Optimisation en nombres entiers

Daniel Porumbel

Contributions aux slides:
Cédric Bentz
Lê Nguyên Hoang

Plan

1. Difficulté des problèmes de nature discrète et explosion combinatoire
2. Méthodes de résolution en optimisation discrète
3. Exemples problèmes résolus avec correction (code)

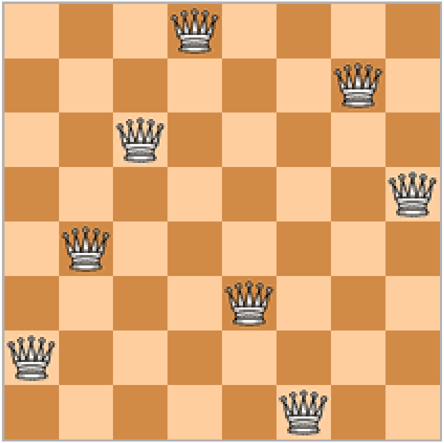
Comment placer n dames sur un échiquier de $n \times n$ cases sans s'attaquer mutuellement ? $\boxed{\text{Considérons } n = 4}$

- Une solution pour $n = 8 \longrightarrow$
- Idée 2 (résolution complète mais naïve) : vérifier n^n solutions candidates !!!
  - Même $\boxed{n = 4}$ demande 256 solutions
    - voir vidéos pour $n = 8$

Un problème de nature discrète : les *n* dames

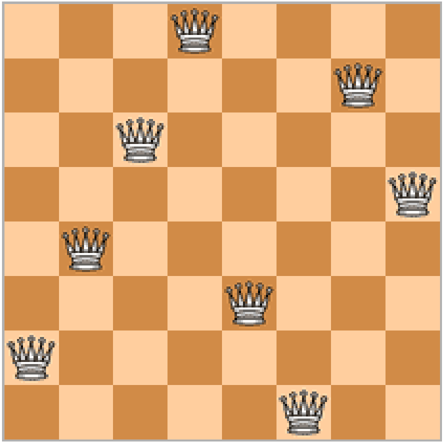
Comment placer n dames sur un échiquier de $n \times n$ cases sans s'attaquer mutuellement ? $\boxed{\text{Considérons } n = 4}$

- Une solution pour $n = 8 \longrightarrow$
  - Idée 1 : placer n dames n'importe où et essayer de réparer étape par étape. Cela peut mener à une solution ou pas...
- Idée 2 (résolution complète mais naïve) : vérifier n^n solutions candidates ! ! !
  - Même $\boxed{n = 4}$ demande 256 solutions
    - voir vidéos pour $n = 8$



Comment placer n dames sur un échiquier de $n \times n$ cases sans s'attaquer mutuellement ? $\boxed{\text{Considérons } n = 4}$

- Une solution pour $n = 8 \longrightarrow$
- Idée 2 (résolution complète mais naïve) : vérifier n^n solutions candidates ! ! !
  - Même $\boxed{n = 4}$ demande 256 solutions
    - voir vidéos pour $n = 8$



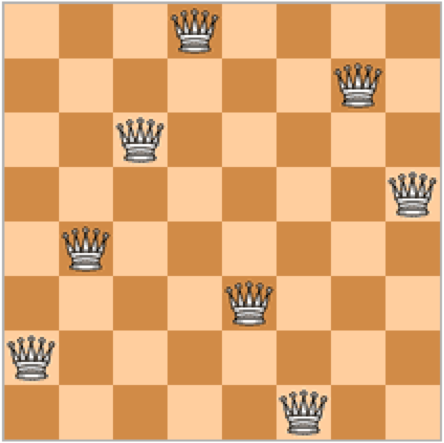
Explosion combinatoire

nombre exponentiel de solutions (possibilités) envisageables.

Un problème de nature discrète : les *n* dames

Comment placer n dames sur un échiquier de $n \times n$ cases sans s'attaquer mutuellement ? $\boxed{\text{Considérons } n = 4}$

- Une solution pour $n = 8 \longrightarrow$
- Idée 2 (résolution complète mais naïve) : vérifier n^n solutions candidates ! ! !
  - Même $\boxed{n = 4}$ demande 256 solutions
    - voir vidéos pour $n = 8$

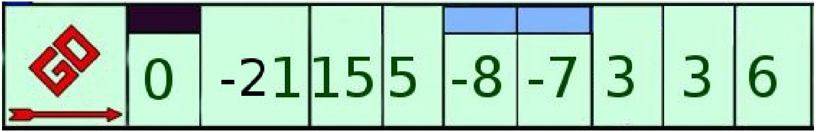


Explosion combinatoire

nombre exponentiel de solutions (possibilités) envisageables.

Un Jeu : Décision Optimale de Sauts

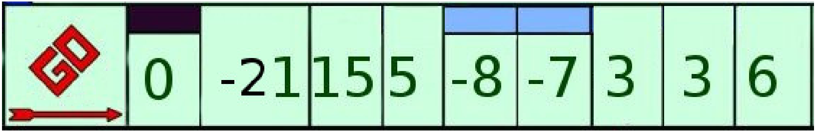
On considère le début d'un jeu de type *Monopoly* :



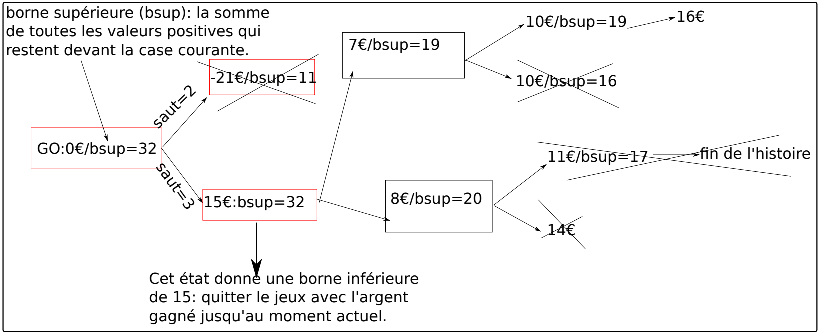
On avance vers à l'aide des opérations suivantes :

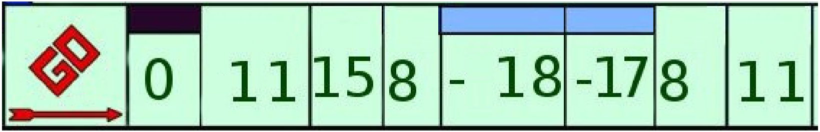
- avancer 2 cases
- avancer 3 cases
- quitter avec l'argent déjà récupéré

Comment calculer le profit maximal (noté OPT) ?


GO
0
-21
15
5
-8
-7
3
3
6

Voici l'arbre de branchement (flèche vers le haut=saut de 2, flèche vers la bas=saut de 3)





Un exercice avec un jeu plus petit.

Comment dérober le maximum ?

- Un cambrioleur arrive à s'introduire dans une banque

Il peut voler

- des barres d'or et
- des liasses de billets

Problème : 2 limitations de son sac-à-dos

- charge max : 20kg
- volume max : 32 litres

une barre d'or : 300000\$, 8kg, 6litres

un paquet de billets : 100000\$, 3kg, 6litres

Comment dérober le maximum ?

- Un cambrioleur arrive à s'introduire dans une banque

Il peut voler

- des barres d'or et
- des liasses de billets

Problème : 2 limitations de son sac-à-dos

- charge max : 20kg
- volume max : 32 litres

une barre d'or : 300000\$, 8kg, 6litres

un paquet de billets : 100000\$, 3kg, 6litres

Comment dérober le maximum ?

- Un cambrioleur arrive à s'introduire dans une banque

Il peut voler

- des barres d'or et
- des liasses de billets

Problème : 2 limitations de son sac-à-dos

- charge max : 20kg
- volume max : 32 litres

une barre d'or : 300000$, 8kg, 6litres

un paquet de billets : 100000$, 3kg, 6litres

Comment dérober le maximum ?

- Un cambrioleur arrive à s’introduire dans une banque

Il peut voler

- des barres d’or et
- des liasses de billets

Problème : 2 limitations de son sac-à-dos

- charge max : 20kg
- volume max : 32 litres

une barre d’or : 300000\$, 8kg, 6litres
un paquet de billets : 100000\$, 3kg, 6litres

Et si on faisait l’effort de porter 1kg de plus ? Ou 2kg ?

Un programme linéaire en nombres entiers

variable x_1 nombre de barres d'or (300000\$, 8kg, 6l)

variable x_2 nombre de paquets de billets (100000\$, 3kg, 6litres)

$$\begin{array}{cl}
\max 3x_1 + x_2 & \leftarrow \text{maximiser le profit} \\
\left.\begin{array}{r} 8x_1 + 3x_2 \leq 20 \\ 6x_1 + 6x_2 \leq 32 \end{array}\right\} & \leftarrow \text{Contraintes de poids et volume} \\
x_1, x_2 \in \mathbb{Z}_+ & \leftarrow \text{Contraintes d'intégralité}
\end{array}$$

Un simple exemple `glpk`

```
var x1>=0;
var x2>=0;
maximize obj:3*x1+x2;
subject to c1: 8*x1+3*x2<=20;
subject to c2: 6*x1+6*x2<=32;
solve;
display x1,x2;
end;
```

Vous pouvez l'exécuter à
`cocoto.github.io/glpk-online/`

Pour passer en nombres entiers, on déclare :

`var x1>=0,integer;`

Un simple exemple `glpk`

```
var x1>=0;
var x2>=0;
maximize obj:3*x1+x2;
subject to c1: 8*x1+3*x2<=20;
subject to c2: 6*x1+6*x2<=32;
solve;
display x1,x2;
end;
```

Vous pouvez l'exécuter à
`cocoto.github.io/glpk-online/`
Pour passer en nombres entiers, on déclare :
`var x1>=0,integer;`

Le même modele en `julia` 1

```
using JuMP;
using GLPK;                      #ou "using Cbc"
m = Model(GLPK.Optimizer);       #ou "Cbc.Optimizer"

# Define the variables
@variable(m, x>=0, Int);
@variable(m, y>=0, Int);

# La fonction objectif
@objective(m, Max, 3x+y);

# Les contraintes
@constraint(m, 8x+3y<=20);
@constraint(m, 6x+6y<=32);
```

. . . , à suivre slide suivant, . . .

