M4202Cip – Recherche opérationnelle (V)
GNU Linear Programming Kit (GLPK)

bruno.colombel@univ-amu.fr

IUT d'Aix-Marseille
Site d'Arles
DUT Informatique

2019–2020

GLPK

GNU Linear Programming Kit

- Ensemble d'algorithmes permettant de résoudre des problèmes
  - programmation linéaire
  - programmation linéaire en nombres entiers

Découverte de GLPK (Linux magazine) avec le voyageur de commerce . . .

GLPK

GNU Linear Programming Kit

- Ensemble d'algorithmes permettant de résoudre des problèmes
  - programmation linéaire
  - programmation linéaire en nombres entiers
- API C, C++, R, Java, Python, etc.
- contient également son propre solveur : `glpsol`

Découverte de GLPK (Linux magazine) avec le voyageur de commerce . . .

GLPK

$$
\begin{aligned}
\max z \quad &= 3x_1 + 2x_2 \\
\text{s.c.} \quad & x_1 <= 4 \\
& x_2 \leq 6 \\
& 3x_1 + 2x_2 \leq 18 \\
& x_1, x_2 \geq 0
\end{aligned}
$$

Modélisation GLPK

GLPK

```
$glpsol -m exemple.mod


OPTIMAL LP SOLUTION FOUND
Time used:   0.0 secs
Memory used: 0.1 Mb (118308 bytes)
x1=2.000000
x2=6.000000
z=36.000000
Model has been successfully processed
```

Sac à dos

Sudoku

Problème de production

Sommaire

Problème du sac à dos

On considère le problème du sac-à-dos de capacité $W = 22$ avec les objets suivants :

Objets	a	b	c	d	e
poids	3	4	3	3	13
utilité	12	12	9	15	26

Solution optimale : a, c, d, e (poids = 22, utilité = 62)

Problème du sac à dos

- **Données** :
  - ensemble $\{a, b, c, d, e\}$ d'objets ;
  - p_i leur poids et u_i leur utilité.

Problème du sac à dos

- **Données** :
  - ensemble $\{a, b, c, d, e\}$ d'objets ;
  - p_i leur poids et u_i leur utilité.
- **variables de décision**

$$x_i = \begin{cases} 1 \text{ si l'objet } i \text{ est choisi} \\ 0 \text{ sinon} \end{cases}$$

Problème du sac à dos

- **Données** :
  - ensemble $\{a, b, c, d, e\}$ d'objets ;
  - p_i leur poids et u_i leur utilité.
- **variables de décision**

$$x_i = \begin{cases} 1 \text{ si l'objet } i \text{ est choisi} \\ 0 \text{ sinon} \end{cases}$$

- **Objectif**

$$\max u_a x_a + u_b x_b + u_c x_c + u_d x_d + u_e x_e$$

- **Contraintes**

$$p_a x_a + p_b x_b + p_c x_c + p_d x_d + p_e x_e \leq 22$$

Problème du sac à dos

Objets	a	b	c	d	e
poids	3	4	3	3	13
utilité	12	12	9	15	26

$$\begin{aligned} \max z \quad & 12x_a + 12x_b + 9x_c + 15x_d + 26x_e \\ \text{s.c.} \quad & 3x_a + 4x_b + 3x_c + 3x_d + 13x_e \leq 22 \end{aligned}$$

modélisation GLPK

Problème du sac à dos

```
$glpsol -m exemple2.mod
INTEGER OPTIMAL SOLUTION FOUND
Time used:   0.0 secs
Memory used: 0.1 Mb (148372 bytes)
x1=1.000000
x2=0.000000
x3=1.000000
x4=1.000000
x5=1.000000
z=62.000000
```

Problème du sac à dos

```
$glpsol -m exemple2.mod
INTEGER OPTIMAL SOLUTION FOUND
Time used:   0.0 secs
Memory used: 0.1 Mb (148372 bytes)
x1=1.000000
x2=0.000000
x3=1.000000
x4=1.000000
x5=1.000000
z=62.000000
```

Et pour d'autres instances ?

Problème du sac à dos

- **Données** :
  - ensemble $\{a, b, c, d, e\}$ d'objets ;
  - p_i leur poids et u_i leur utilité.
- **variables de décision**

$$x_i = \begin{cases} 1 \text{ si l'objet } i \text{ est choisi} \\ 0 \text{ sinon} \end{cases}$$

- **Objectif**

$$\max u_a x_a + u_b x_b + u_c x_c + u_d x_d + u_e x_e$$

- **Contraintes**

$$p_a x_a + p_b x_b + p_c x_c + p_d x_d + p_e x_e \leq 22$$

Problème du sac à dos

- **Données** :
  - ensemble $\{1, 2, 3, \cdots, n\}$ d'objets ;
  - p_i leur poids et u_i leur utilité ;
  - W : capacité du sac à dos
- **variables de décision**

$$x_i = \begin{cases} 1 \text{ si l'objet } i \text{ est choisi} \\ 0 \text{ sinon} \end{cases}$$

