_prob_name(prob, "musee");
```
- Précision du sens d'optimisation (GLP_MIN = min, GLP_MAX = max)
```
glp_set_obj_dir(prob, GLP_MIN);
```

Aucune différence ici!

# Exemple : modèle implicite (7/11)

Déclaration des contraintes (initialement vides)

- On a `p.nbcontr` contraintes
  ```
  glp_add_rows(prob, p.nbcontr);
  ```
- Pour chaque contrainte
  ```
  for(i = 1;i <= p.nbcontr;i++) {
  ```
  - Contraintes vues sous la forme
    $$(l_i \leq) \sum_{j=1}^{n} a_{ij}x_j (\leq u_i)$$
    où $\sum_{j=1}^{n} a_{ij}x_j$ n'est pas encore déclaré
  - Précision du (ou des) membre(s) de droite
    ```
    glp_set_row_bnds(prob, i, GLP_LO, p.droite[i-1],
    0.0);
    ```

  ```
  }
  ```

# Exemple : modèle implicite (8/11)

Déclaration des variables

- On a `p.nbvar` variables `glp_add_cols(prob, p.nbvar);`
- Pour chaque variable

```
for(i = 1;i <= p.nbvar;i++) {
```

  - Bornes sur les variables $l_i \leq x_i \leq u_i$
    `glp_set_col_bnds(prob,i,GLP_DB,0.0,1.0);`
    Même principe que `glp_set_row_bnds`
  - Type des variables (par défaut continues)
    `glp_set_col_kind(prob, i, GLP_BV);`
    (`GLP_BV` = binaire, `GLP_IV` = entière)
  - Coefficients des variables dans la fonction objectif
    `glp_set_obj_coef(prob,i,p.couts[i - i]);`

```
}
```

# Exemple : modèle implicite (9/11)

Remplissage de la matrice creuse des contraintes

- Déclaration et allocation mémoire des trois tableaux

```
int nbcreux = 0;
for(i = 0;i < p.nbcontr;i++) nbcreux += p.sizeContr[i];
ia = (int *) malloc ((1 + nbcreux) * sizeof(int));
ja = (int *) malloc ((1 + nbcreux) * sizeof(int));
ar = (double *) malloc ((1 + nbcreux) * sizeof(double));
```

- Remplissage (plus direct ici)

```
int pos = 1;
for(i = 0;i < p.nbcontr;i++) {
   for(j = 0;j < p.sizeContr[i];j++) {
     ia[pos] = i + 1;
     ja[pos] = p.contr[i][j];
     ar[pos] = 1.0;
     pos++;
   }
}
```

# Exemple : modèle implicite (10/11)

- Chargement de la matrice dans le problème

```
glp_load_matrix(prob,nbcreux,ia,ja,ar);
```

- Résolution

```
glp_simplex(prob,NULL); // toujours
glp_intopt(prob,NULL); // en cas de variables
entières/binaires
```

# Exemple : modèle implicite (11/11)

- Récupération du résultat

```
double z;
double *x = (double *) malloc (p.nbvar * sizeof(double));
z = glp_mip_obj_val(prob);
for(i = 0;i < p.nbvar; i++) x[i] = glp_mip_col_val(prob,i+1);
```

- Libération mémoire du problème

```
glp_delete_prob(prob);
```

- Libération mémoire des champs de la variable `p`, et des tableaux `ia, ja, ar, x`

# Conclusion

- Demande de bien comprendre la matrice creuse des contraintes avant de commencer à coder
- Attention aux indices!
- Exemple complet `generic.c` disponible sur madoc