Le même modele en julia 2

Une fois un modèle m défini, on l'optimise ainsi :

```
# Faire tourner le solveur
optimize!(m);

# Output
println(objective_value(m))   # val opt obj
println("x = ", value.(x),    # val opt x
        "\n",                 # saut de ligne
       "y = ", value.(y))     # val opt y
```

Que pensez vous du problème suivant ?

- Vous êtes un restaurateur de luxe, mais vous n'avez que 20 chaises et 5 tables. Vous pouvez offrir des tables de 8 personnes ou de 3 personnes.
- Une table de 8 vous apporte 3000 euros
- Une table de 3 vous apporte 1000 euros

$$\max 3 \cdot x_1 + x_2$$
$$8x_1 + 3x_2 \leq 20$$
$$x_1 + x_2 \leq 5$$
$$x_1, x_2 \in \mathbb{Z}_+$$

Peut-on gagner plus d'argent si on apporte encore une chaise (ou deux) ?

Que pensez vous du problème suivant ?

- Vous êtes un restaurateur de luxe, mais vous n'avez que 20 chaises et 5 tables. Vous pouvez offrir des tables de 8 personnes ou de 3 personnes.
- Une table de 8 vous apporte 3000 euros
- Une table de 3 vous apporte 1000 euros

$$\max 3 \cdot x_1 + x_2$$
$$8x_1 + 3x_2 \leq 20$$
$$x_1 + x_2 \leq 5$$
$$x_1, x_2 \in \mathbb{Z}_+$$

Peut-on gagner plus d'argent si on apporte encore une chaise (ou deux) ?

Que pensez vous du problème suivant ?

- Vous êtes un restaurateur de luxe, mais vous n'avez que 20 chaises et 5 tables. Vous pouvez offrir des tables de 8 personnes ou de 3 personnes.
- Une table de 8 vous apporte 3000 euros
- Une table de 3 vous apporte 1000 euros

$$\max 3 \cdot x_1 + x_2$$
$$8x_1 + 3x_2 \leq 20$$
$$x_1 + x_2 \leq 5$$
$$x_1, x_2 \in \mathbb{Z}_+$$

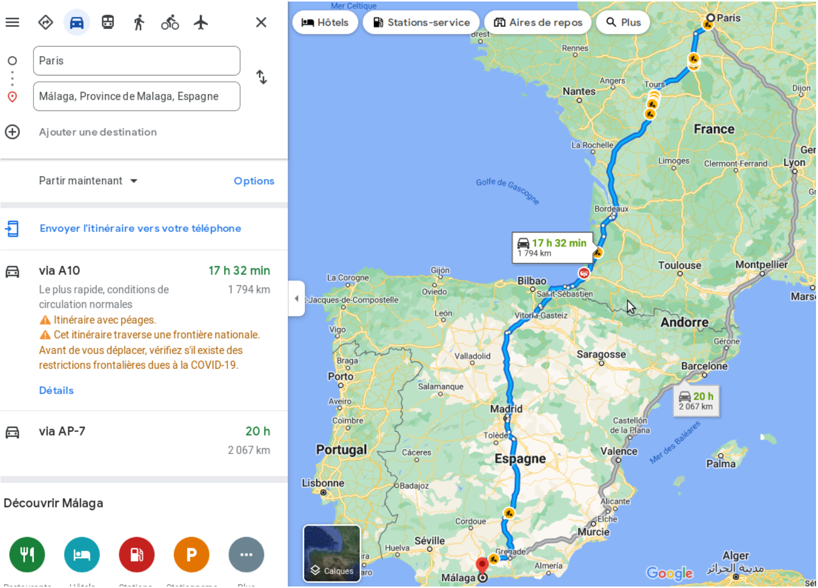
Peut-on gagner plus d'argent si on apporte encore une chaise (ou deux) ?

Que pensez vous du problème suivant ?

- Vous êtes un restaurateur de luxe, mais vous n'avez que 20 chaises et 5 tables. Vous pouvez offrir des tables de 8 personnes ou de 3 personnes.
- Une table de 8 vous apporte 3000 euros
- Une table de 3 vous apporte 1000 euros

$$\max 3 \cdot x_1 + x_2$$
$$8x_1 + 3x_2 \leq 20$$
$$x_1 + x_2 \leq 5$$
$$x_1, x_2 \in \mathbb{Z}_+$$

Peut-on gagner plus d'argent si on apporte encore une chaise (ou deux) ?


Paris
Málaga, Province de Malaga, Espagne
Ajouter une destination
Partir maintenant
Options
Envoyer l'itinéraire vers votre téléphone
via A10
17 h 32 min
Le plus rapide, conditions de circulation normales
1 794 km
Itinéraire avec péages.
Cet itinéraire traverse une frontière nationale. Avant de vous déplacer, vérifiez s'il existe des restrictions frontalières dues à la COVID-19.
Détails
via AP-7
20 h
2 067 km
Découvrir Málaga
Hôtels
Stations-service
Aires de repos
Plus
Mer Celtique
Paris
Rennes
Nantes
Angers
Tours
France
La Rochelle
Limoges
Clermont-Ferrand
Lyon
Dijon
Golfe de Gascogne
Bordeaux
17 h 32 min
1 794 km
Toulouse
Montpellier
Bilbao
Saint-Sébastien
Vitoria-Gasteiz
Andorre
Gérone
Barcelone
La Corogne
Gijón
Oviedo
León
Vigo
Braga
Porto
Valladolid
Saragosse
Salamanque
Aveiro
Coimbre
Madrid
Tolède
20 h
2 067 km
Castellón de la Plana
Valence
Mer des Baléares
Palma
Portugal
Cáceres
Espagne
Lisbonne
Badajoz
Alicante
Elche
Murcie
Cordoue
Séville
Huelva
Grenade
Almería
Málaga
Calques
Alger
مدينة الجزائر
Google

```
Le chemin le plus court vers Malaga
         P------+          Variables:
      /  |      L                   xPT, xPL, xPB, xTV, xLV, xVM, xBM
   /     |      |          km:      600  400  500  500  800  300  700
 B       |      |          péages:   0   60    40   50   0     0   30
  |      T      |          mer:      0    0    50   0   300  300  100
   |       \    |
    |        \  |
     |         V
      |      /
        \  /
         M

min 600*xPT+400*xPL+ 500*xPB+500*xTV+800*xLV+300*xVM+700*xBM
    60* xPL+         40*xPB+ 50*xTV+                30*xBM <= 90euros
                     50*xPB+       300*xLV+300*xVM+100*xBM >= 200
    xPT+xPL+xPB = 1
    xVM+xBM     = 1
    xLV + xTV = xVM     xPB       = xBM
    xPT       = xTV     xPL       = xLV
    toutes les variables x: binaires
```

```
using JuMP;                  #Le chemin le plus court vers Malaga
using GLPK;                  #         P------+  Variables:
m = Model(GLPK.Optimizer);#      /  |      L              xPT, xPL, xPB, xTV, xLV, xVM, xBM
                             #   /     |      |  km:        600  400  500  500  800  300  700
@variable(m, xPT,Bin);       # B       |      |  péages:     0   60    40   50   0      0   30
@variable(m, xPL,Bin);       #  |      T      |
@variable(m, xPB,Bin);       #   |       \    |
@variable(m, xTV,Bin);       #    |        \  |
@variable(m, xLV,Bin);       #     |         V
@variable(m, xVM,Bin);       #      |     /
@variable(m, xBM,Bin);       #       \   /
                             #         M
@objective(m, Min, 600*xPT+400*xPL+ 500*xPB+500*xTV+800*xLV+300*xVM+700*xBM);

@constraint(m, 60* xPL+    40*xPB+ 50*xTV+                   30*xBM <= 50);
@constraint(m, xPT+xPL+xPB == 1);
@constraint(m, xVM+xBM     == 1);
@constraint(m, xLV + xTV == xVM);
@constraint(m, xPT        == xTV);
@constraint(m, xPB        == xBM);
@constraint(m, xPL        == xLV);
optimize!(m);
println(objective_value(m));
println("xPT=",value.(xPT));
println("xPL=",value.(xPL));
println("xPB=",value.(xPB));
println("xTV=",value.(xTV));
println("xLV=",value.(xLV));
println("xVM=",value.(xVM));
println("xBM=",value.(xBM));

termination_status(m);
```

Vers l’optimisation à grande échelle

- Le problème du cambrioleur : 2 variables
- Mais : souvent besoin d’un tableau de n ou n^2 variables
- Ex : Soit n articles et un tableau de prix, ex $a = [1, 10, 11, 3]$
  - pas le droit de prendre deux articles consecutifs ;
  - quels articles choisir pour maximiser le prix total ?

```
param n, >0, integer;
param a{1..n}, integer;
var x{1..n}, >=0, binary;
s.t. c_consec{i in 1..n-1}: x[i]+x[i+1]<=1;
maximize obj: sum{i in 1..n} a[i]*x[i];
solve;
for {i in 1..n}{ printf "%d ", x[i]; }
data;
param n:=4;
param a:= 1 1
          2 10
          3 11
          4 3 ;
end;
```