- **Objectif**

$$\max \sum_{i=1}^{n} u_i x_i$$

- **Contraintes**

$$\sum_{i=1}^{n} p_i x_i \leq W$$

Séparer les données et la modélisation

fichier de modélisation

fichier de données

```
$glpsol -m sac-a-dos.mod -d sac-a-dos.dat -o sac-a-dos.sol
```

fichier de solution

Sommaire

Sudoku

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

niveau facile

Sudoku

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

niveau difficile

Sudoku : modélisation

- **Contraintes**
  - un seul chiffre par case ;
  - au moins une fois chaque chiffre sur chaque ligne ;
  - au moins une fois chaque chiffre sur chaque colonne ;
  - au moins une fois chaque chiffre dans chaque block ;
  - au plus une fois chaque chiffre sur chaque ligne ;
  - au plus une fois chaque chiffre sur chaque colonne ;
  - au plus une fois chaque chiffre dans chaque block ;

Sudoku : modélisation

- **Contraintes**
  - un seul chiffre par case ;
  - au moins une fois chaque chiffre sur chaque ligne ;
  - au moins une fois chaque chiffre sur chaque colonne ;
  - au moins une fois chaque chiffre dans chaque block ;
  - au plus une fois chaque chiffre sur chaque ligne ;
  - au plus une fois chaque chiffre sur chaque colonne ;
  - au plus une fois chaque chiffre dans chaque block ;
- **Variables de décision** On utilise $9^3 = 729$ variables x_{ijk} définies par :

$$x_{ijk} = \begin{cases} 1 & \text{si la case à l'intersection de la ligne } i \\ & \text{et de la colonne } j \text{ est égale à } k \\ 0 & \text{sinon} \end{cases}$$

Sudoku : données

Une matrice T avec les cases fixées
T contient les cases de la grille dans lesquelles on a déjà mis un chifre de 1 à 9 :

$$T[i,j] = \text{nombre dans la case } (i,j)$$

```
param n_fix:=40; # nombre de cases fixées

param T: 1 2 3 :=
1   1 1 7
2   1 4 2
3   1 6 9
4   1 9 1
5   2 1 4
6   2 2 9
7   2 3 1
8   2 5 3
9   2 7 8
```

Sudoku : modélisation

$$x_{ijk} = \begin{cases} 1 & \text{si la case à l'intersection de la ligne } i \\ & \text{et de la colonne } j \text{ est égale à } k \\ 0 & \text{sinon} \end{cases}$$

```
#variables xijk=1 si case ij=k

var x{ i in 1..9, j in 1..9, k in 1.. 9 } binary;
```

Sudoku : modélisation

un seul chiffre par case

```
# un seul chiffre par case

valeurs {i in 1..9, j in 1.. 9}:
      sum {k in 1..9} x[i,j,k] = 1;
```

Sudoku : modélisation

au moins une fois chaque chiffre sur chaque ligne
au plus une fois chaque chiffre sur chaque ligne

```
# exactement chaque chiffre par ligne

lignes {i in 1..9, k in 1.. 9}:
      sum {j in 1..9} x[i,j,k] = 1;
```

Sudoku : modélisation

au moins une fois chaque chiffre sur chaque colonne
au plus une fois chaque chiffre sur chaque colonne

```
# exactement chaque chiffre par colonne

colonnes {j in 1..9, k in 1.. 9}:
      sum {i in 1..9} x[i,j,k] = 1;
```

Sudoku : modélisation

au moins une fois chaque chiffre dans chaque block
au plus une fois chaque chiffre dans chaque block

```
# exactement chaque chiffre par blocs de 3x3

blocs {xx in {0, 3, 6}, yy in {0, 3, 6}, k in 1..9}:
    sum {i in (xx+1)..(xx+3), j in (yy+1)..(yy+3)}
        x[i,j,k] = 1;
```

Sudoku : modélisation

tenir compte des cases déjà remplies

```
#cases données

C{i in 1..n_fix} :
     x[T[i,1], T[i,2], T[i,3]] = 1;
```

Sodoku : résolution

Fichier de modélisation

Fichier de modélisation (facile)

Fichier de modélisation (difficile)

```
$glpsol -m sudoku.mod -d Sudoku_data.dat
```

Sodoku : résolution

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

Sodoku : résolution

```
-------------------------------------
 7 | 3 | 6 | 2 | 8 | 9 | 5 | 4 | 1 |
-------------------------------------
 4 | 9 | 1 | 5 | 3 | 7 | 8 | 6 | 2 |
-------------------------------------
 2 | 8 | 5 | 4 | 6 | 1 | 3 | 9 | 7 |
-------------------------------------
 5 | 4 | 9 | 6 | 2 | 3 | 1 | 7 | 8 |
-------------------------------------
 3 | 6 | 2 | 7 | 1 | 8 | 9 | 5 | 4 |
-------------------------------------
 8 | 1 | 7 | 9 | 5 | 4 | 6 | 2 | 3 |
-------------------------------------
 9 | 5 | 3 | 8 | 7 | 2 | 4 | 1 | 6 |
-------------------------------------
 6 | 7 | 8 | 1 | 4 | 5 | 2 | 3 | 9 |
-------------------------------------
 1 | 2 | 4 | 3 | 9 | 6 | 7 | 8 | 5 |
-------------------------------------
Model has been successfully processed
```