Vers l’optimisation à grande échelle

- Le problème du cambrioleur : 2 variables
- Mais : souvent besoin d’un tableau de n ou n^2 variables
- Ex : Soit n articles et un tableau de prix, ex $a = [1, 10, 11, 3]$
  - pas le droit de prendre deux articles consecutifs ;
  - quels articles choisir pour maximiser le prix total ?

```
param n, >0, integer;
param a{1..n}, integer;
var x{1..n}, >=0, binary;
s.t. c_consec{i in 1..n-1}: x[i]+x[i+1]<=1;
maximize obj: sum{i in 1..n} a[i]*x[i];
solve;
for {i in 1..n}{ printf "%d ", x[i]; }
data;
param n:=4;
param a:= 1 1
          2 10
          3 11
          4 3 ;
end;
```

```julia
using JuMP;
using GLPK;
m = Model(GLPK.Optimizer);

n = 4;                         #on a 4 articles
a = [1, 10, 11, 3];            #avec 4 profits

@variable(m, x[1:n], Bin);#n variables binaires

@objective(m, Max, sum(a[i]*x[i] for i in 1:n))

@constraint(m, [i in 1:n-1], x[i]+x[i+1] <= 1)

optimize!(m);                   # on optimize!
println(objective_value(m)) # opt obj val
println("x = ", value.(x))  # opt x
```

Le modèle pour les n dames

au tableau

Le modèle des *n* dames en Julia

```
using JuMP
using GLPK
m = Model(GLPK.Optimizer);
n=8
@variable(m, x[1:n,1:n], Bin)
@objective(m, Max, sum( x[i,j] for i in 1:n, j in 1:n))
for j in 1:n
   @constraint(m, sum(x[i,j] for i in 1:n) == 1)
end
for i in 1:n
   @constraint(m, sum(x[i,j] for j in 1:n) == 1)
end
for k in 2:2*n
   @constraint(m,sum(x[i,j] for i in 1:n, j in 1:n if i+j == k) <= 1)
end
for k in -(n-1):(n-1)
   @constraint(m,sum(x[i,j] for i in 1:n, j in 1:n if i-j == k) <= 1)
end
optimize!(m)
display(Int.(value.(x)))  #chagement du type en int avant l'affichage
```

Plan

1. Difficulté des problèmes de nature discrète et explosion combinatoire
2. Méthodes de résolution en optimisation discrète
3. Exemples problèmes résolus avec correction (code)

1. Difficulté des problèmes de nature discrète et explosion combinatoire

2. Méthodes de résolution en optimisation discrète
   - La relaxation linéaire et les bornes associées
   - Méthode à base de coupes
   - L’algorithme Branch-and-Bound (séparation et évaluation)

3. Exemples problèmes résolus avec correction (code)

Rappel problème

Rappels caractéristiques

		Prix,	Poids,	Volume
or	$3\times$	10000\$,	8kg,	6litres
billets	$1\times$	10000\$,	3kg,	6litres
max :			20kg,	32l

Solution heuristique : l'or d'abord

- $2 \times$ or $+ 1 \times$billets
- Profit$= 2 \times 3 + 1 \times 1 = 7$

Rappel problème

Rappels caractéristiques

	Prix,	Poids, Volume
• or	$3\times$ 10000\$,	8kg, 6litres
• billets	$1\times$ 10000\$,	3kg, 6litres
• max :		20kg, 32l

Solution heuristique : l'or d'abord

- $2 \times$ or + $1\times$billets
- Profit$= 2 \times 3 + 1 \times 1 = 7$

Relaxation linéaire : Intuition

Rappels caractéristiques

		Prix,	Poids,	Volume
•	or	$3\times$ 10000\$,	8kg,	6litres
•	billets	$1\times$ 10000\$,	3kg,	6litres
•	max :		20kg,	32l

Solution heuristique : l'or d'abord

- $2 \times$ or $+ 1\times$billets$\rightarrow$ $7\times$10000\$

Peut-il espérer un profit beaucoup plus important que 7 ?

- Si on pouvait fondre l'or, la solution optimale serait de prendre le maximum d'or : 2.5 lingots !

$\Longrightarrow$ Valeur optimale relaxée : 7.5

$\Longrightarrow$ Le profit de 7 est optimal en nombre entiers !

Relaxation linéaire : Intuition

Rappels caractéristiques

		Prix,	Poids,	Volume
• or	$3\times$	10000\$,	8kg,	6litres
• billets	$1\times$	10000\$,	3kg,	6litres
• max :			20kg,	32l

Solution heuristique : l'or d'abord

- $2 \times$ or + $1 \times$billets$\rightarrow$ $7\times$10000\$

Peut-il espérer un profit beaucoup plus important que 7 ?

- Si on pouvait fondre l'or, la solution optimale serait de prendre $\boxed{\text{le maximum d'or}}$: 2.5 lingots !

$\Longrightarrow$ Valeur optimale relaxée : 7.5

$\Longrightarrow$ Le profit de 7 est optimal en nombre entiers !

Relaxation linéaire : Intuition

Rappels caractéristiques

		Prix,	Poids, Volume
•	or	$3\times$ 10000\$,	8kg, 6litres
•	billets	$1\times$ 10000\$,	3kg, 6litres
•	max :		20kg, 32l

Solution heuristique : l'or d'abord

- $2 \times$ or + $1 \times$ billets $\rightarrow$ $7 \times 10000\$$

Peut-il espérer un profit beaucoup plus important que 7 ?

- Si on pouvait fondre l'or, la solution optimale serait de prendre $\boxed{\text{le maximum d'or}}$: 2.5 lingots !

$\Longrightarrow$ Valeur optimale relaxée : 7.5

$\Longrightarrow$ Le profit de 7 est optimal en nombre entiers !

$$
\begin{array}{rl}
\max 3x_1 + x_2 & \leftarrow \text{prix : or=300000\$ billets :100000\$} \\
8x_1 + 3x_2 \leq 20 & \leftarrow \text{Limite de poids} \\
6x_1 + 6x_2 \leq 32 & \leftarrow \text{Limite de volume} \\
x_1, x_2 \in \mathbb{Z}_+ & \leftarrow \text{Contraintes d'intégralité}
\end{array}
$$

- Ce type de programmes est difficile en nombre entiers à cause de l'expositon combinatoire !

  Mais il serait facile si les variables n'étaient $\boxed{\text{pas}}$ entières.

$$
\begin{array}{rl}
\max 3x_1 + x_2 & \\
8x_1 + 3x_2 \leq 20 & \\
6x_1 + 6x_2 \leq 32 & \\
x_1, x_2 \in \mathbb{R}_+ & \leftarrow \text{Pas d'intégralité} \implies \text{Facile !}
\end{array}
$$

Optimum fractionnaire → optimum entier ? ?

$$\max 3x_1 + x_2$$
$$8x_1 + 3x_2 \leq 20$$
$$6x_1 + 6x_2 \leq 32$$
$$x_1, x_2 \in \mathbb{Z}_+$$

Les points rouges sont les solutions en nb entiers.
Le carré rouge est la solution optimale du programme continu ($x_1, x_2 \in \mathbb{R}_+$).

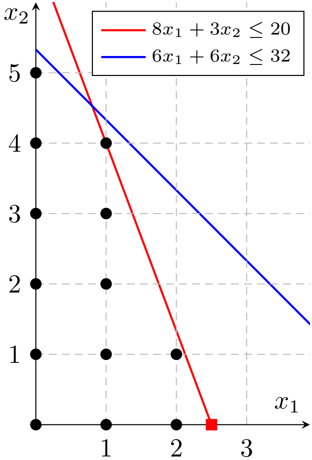
De l'optimum fractionnaire vers une solution entière

Trouver l'optimum fractionnaire (continu) et prendre le point entier le plus proche $\implies (x_1, x_2) = (2, 0)$, pas optimal en nb ent.

Optimum fractionnaire → optimum entier ? ?

$$\max 3x_1 + x_2$$
$$8x_1 + 3x_2 \leq 20$$
$$6x_1 + 6x_2 \leq 32$$
$$x_1, x_2 \in \mathbb{Z}_+$$

Les points rouges sont les solutions en nb entiers.
Le carré rouge est la solution optimale du programme continu ($x_1, x_2 \in \mathbb{R}_+$).



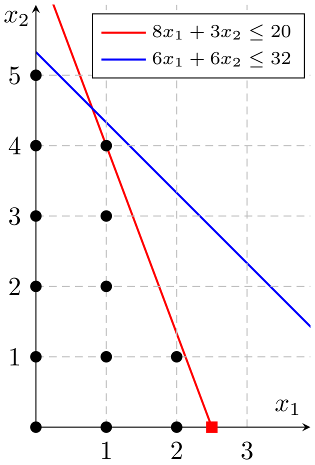
De l'optimum fractionnaire vers une solution entière

Trouver l'optimum fractionnaire (continu) et prendre le point entier le plus proche $\Longrightarrow$ $(x_1, x_2) = (2, 0)$, pas optimal en nb ent.

Optimum fractionnaire → optimum entier ? ?

$$\max 3x_1 + x_2$$
$$8x_1 + 3x_2 \leq 20$$
$$6x_1 + 6x_2 \leq 32$$
$$x_1, x_2 \in \mathbb{Z}_+$$

Les points rouges sont les solutions en nb entiers.
Le carré rouge est la solution optimale du programme continu ($x_1, x_2 \in \mathbb{R}_+$).



De l'optimum fractionnaire vers une solution entière

Trouver l'optimum fractionnaire (continu) et prendre le point entier le plus proche $\implies$ $(x_1, x_2) = (2, 0)$, pas optimal en nb ent.

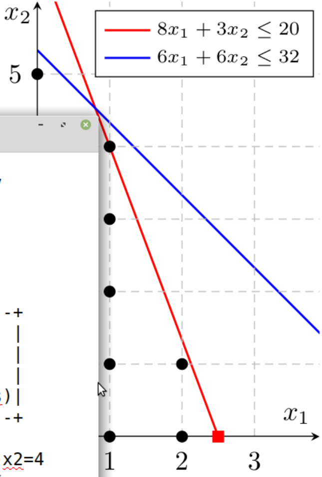
Branch-and bounds in advance

$$\max 3x_1 + x_2$$
$$8x_1 + 3x_2 \leq 20$$
$$6x_1 + 6x_2 \leq 32$$

[No Name] + - GVIM8

File Edit Tools Syntax Buffers Window Help

```
           +---------------------+
           |racine Branch&Bound |              borneInf = 7
           |borneSup 7.5:(2.5,0)|
           +---------------------+
          /          x1=0|          \
   x1=2  /               |           \ x1=1
        /                |            \
+------------+   +------------+   +------------------------+
|max 6+x2    |   |max x2      |   |max 3+ x2               |
|    x2<=1.33|   |    x2<=6.33|   |       x2<=4            |
|    x2<=3.33|   |    x2<=5.33|   |       x2<=4.33         |
|sol:x2 =1.33|   |sol x2 =5.33|   |sol    x2 =4 (en nb réels)|
+------------+   |opt relaché |   +------------------------+
      /   \      |5.33 < bInf |   |     |     |     |     |
x2=0/      \x2=1 +------------+   |x2=0 |x2=1 |x2=2 |x2=3 |x2=4
    /        \                    *     *     *     *     *
  6           7                   3     4     5     6     7
```



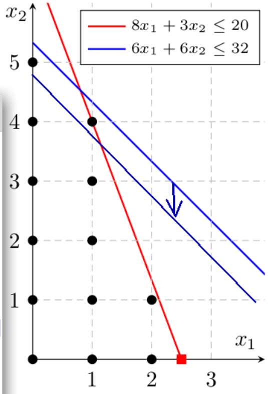
Et si on changeait 32 en 29 ?

$$\max 3x_1 + x_2$$
$$8x_1 + 3x_2 \leq 20$$
$$6x_1 + 6x_2 \leq \cancel{32}\ 29$$

[No Name] + - GVIM8

File Edit Tools Syntax Buffers Window Help

```
         +---------------------+
         |racine Branch&Bound |                    borneInf = 7
         |borneSup 7.5:(2.5,0)|
         +---------------------+
           /          x1=0|         \
  x1=2  /                 |          \ x1=1
       /                  |            \
+-----------+    +-----------+    +-----------------------+
|max  6+x2  |    |max  x2    |    |max  3+x2   Opt réels 6.83 |
|    x2<=1.33|   |    x2<=6.33|   |    x2<=4            < 7 |
|    x2<=3.33|   |    x2<=5.33|   |    x2<=3.83             |
|sol:x2 =1.33|   |sol  x2 =5.33|  |sol  x2 =3.83 (en nb réels)|
+-----------+    |opt relaché |   ------------------------
      /    \     |5.33 < bInf |
x2=0/      \x2=1 +-----------+        branche coupée
    /         \
   6           7!!!
~
```



1. Difficulté des problèmes de nature discrète et explosion combinatoire

2. Méthodes de résolution en optimisation discrète
   - La relaxation linéaire et les bornes associées
   - Méthode à base de coupes
   - L'algorithme Branch-and-Bound (séparation et évaluation)

3. Exemples problèmes résolus avec correction (code)

Meilleure modélisation à base de coupes 1

Rappel contraintes :

$$8x_1 + 3x_2 \leq 20$$

$$6x_1 + 6x_2 \leq 32$$

Et si on divisait la première contrainte par 8 et la deuxième par 6 ?

$$x_1 + 0.375x_2 \leq 2.5$$

$$x_1 + x_2 \leq 5.333$$

Si on prend la partie entière de chaque terme :

$$x_1 \leq 2$$

$$x_1 + x_2 \leq 5$$

Meilleure modélisation à base de coupes 1

Rappel contraintes :

$$8x_1 + 3x_2 \leq 20$$

$$6x_1 + 6x_2 \leq 32$$

Et si on divisait la première contrainte par 8 et la deuxième par 6 ?

$$x_1 + 0.375x_2 \leq 2.5$$

$$x_1 + x_2 \leq 5.333$$

Si on prend la partie entière de chaque terme :

$$x_1 \leq 2$$

$$x_1 + x_2 \leq 5$$

Meilleure modélisation à base de coupes 1

Rappel contraintes :

$$8x_1 + 3x_2 \leq 20$$

$$6x_1 + 6x_2 \leq 32$$

Et si on divisait la première contrainte par 8 et la deuxième par 6 ?

$$x_1 + 0.375x_2 \leq 2.5$$

$$x_1 + x_2 \leq 5.333$$

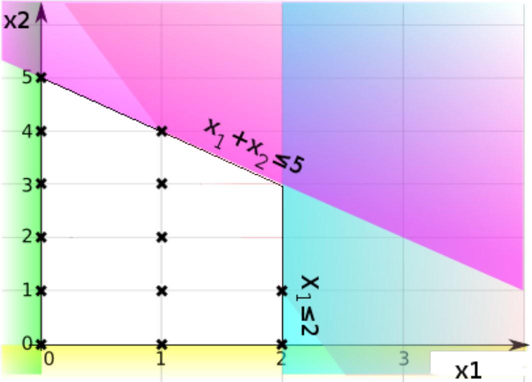
Si on prend la partie entière de chaque terme :

$$x_1 \leq 2$$

$$x_1 + x_2 \leq 5$$

Voici la zone réalisable avec ces deux contraintes

$$\max \ldots$$
$$x_1 \leq 2$$
$$x_1 + x_2 \leq 5$$
$$x_1, x_2 \in \mathbb{R}_+$$



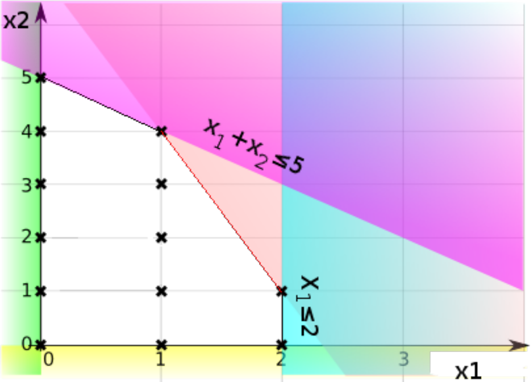
Peut-on ajouter une contrainte intelligente (coupe) pour décrire cette $\boxed{\text{enveloppe convexe}}$ des points entiers ?

Astuce $a \cdot x_1 + b \cdot x_2 = const$ en (2,1) et (1,4) $\implies 3x_1 + x_2 \leq 7$

Meilleure modélisation à base de coupes 2

Quel est l'avantage d'une zone réalisable comme ci-dessous ?

$$\max \ldots$$
$$x_1 \leq 2$$
$$x_1 + x_2 \leq 5$$
$$x_1, x_2 \in \mathbb{R}_+$$

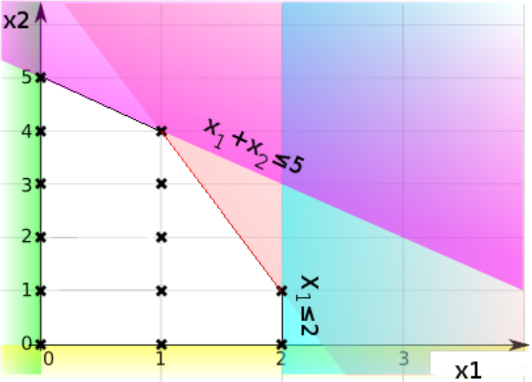


Peut-on ajouter une contrainte intelligente (coupe) pour décrire cette $\boxed{\text{enveloppe convexe}}$ des points entiers ?

Astuce $a \cdot x_1 + b \cdot x_2 = const$ en (2,1) et (1,4) $\implies 3x_1 + x_2 \leq 7$

Meilleure modélisation à base de coupes 2

$$\max \ldots$$
$$x_1 \leq 2$$
$$x_1 + x_2 \leq 5$$
$$x_1, x_2 \in \mathbb{R}_+$$

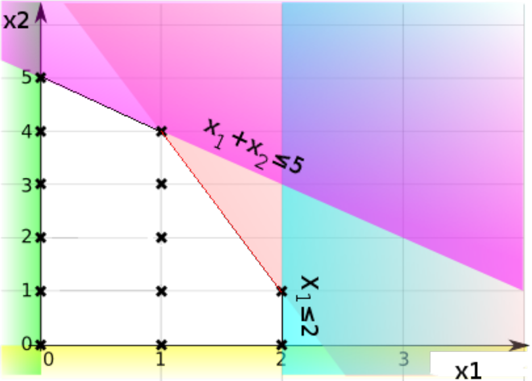


Peut-on ajouter une contrainte intelligente (coupe) pour décrire cette $\boxed{\text{enveloppe convexe}}$ des points entiers ?

Astuce $a \cdot x_1 + b \cdot x_2 = const$ en (2,1) et (1,4) $\implies 3x_1 + x_2 \leq 7$

Meilleure modélisation à base de coupes 2

$$\begin{aligned} &\max \ldots \\ x_1 &\leq 2 \\ x_1 + x_2 &\leq 5 \\ {\color{blue}3x_1 + x_2} &{\color{blue}\leq 7} \end{aligned}$$



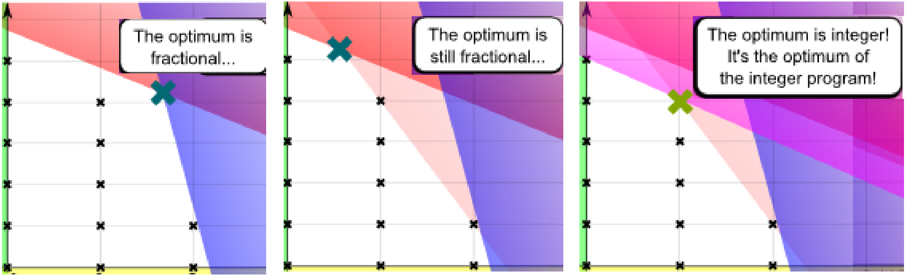
Peut-on ajouter une contrainte intelligente (coupe) pour décrire cette $\boxed{\text{enveloppe convexe}}$ des points entiers ?

Astuce $a \cdot x_1 + b \cdot x_2 = const$ en (2,1) et (1,4) $\implies 3x_1 + x_2 \leq 7$

Après avoir ajouté une première coupe, l'optimum fractionnaire n'est pas forcement entier. $\implies$ On peut itérer le processus :

Exemple d'exécution de la méthode des `Plans Coupants` :

1. L'optimum de la relaxation linéaire est non-entier. $\longrightarrow$ 2. Après une coupe, l'optimum reste non-entier. $\longrightarrow$ 3. Après la 2ème coupe, l'optimum est bien entier !!