Sodoku : résolution

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

Sodoku : résolution

```
Solution :
-----------------------------------
 7 | 9 | 8 | 6 | 3 | 5 | 4 | 2 | 1 |
-----------------------------------
 1 | 2 | 6 | 9 | 7 | 4 | 5 | 8 | 3 |
-----------------------------------
 4 | 5 | 3 | 2 | 1 | 8 | 6 | 7 | 9 |
-----------------------------------
 9 | 7 | 2 | 5 | 8 | 6 | 3 | 1 | 4 |
-----------------------------------
 5 | 6 | 4 | 1 | 2 | 3 | 8 | 9 | 7 |
-----------------------------------
 3 | 8 | 1 | 4 | 9 | 7 | 2 | 5 | 6 |
-----------------------------------
 6 | 1 | 7 | 3 | 5 | 2 | 9 | 4 | 8 |
-----------------------------------
 8 | 3 | 5 | 7 | 4 | 9 | 1 | 6 | 2 |
-----------------------------------
 2 | 4 | 9 | 8 | 6 | 1 | 7 | 3 | 5 |
-----------------------------------
Model has been successfully processed
```

Sommaire

Problème de production

Une usine produit deux composantes A et B d'un moteur d'avion.
La direction veut planifier sa production pour les 3 prochains mois.

Problème de production

Notification des besoins pour les trois prochains mois.

	avril	mai	juin
A	1 000	3 000	5 000
B	1 000	500	3 000

Problème de production

Capacités mensuelles

	machine (h)	hommes (h)	stock (m^3)
avril	400	300	10 000
mai	500	300	10 000
juin	600	300	10 000

Problème de production

Capacités par unité de production

	machine (h/unité)	homme (h/unité)	stock (m^3/unité)
A	0,10	0,05	2
B	0,08	0,07	3

Problème de production

Informations complémentaires

- coûts de production : 20 par unités de A et 10 par unités de B ;
- coût de stockage : 1,5 % de la valeur ;
- horaire mensuel de base : 225 ;
- coût de l'heure supplémentaire de travail : 10 ;
- stock fin mars : 500 A et 200 B ;
- stock minimum imposé fin juin : 400 A et 200 B.

Problème de production

Problématique

Trouver un plan de production des trois prochain mois qui minimise les coûts.

Problème de production

Variables

- production : `x[produit, mois]` ;
- stock : `s[produit, mois]` ;
- heures supplémentaires : `l[mois]` .

Combien de variables ?

Problème de production

Variables

- production : `x[produit, mois]` ;
- stock : `s[produit, mois]` ;
- heures supplémentaires : `l[mois]` .

Combien de variables ?

15 variables

Problème de production

Objectif

$$\min(\text{production} + \text{stock} + \text{heures supplémentaires})$$

Problème de production

Contraintes

- définition du stock
- stock minimum fin juin
- capacités des machines
- capacités des hommes
- capacités des stocks
- définition des heures supplémentaires

Combien de contraintes ?

Problème de production

Contraintes

- définition du stock
- stock minimum fin juin
- capacités des machines
- capacités des hommes
- capacités des stocks
- définition des heures supplémentaires

Combien de contraintes ?

20 contraintes

Problème de production

Données

```
## Production de moteurs d''avions
##   GLPK  Fichier de modélisation

set produit;
set mois;

param stock_init{i in produit};
param stock_final{i in produit};
param besoin{i in produit, j in mois};
param capacite_mois{i in mois, j in 1..3};
param capacite_heure{i in produit, j in 1..3};
```

Problème de production

Variables

```
var x{i in produit, j in mois}>=0;
var s{i in produit, j in mois}>=0;
var l{i in mois}>=0;
```

Objectif

```
minimize cout:
  20*(sum{j in mois} x['A',j]) +
  10*(sum{j in mois} x['B',j]) +
  0.015*(20*(sum{j in mois} s['A',j]) +
  10*(sum{j in mois} s['B',j])) +
  sum{i in mois} 10*l[i];
```

Problème de production

Réolution

```
Objective:  cout = 224724.2857 (MINimum)

   No. Column name  St    Activity
------ ------------ -- -------------
     1 x[B,avril]   B         2857.14
     2 x[B,mai]     B         1214.29
     3 x[B,juin]    B         428.571
     4 x[A,avril]   B             500
     5 x[A,mai]     B            3000
     6 x[A,juin]    B            5400
     7 s[B,avril]   B         2057.14
     8 s[B,mai]     B         2771.43
     9 s[B,juin]    B             200
    10 s[A,avril]   NL              0
    11 s[A,mai]     NL              0
    12 s[A,juin]    B             400
```