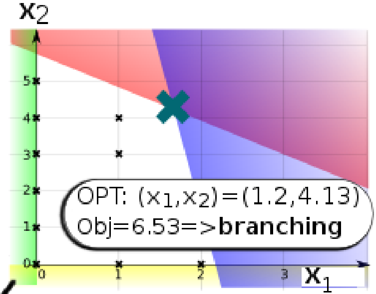
1. Difficulté des problèmes de nature discrète et explosion combinatoire

2. Méthodes de résolution en optimisation discrète
   - La relaxation linéaire et les bornes associées
   - Méthode à base de coupes
   - L'algorithme Branch-and-Bound (séparation et évaluation)

3. Exemples problèmes résolus avec correction (code)

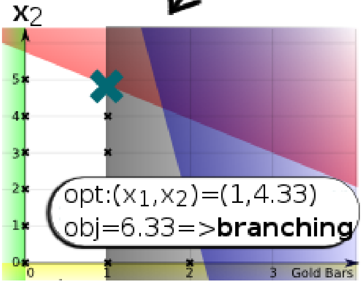
$$\max 2x_1 + x_2$$
$$8x_1 + 3x_2 \leq 22$$
$$6x_1 + 6x_2 \leq 32$$
$$\color{red}{x_1, x_2 \in \mathbb{Z}_+}$$

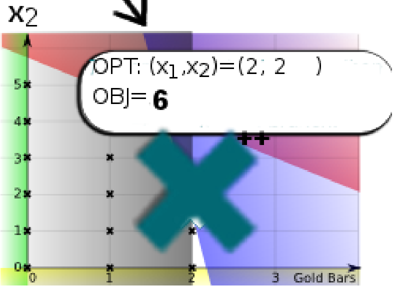
Exemple
Branch & Bound
(séparation&évaluation)

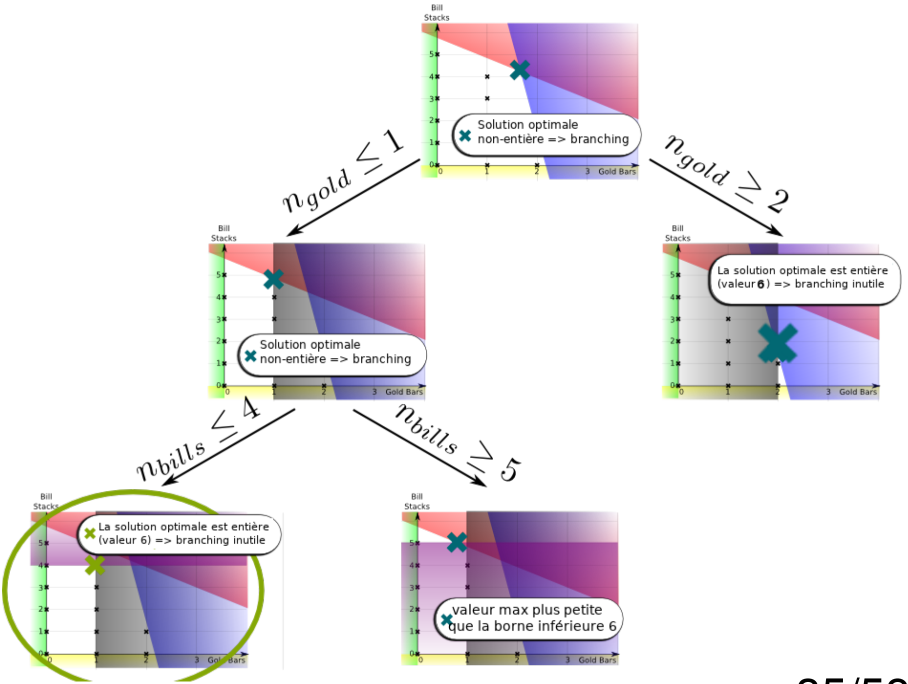


x1≤1

x1≥2






Solution optimale
non-entière => branching
$n_{gold} \leq 1$
$n_{gold} \geq 2$
Solution optimale
non-entière => branching
La solution optimale est entière
(valeur 6) => branching inutile
$n_{bills} \leq 4$
$n_{bills} \geq 5$
La solution optimale est entière
(valeur 6) => branching inutile
valeur max plus petite
que la borne inférieure 6
Bill Stacks
Gold Bars

Exercice *Branch and Bound*

$$\max 3x_1 + x_2$$
$$8x_1 + 4x_2 \leq 22$$
$$6x_1 + 6x_2 \leq 32$$
$$x_1, x_2 \in \mathbb{Z}_+$$

À partir de la solution optimale fractionnaire $(x_1, x_2) = (\frac{22}{8}, 0)$ de profit $3\frac{22}{8} = 8.25$, on considère deux cas :

- $x_1 = 2$ optimum fractionnaire : $(x_1, x_2) = (2, \frac{6}{4})$ de valeur $6 + 1.5 = 7.5$
  - $x_2 = 1$ Valeur : 7
  - $x_2 = 0$ Valeur : 6
- $x_1 \leq 1$ optimum fractionnaire : $(x_1, x_2) = (1, \frac{14}{4})$ de valeur $3 + 3.5 = 6.5 \Longrightarrow$ on coupe toute la branche!

$$\max 3x_1 + x_2$$
$$8x_1 + 4x_2 \leq 22$$
$$6x_1 + 6x_2 \leq 32$$
$$x_1, x_2 \in \mathbb{Z}_+$$

À partir de la solution optimale fractionnaire $(x_1, x_2) = (\frac{22}{8}, 0)$ de profit $3\frac{22}{8} = 8.25$, on considère deux cas :

$x_1 = 2$ optimum fractionnaire : $(x_1, x_2) = (2, \frac{6}{4})$ de valeur $6 + 1.5 = 7.5$

- $x_2 = 1$ Valeur : 7
- $x_2 = 0$ Valeur : 6

$x_1 \leq 1$ optimum fractionnaire : $(x_1, x_2) = (1, \frac{14}{4})$ de valeur $3 + 3.5 = 6.5 \implies$ on coupe toute la branche !

Exercice *Branch and Bound*

$$\max 3x_1 + x_2$$
$$8x_1 + 4x_2 \leq 22$$
$$6x_1 + 6x_2 \leq 32$$
$$x_1, x_2 \in \mathbb{Z}_+$$

À partir de la solution optimale fractionnaire $(x_1, x_2) = (\frac{22}{8}, 0)$ de profit $3\frac{22}{8} = 8.25$, on considère deux cas :

- $x_1 = 2$ optimum fractionnaire : $(x_1, x_2) = (2, \frac{6}{4})$ de valeur $6 + 1.5 = 7.5$
  - $x_2 = 1$ Valeur : 7
  - $x_2 = 0$ Valeur : 6
- $x_1 \leq 1$ optimum fractionnaire : $(x_1, x_2) = (1, \frac{14}{4})$ de valeur $3 + 3.5 = 6.5 \implies$ on coupe toute la branche !

Exercice *Branch and Bound*

$$\max 3x_1 + x_2$$
$$8x_1 + 4x_2 \leq 22$$
$$6x_1 + 6x_2 \leq 32$$
$$x_1, x_2 \in \mathbb{Z}_+$$

À partir de la solution optimale fractionnaire $(x_1, x_2) = (\frac{22}{8}, 0)$ de profit $3\frac{22}{8} = 8.25$, on considère deux cas :

$x_1 = 2$ optimum fractionnaire : $(x_1, x_2) = (2, \frac{6}{4})$ de valeur $6 + 1.5 = 7.5$

- $x_2 = 1$ Valeur : 7
- $x_2 = 0$ Valeur : 6

$x_1 \leq 1$ optimum fractionnaire : $(x_1, x_2) = (1, \frac{14}{4})$ de valeur $3 + 3.5 = 6.5 \implies$ on coupe toute la branche !

Héuristiques et Méta-heuristiques

Heuristique classique : remplir le sac avec le maximum d'or et compléter avec des billets

- Optimale si le prix de l'or est 300000$ mais $\boxed{\text{pas toujours}}$

Méta-heuristique : Une heuristique qui s'applique à une large familles de problèmes, ex, $\boxed{\text{recherche locale}}$

Méthodes exactes

1. Énumération complète → explosion combinatoire
2. Relaxation linéaire
   - On imagine qu'on pouvait fondre l'or → une borne supérieure de l'optimum entier (un profit exagéré)
3. Utiliser 2 mais ajouter des coupes pour raffiner la zone réalisable et rapprocher l'optimum fractionnaire de l'optimum entier
   - Une coupe est une contrainte artificielle : elle n'apparaît pas dans le problème de départ, mais elle peut mieux délimiter la zone réalisable fractionnaire

Héuristiques et Méta-heuristiques

Heuristique classique : remplir le sac avec le maximum d'or et compléter avec des billets

Méta-heuristique : Une heuristique qui s'applique à une large familles de problèmes, ex, recherche locale

Méthodes exactes

1. Énumération complète → explosion combinatoire
2. Relaxation linéaire
   - On imagine qu'on pouvait fondre l'or → une borne supérieure de l'optimum entier (un profit exagéré)
3. Utiliser 2 mais ajouter des coupes pour raffiner la zone realisable et rapprocher l'optimum fractionnaire de l'optimum entier
   - Une coupe est une contrainte artificielle : elle n'apparaît pas dans le problème de départ, mais elle peut mieux délimiter la zone réalisable fractionnaire
4. Combiner 1 et 2 → *Branch-and-Bound*

Héuristiques et Méta-heuristiques

Heuristique classique : remplir le sac avec le maximum d'or et compléter avec des billets
Méta-heuristique : Une heuristique qui s'applique à une large familles de problèmes, ex, $\boxed{\text{recherche locale}}$

Méthodes exactes

1. Énumération complète → explosion combinatoire
2. Relaxation linéaire
   - On imagine qu'on pouvait fondre l'or → une borne supérieure de l'optimum entier (un profit exagéré)
3. Utiliser 2 mais ajouter des coupes pour raffiner la zone realisable et rapprocher l'optimum fractionnaire de l'optimum entier
   - Une coupe est une contrainte artificielle : elle n'apparaît pas dans le problème de départ, mais elle peut mieux délimiter la zone réalisable fractionnaire
4. Combiner 1 et 2 → *Branch-and-Bound*

Héuristiques et Méta-heuristiques

Heuristique classique : remplir le sac avec le maximum d'or et compléter avec des billets
Méta-heuristique : Une heuristique qui s'applique à une large familles de problèmes, ex, $\boxed{\text{recherche locale}}$

Méthodes exactes

1. Énumération complète → explosion combinatoire
2. Relaxation linéaire
   - On imagine qu'on pouvait fondre l'or → une borne supérieure de l'optimum entier (un profit exagéré)
3. Utiliser 2 mais ajouter des coupes pour raffiner la zone realisable et rapprocher l'optimum fractionnaire de l'optimum entier
   - Une coupe est une contrainte artificielle : elle n'apparaît pas dans le problème de départ, mais elle peut mieux délimiter la zone réalisable fractionnaire
4. Combiner 1 et 2 → *Branch-and-Bound*

Héuristiques et Méta-heuristiques

Heuristique classique : remplir le sac avec le maximum d'or et compléter avec des billets

Méta-heuristique : Une heuristique qui s'applique à une large familles de problèmes, ex, $\boxed{\text{recherche locale}}$

Méthodes exactes

1. Énumération complète → explosion combinatoire
2. Relaxation linéaire
   - On imagine qu'on pouvait fondre l'or → une borne supérieure de l'optimum entier (un profit exagéré)
3. Utiliser 2 mais ajouter des coupes pour raffiner la zone realisable et rapprocher l'optimum fractionnaire de l'optimum entier
   - Une coupe est une contrainte artificielle : elle n'apparaît pas dans le problème de départ, mais elle peut mieux délimiter la zone réalisable fractionnaire
4. Combiner 1 et 2 → *Branch-and-Bound*

Héuristiques et Méta-heuristiques

Heuristique classique : remplir le sac avec le maximum d'or et compléter avec des billets

Méta-heuristique : Une heuristique qui s'applique à une large familles de problèmes, ex, recherche locale

Méthodes exactes

1. Énumération complète → explosion combinatoire
2. Relaxation linéaire
   - On imagine qu'on pouvait fondre l'or → une borne supérieure de l'optimum entier (un profit exagéré)
3. Utiliser 2 mais ajouter des coupes pour raffiner la zone realisable et rapprocher l'optimum fractionnaire de l'optimum entier
   - Une coupe est une contrainte artificielle : elle n'apparaît pas dans le problème de départ, mais elle peut mieux délimiter la zone réalisable fractionnaire
4. Combiner 1 et 2 → *Branch-and-Bound*

Plan

1. Difficulté des problèmes de nature discrète et explosion combinatoire
2. Méthodes de résolution en optimisation discrète
3. Exemples problèmes résolus avec correction (code)

1. Difficulté des problèmes de nature discrète et explosion combinatoire

2. Méthodes de résolution en optimisation discrète

3. Exemples problèmes résolus avec correction (code)

Problème du ... ?...

$$\max c^\top x = \sum_{i=1}^n c_i x_i$$
$$\sum_{i=1}^n a_i x_i \leq b$$
$$x_i \in \{0, 1\}$$

- n articles $\{1, 2, \dots n\}$
- n poids $a_1, a_2, \dots a_n$,
- n valeurs (prix) $c_1, c_2, \dots c_n \oplus$ une capacité b
- on veut maximiser le cout des articles choisis

Variables de décision

$x_i = 1$ si l'article i est sélectionnée, 0 sinon

Exemple : $a = [4, 3, 2, 1]$, $c = [44, 30, 20, 15]$ et $b = 6$

Comment comparer ce modèle avec $\boxed{\text{sa relaxation linéaire}}$?

Problème du sac à dos

$$\begin{aligned} \max c^\top x = \textstyle\sum_{i=1}^n c_i x_i \\ \textstyle\sum_{i=1}^n a_i x_i \leq b \\ x_i \in \{0, 1\} \end{aligned}$$

- n articles $\{1, 2, \dots n\}$
- n poids $a_1, a_2, \dots a_n$,
- n valeurs (prix) $c_1, c_2, \dots c_n \oplus$ une capacité b
- on veut maximiser le cout des articles choisis

Variables de décision

$x_i = 1$ si l'article i est sélectionnée, 0 sinon

Exemple : $a = [4, 3, 2, 1]$, $c = [44, 30, 20, 15]$ et $b = 6$

Comment comparer ce modèle avec $\boxed{\text{sa relaxation linéaire}}$?

Problème du sac à dos

$$\begin{aligned} \max c^\top x = \textstyle\sum_{i=1}^n c_i x_i \\ \textstyle\sum_{i=1}^n a_i x_i \leq b \\ x_i \in \{0,1\} \end{aligned}$$

- n articles $\{1, 2, \dots n\}$
- n poids $a_1, a_2, \dots a_n$,
- n valeurs (prix) $c_1, c_2, \dots c_n$ $\oplus$ une capacité b
- on veut maximiser le cout des articles choisis

Variables de décision

$x_i = 1$ si l'article i est sélectionnée, 0 sinon

Exemple : $a = [4, 3, 2, 1]$, $c = [44, 30, 20, 15]$ et $b = 6$

Comment comparer ce modèle avec $\boxed{\text{sa relaxation linéaire}}$?

```
var x1>=0;
var x2>=0;
maximize obj:3*x1+x2;
subject to fixer_x1: x1=2;
subject to c1: 8*x1+3*x2<=20;
subject to c2: 6*x1+6*x2<=32;
solve;
display obj,x1,x2;
end;
~
~
~
~
~
~
~
~
~
~
~
~
~
slide7.mod        9,4          All
```

```
param n, >0, integer;
param a{1..n}, integer;
param c{1..n}, integer;
param b;
var x{1..n}, >=0, binary;
maximize obj: sum{i in 1..n} c[i]*x[i];
s.t. poids_max: sum{i in 1..n} a[i]*x[i]<=b;
solve;
display obj;
for {i in 1..n}{ printf "%g ", x[i]; }
data;
param n:=4;
param b:=6;
param a:= 1  4
          2  3
          3  2
          4  1 ;
param c:= 1  44
          2  30
          3  20
          4  15 ;
end;
slide29-sac-a-dos.mod
```

Nous avons 95 écoliers

- 1 bus disponible de 50 places à 1700 euros
- 3 bus disponible de 30 places à 1000 euros par bus
- 2 bus disponible de 40 places à 1200 euros par bus

Deux problèmes à modéliser

- Minimiser le coût avec les données ci-dessus !
- En plus du coût fixe c indiqué plus haut, chaque bus a un coût de $0.01 \times c$ par écolier transporté.

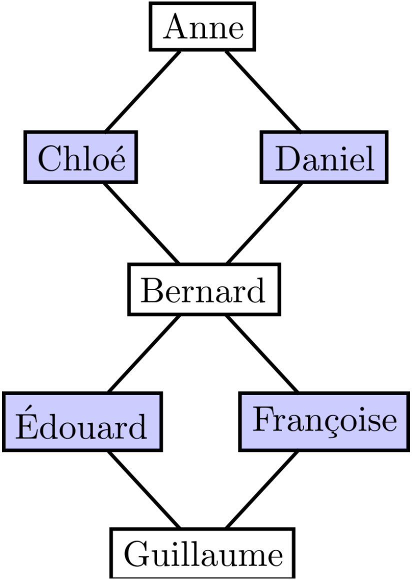
Nous avons 95 écoliers

- 1 bus disponible de 50 places à 1700 euros
- 3 bus disponible de 30 places à 1000 euros par bus
- 2 bus disponible de 40 places à 1200 euros par bus

Deux problèmes à modéliser

- Minimiser le coût avec les données ci-dessus !
- En plus du coût fixe c indiqué plus haut, chaque bus a un coût de $0.01 \times c$ par écolier transporté.

Le problème du stable maximal



Si un sommet=un individu et chaque arête correspond à deux individus qui se détestent $\Longrightarrow$ trouver une équipe maximale qu'avec des gens qui ne se détestent pas

Le problème du stable maximal

Soit un graphe $G(V, E)$:

S =stable de G : $\forall u, v \in S \implies u$ et v pas reliées ($\{u, v\} \notin E$)

Objectif : trouver le stable de taille maximale

$$\max \sum_{v \in V} x_v$$
$$x_u + x_v \leq 1 \quad \forall \{u, v\} \in E$$
$$x_v \in \{0, 1\} \quad \forall v \in V$$

D'autres applications :

- ouvrir le maximum de magasins en France sous la contrainte : ne pas ouvrir 2 magasins à une distance <100 km
- les huit dames

Soit un graphe $G(V, E)$:

S =stable de G : $\forall u, v \in S \implies u$ et v pas reliées ($\{u, v\} \notin E$)

Objectif : trouver le stable de taille maximale

$$\max \sum_{v \in V} x_v$$
$$x_u + x_v \leq 1 \quad \forall \{u, v\} \in E$$
$$x_v \in \{0, 1\} \quad \forall v \in V$$

D'autres applications :

- ouvrir le maximum de magasins en France sous la contrainte : ne pas ouvrir 2 magasins à une distance <100 km
- les huit dames

Le problème du stable maximal

Soit un graphe $G(V, E)$:

S =stable de G : $\forall u, v \in S \implies u$ et v pas reliées ($\{u, v\} \notin E$)

Objectif : trouver le stable de taille maximale

$$\begin{array}{ll} \max \sum_{v\in V} x_v & \\ x_u + x_v \leq 1 & \forall \{u, v\} \in E \\ x_v \in \{0, 1\} & \forall v \in V \end{array}$$

D'autres applications :

- ouvrir le maximum de magasins en France sous la contrainte : ne pas ouvrir 2 magasins à une distance <100 km
- les huit dames

Le problème du stable maximal

Soit un graphe $G(V, E)$:

S =stable de G : $\forall u, v \in S \implies u$ et v pas reliées ($\{u, v\} \notin E$)

Objectif : trouver le stable de taille maximale

$$\begin{array}{ll} \max \sum_{v \in V} x_v & \\ x_u + x_v \leq 1 & \forall \{u, v\} \in E \\ x_v \in \{0, 1\} & \forall v \in V \end{array}$$

D'autres applications :

- ouvrir le maximum de magasins en France sous la contrainte : ne pas ouvrir 2 magasins à une distance <100 km
- $\boxed{\text{les huit dames}}$

Problème du stable en GLPK

Il faut préciser plusieurs blocs :

```
param n;
param mat {1..n,1..n}, integer;

var x{1..n}, >=0, integer;

subject to nomConstr {i in 1..n,
          j in 1..n:mat[i,j]==1}: x[i] + x[j]<=1;

maximize obj: sum{i in 1..n} x[i];

solve;

printf "\n sum=%d\n", (sum{i in 1..n} x[i]);
data;    ->ces 3 lignes
.....    ->dans un autre
end      ->fichier
```

Le bloc à la fin

```
param n:=4;
param a:1 2 3 4:=
       1 0 1 1 1
       2 1 0 0 1
       3 1 0 0 1
       4 1 1 1 1;
end;
```

Ce bloc est optionnel et il peut apparaître dans un fichier séparé (voir l'option $-d$ de `glpsol`)

Comparons le temps de calcul en nombre réels est entiers

Le bloc à la fin

```
param n:=4;
param a:1 2 3 4:=
       1 0 1 1 1
       2 1 0 0 1
       3 1 0 0 1
       4 1 1 1 1;
end;
```

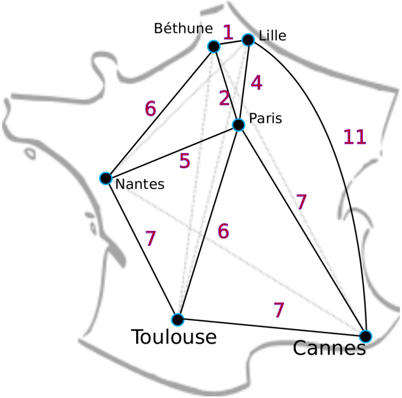
Ce bloc est optionnel et il peut apparaître dans un fichier séparé (voir l'option $-d$ de `glpsol`)

Comparons le temps de calcul en nombre réels est entiers

Le code final

```
param n, >0, integer;
param a{1..n,1..n}, integer;
var x{1..n}, >=0, binary;
s.t. c{i in 1..n, j in i+1..n: a[i,j]==1}:
    x[i] + x[j]<=1;
maximize obj: sum{i in 1..n} x[i];
solve;
printf "\nsum=%d\n", (sum{i in 1..n} x[i]);
data;
param n:=4;
param a:1 2 3 4:=
      1 0 1 1 1
      2 1 0 0 1
      3 1 0 0 1
      4 1 1 1 0;
end;
```

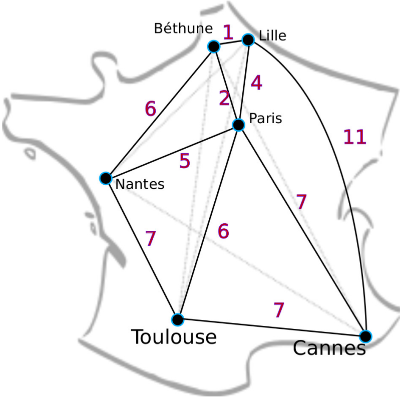
Il faut partir de Paris, visiter chaque ville une fois et revenir à Paris.



$d(\text{Paris,Lille}) = 4,\ d(\text{Paris,Toulouse}) = 7,\ \ldots$

Le voyageur de commerce

Il faut partir de Paris, visiter chaque ville une fois et revenir à Paris. Et minimiser le coût total !



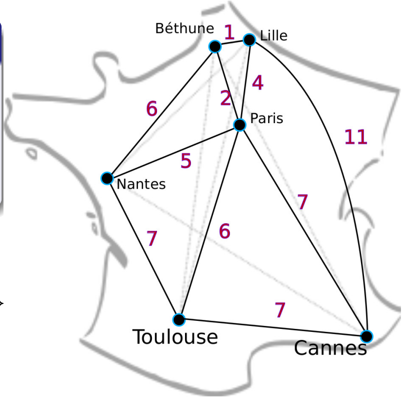
$d(\text{Paris,Lille}) = 4, \; d(\text{Paris,Toulouse}) = 7, \; \ldots$

Le voyageur de commerce

Il faut partir de Paris, visiter chaque ville une fois et revenir à Paris. Et minimiser le coût total !

Rappels graphe pondéré $G(V, E, d)$

V	ensemble de sommets
E	un ensemble d'arêtes $\{u, v\}$
$d(u, v)$	fonction de distance de u à v



$V = \{\text{Paris, Lille, Cannes, } \ldots\}$

$$E = \{ \quad \{\text{Paris,Lille}\}, \{\text{Paris,Toulouse}\} \{\text{Paris,Cannes}\}, \ldots\ldots \}$$

$d(\text{Paris,Lille}) = 4, \; d(\text{Paris,Toulouse}) = 7, \; \ldots$

$$\min_{x,u} \sum_{i \neq j}^{n} c_{ij} x_{ij}$$

$$\begin{array}{rrcl} \forall i \leq n & \sum_{j=0}^{n} x_{ij} & = & 1 \\ \forall j \leq n & \sum_{i=0}^{n} x_{ij} & = & 1 \\ \forall i \neq j \neq 0 & u_i - u_j + n x_{ij} & \leq & n - 1 \\ \forall i, j \leq n & x_{ij} & \in & \{0, 1\}. \end{array}$$

Le problème de la bi-partition équitable

On doit trouver deux sous-ensembles d'articles de même valeur totale $\Longrightarrow$ minimiser la valeur absolue d'une différence :

$$\begin{aligned} \min \left| \sum_{t=1}^{n} a_i x_i \right| \\ x_i \in \{-1, 1\} \end{aligned}$$

$$\xrightarrow{\text{Linéarisation}}$$

$$\begin{aligned} \min y \\ \sum_{i=1}^{n} a_i x_i \leq y \\ -\sum_{i=1}^{n} a_i x_i \leq y \\ x_i \in \{-1, 1\} \\ y \in [0, \infty] \end{aligned}$$

Le problème de la bi-partition équitable

On doit trouver deux sous-ensembles d'articles de même valeur totale $\Longrightarrow$ minimiser la valeur absolue d'une différence :

$$\begin{aligned} \min \left| \sum_{t=1}^{n} a_i x_i \right| \\ x_i \in \{-1, 1\} \end{aligned}$$

$$\xrightarrow{\text{Linéarisation}}$$

$$\begin{aligned} \min y \\ \sum_{i=1}^{n} a_i x_i \leq y \\ -\sum_{i=1}^{n} a_i x_i \leq y \\ x_i \in \{-1, 1\} \\ y \in [0, \infty] \end{aligned}$$

FIN DES SLIDES PRÉSENTÉS EN COURS

Mais vous pouvez parcourir les slides qui restent pour voir des exemples de problèmes (utiles pour le TP noté).

Problème de la couverture minimum par les sommets

- **Contexte** : on veut placer le moins possible de caméras de surveillance à des carrefours de façon à surveiller toutes les rues d'un quartier
- **Formalisation** : on représente le quartier par un graphe, dont les sommets sont les carrefours et les arêtes les rues entre ces carrefours (on cherche alors un ensemble de sommets de taille minimum qui « couvre » toutes les arêtes du graphe)
- **Modèle PLNE** ?
  - Une variable 0-1 par sommet du graphe

Modélisation problème de couverture

Données et paramètres

Graphe $G(V, E)$: sommets $V = \{1, 2, \ldots\}$, arrêtes E à surveiller

Variables de décision

$x_i = 1$ si on place une caméra au sommet i, 0 sinon

$\Longrightarrow$

$$\begin{aligned} \min \textstyle\sum_i x_i & \\ x_i + x_j \geq 1 & \quad \forall i, j \in E \\ x_i \in \{0, 1\} & \end{aligned}$$

Problème de localisation (de caméras)

Si on voulait placer k caméras et maximiser le nombre de sommets "surveillés" ? Réponse : on utilise une variable y_i qui indique si le sommet i est surveillé par une caméra ou pas.

$$\max \sum_i y_i$$

on doit lier y avec x qui encode le placement des k caméras.

$$\begin{aligned} \sum_{i=1}^{n} x_i &= \mathrm{k} \\ y_i \leq x_i + \sum_{\{i,j\}\in E} x_j & \quad \forall i \in \{1, 2, \ldots n\} \\ y_i \leq 1 & \quad \forall i \in \{1, 2, \ldots n\} \\ y_i, x_i \in \{0, 1\} & \end{aligned}$$

code GLPK pour la contrainte en x et y :

```
s.t. c{i in 1..n}:
    x[i]+sum {j in 1..n: a[i,j]==1}x[j] >=y[i];
```

Et si on voulait minimiser le même objectif ? $\Longrightarrow$ inverser la première inégalité mais doit-on multiplier y_i par n ? ?

Problème de localisation (de caméras)

Si on voulait placer k caméras et maximiser le nombre de sommets “surveillés” ? Réponse : on utilise une variable y_i qui indique si le sommet i est surveillé par une caméra ou pas.

$$\max \sum_i y_i$$

on doit lier y avec x qui encode le placement des k caméras.

$$\begin{aligned} \sum_{i=1}^{n} x_i &= k \\ y_i \leq x_i + \sum_{\{i,j\}\in E} x_j & \quad \forall i \in \{1,2,\ldots n\} \\ y_i \leq 1 & \quad \forall i \in \{1,2,\ldots n\} \\ y_i, x_i \in \{0,1\} & \end{aligned}$$

code GLPK pour la contrainte en x et y :

```
s.t. c{i in 1..n}:
    x[i]+sum {j in 1..n: a[i,j]==1}x[j] >=y[i];
```

Et si on voulait minimiser le même objectif ? $\Longrightarrow$ inverser la première inégalité mais doit-on multiplier y_i par n ? ?

Problème de localisation (de caméras)

Si on voulait placer k caméras et maximiser le nombre de sommets “surveillés” ? Réponse : on utilise une variable y_i qui indique si le sommet i est surveillé par une caméra ou pas.

$$\max \sum_i y_i$$

on doit lier y avec x qui encode le placement des k caméras.

$$\begin{aligned} \sum_{i=1}^{n} x_i &= \mathrm{k} & \\ y_i \leq x_i + \sum_{\{i,j\} \in E} x_j & & \forall i \in \{1, 2, \ldots n\} \\ y_i &\leq 1 & \forall i \in \{1, 2, \ldots n\} \\ y_i, x_i &\in \{0, 1\} & \end{aligned}$$

code GLPK pour la contrainte en x et y :

```
s.t. c{i in 1..n}:
    x[i]+sum {j in 1..n: a[i,j]==1}x[j] >=y[i];
```

Et si on voulait minimiser le même objectif ? $\Longrightarrow$ inverser la première inégalité mais doit-on multiplier y_i par n ? ?

Problème de localisation (de caméras)

Si on voulait placer k caméras et maximiser le nombre de sommets “surveillés” ? Réponse : on utilise une variable y_i qui indique si le sommet i est surveillé par une caméra ou pas.

$$\max \sum_i y_i$$

on doit lier y avec x qui encode le placement des k caméras.

$$\begin{aligned} \sum_{i=1}^{n} x_i &= \text{k} & \\ y_i \leq x_i + \sum_{\{i,j\}\in E} x_j & & \forall i \in \{1, 2, \ldots n\} \\ y_i &\leq 1 & \forall i \in \{1, 2, \ldots n\} \\ y_i, x_i &\in \{0, 1\} & \end{aligned}$$

code GLPK pour la contrainte en x et y :

```
s.t. c{i in 1..n}:
    x[i]+sum {j in 1..n: a[i,j]==1}x[j] >=y[i];
```

Et si on voulait minimiser le même objectif ? $\Longrightarrow$ inverser la première inégalité mais doit-on multiplier y_i par n ? ?

Problème de localisation (de caméras)

Si on voulait placer k caméras et maximiser le nombre de sommets “surveillés” ? Réponse : on utilise une variable y_i qui indique si le sommet i est surveillé par une caméra ou pas.

$$\max \sum_i y_i$$

on doit lier y avec x qui encode le placement des k caméras.

$$\begin{aligned}
\sum_{i=1}^{n} x_i &= \text{k} \\
y_i \leq x_i + \sum_{\{i,j\}\in E} x_j & \quad \forall i \in \{1, 2, \ldots n\} \\
y_i \leq 1 & \quad \forall i \in \{1, 2, \ldots n\} \\
y_i, x_i \in \{0, 1\} &
\end{aligned}$$

code GLPK pour la contrainte en x et y :

```
s.t. c{i in 1..n}:
    x[i]+sum {j in 1..n: a[i,j]==1}x[j] >=y[i];
```

Et si on voulait minimiser le même objectif ? $\Longrightarrow$ inverser la première inégalité mais doit-on multiplier y_i par n ? ?

Problème de localisation (de caméras)

Si on voulait placer k caméras et maximiser le nombre de sommets “surveillés” ? Réponse : on utilise une variable y_i qui indique si le sommet i est surveillé par une caméra ou pas.

$$\max \sum_i y_i$$

on doit lier y avec x qui encode le placement des k caméras.

$$\begin{aligned} \sum_{i=1}^n x_i &= \text{k} \\ y_i \leq x_i + \sum_{\{i,j\}\in E} x_j & \quad \forall i \in \{1, 2, \ldots n\} \\ y_i \leq 1 & \quad \forall i \in \{1, 2, \ldots n\} \\ y_i, x_i \in \{0, 1\} & \end{aligned}$$

code GLPK pour la contrainte en x et y :

```
s.t. c{i in 1..n}:
    x[i]+sum {j in 1..n: a[i,j]==1}x[j] >=y[i];
```

Et si on voulait minimiser le même objectif ? $\Longrightarrow$ inverser la première inégalité mais doit-on multiplier y_i par n ? ?

Problème d'affectation linéaire

- **Contexte** : dans une entreprise, n tâches doivent être affectées à n agents de façon à minimiser le temps total (= somme des temps) mis pour effectuer toutes les tâches, sachant que le temps mis par un agent pour effectuer une tâche dépend de la tâche ET de l'agent
- **Formalisation** : chaque paire (tâche, agent) est munie d'une valeur (temps en minutes), et 1 tâche = 1 agent
- **Modèle PLNE** ?
  - Une variable 0-1 par paire (tâche, agent)

Agents	Tâche 1	Tâche 2	Tâche 3
1	10 min	15 min	40 min
2	5 min	20 min	1 heure
3	10 min	3 heures	3 heures

Problème d'affectation linéaire

Données et paramètres

- Taches $T = \{1, 2, \ldots n\}$
- Machines $A = \{1, 2, \ldots n\}$
- c_{ta} = cout d'affectation de la tache t à la machine a

Variables de décision

$x_{ta} = 1$ si la tache t affectée à la machine a, 0 sinon

$\Longrightarrow$

$$\begin{array}{l} \min \sum_{ta} c_{ta} x_{ta} \\ \sum_{t=1}^{n} x_{ta} = 1 \quad \forall a \in \{1, 2, \ldots n\} \\ \sum_{a=1}^{n} x_{ta} = 1 \quad \forall t \in \{1, 2, \ldots n\} \\ x_{at} \in \{0, 1\} \end{array}$$

Problème d'affectation linéaire : GLPK

On modifie le .mod précédent. La contrainte devient :

```
s.t. contr_pour_chaque_ligne{ligne in 1..n}:
    sum{col in 1..n} x[ligne,col]=1;
s.t. contr_pour_chaque_col{col in 1..n}:
    sum{ligne in 1..n} x[ligne,col]=1;
```

Pour afficher une matrice, on utilise :

```
for {i in 1..n}
{
    for {j in 1..n}
    {
        printf "%d ", (x[i,j]);
    }
    printf "\n";
}
```

Le sous-graphe induit de densité maximale

Objectif

- Entrée : un graphe (V, E)
- Objectif : Sélectionner k sommets $S = \{v_1, v_2, \ldots v_k\} \subset V$ qui comportent le maximum d'arêtes $\{v_i, v_j\} \in E$.

$$
\begin{array}{ll}
\max \sum_{\{i,j\}\in E} x_i \cdot x_j & \\
\sum_{i=1}^{n} x_i \leq k & \\
x_i \in \{0,1\}^n & \forall i \in \{1, 2 \ldots n\}
\end{array}
$$

Défi : le programme n'est pas linéaire et les solveurs (comme `glpk`) ne peuvent pas tous gérer les produits $x_i \cdot x_j$
$\Longrightarrow$ On doit replacer $x_i \cdot x_j$ par des combinaisons linéaires $\Longrightarrow$ linéarisation.

Le sous-graphe induit de densité maximale

Objectif

- Entrée : un graphe (V, E)
- Objectif : Sélectionner k sommets $S = \{v_1, v_2, \ldots v_k\} \subset V$ qui comportent le maximum d'arêtes $\{v_i, v_j\} \in E$.

$$\begin{array}{ll} \max \sum_{\{i,j\}\in E} x_i \cdot x_j & \\ \sum_{i=1}^{n} x_i \leq k & \\ x_i \in \{0,1\}^n & \forall i \in \{1, 2 \ldots n\} \end{array}$$

Défi : le programme n'est pas linéaire et les solveurs (comme `glpk`) ne peuvent pas tous gérer les produits $x_i \cdot x_j$

$\implies$ On doit replacer $x_i \cdot x_j$ par des combinaisons linéaires $\implies$ linéarisation.

Linéarisation de $x_i \cdot x_j$ vers y_{ij}

On ajoute des variables additionnelles $y_{ij} \in \{0, 1\}$ et on impose :

$$y_{ij} \leq x_i$$
$$y_{ij} \leq x_j$$

et ....

$$y_{ij} \geq x_j + x_i - 1$$

$\implies$ $y_{ij} = x_i \cdot x_j$ parce que toutes les variables sont binaires !

$$\implies$$

On résout un PLNE (programme linéaire en nombre entiers) avec objectif $\sum y_{ij}$ et on obtient la solution entière du programme initial (quadratique en nombre entiers)

Linéarisation de $x_i \cdot x_j$ vers y_{ij}

On ajoute des variables additionnelles $y_{ij} \in \{0, 1\}$ et on impose :

$$y_{ij} \leq x_i$$
$$y_{ij} \leq x_j$$

et

$$y_{ij} \geq x_j + x_i - 1$$

$\implies y_{ij} = x_i \cdot x_j$ parce que toutes les variables sont binaires !

$$\implies$$

On résout un PLNE (programme linéaire en nombre entiers) avec objectif $\sum y_{ij}$ et on obtient la solution entière du programme initial (quadratique en nombre entiers)

La contrainte $x \neq 7$ via un PL

$x \geq 8 - y \cdot C$
$x \leq 6 + (1 - y) \cdot C$
$y \in \{0, 1\}$

Si $y = 1$, alors $x \leq 6$ (donc $x = 7$ est impossible)
Si $y = 0$, alors $x \geq 8$ (donc $x = 7$ est impossible)

$\boxed{\text{Comment linéariser } x_1 \neq x_2\,?}$