



# Production Industrielle

Guilherme ESPINDOLA-WINCK  
Enseignant-Chercheur (MCF) à Centrale Lille et  
au CRISTAL

## **Partie II : Ordonnancement et planification de la production**

3

# Agenda

- Ordonnancement et Planification de la Production – LP et MIP

4



# Ordonnancement et planification

5

# Ordonnancement et Planification de la Production

- Programmation linéaire

- Exemple : Maximiser le profit de l'usine en tenant compte de la disponibilité de la matière première m et de l'utilisation de chaque matière première par produit p
- Deux produits  $p=1,2$  ; Trois matières premières  $m=1,2,3$
- $x(p)$  : le nombre de produits p à fabriquer
- $L(p)$  : profit du produit p
- $\max \sum_p L(p)x(p)$  : fonction objective
- $C(m, p)$  : utilisation de m par p
- $D(m)$  : disponibilité de m
- $\sum_p C(m, p)x(p) \leq D(m)$  : contrainte pour chaque m

	$D(m)$
$m=1$	6000
$m=2$	6000
$m=3$	2500

	$C(m, p)$	$p=1$	$p=2$
$m=1$	1	1	
$m=2$	2	2	
$m=3$	0	1	

# Ordonnancement et Planification de la Production

- Programmation linéaire

$$\min_x f^T x \text{ such that} \begin{cases} A \cdot x \leq b, \\ Aeq \cdot x = beq, \\ lb \leq x \leq ub. \end{cases}$$

- Donner les paramètres du problème :

7



# Ordonnancement et Planification de la Production

- Programmation linéaire

$$\min_x f^T x \text{ such that} \begin{cases} A \cdot x \leq b, \\ Aeq \cdot x = beq, \\ lb \leq x \leq ub. \end{cases}$$

- Donner les paramètres du problème :

solution optimale :

3000

0

profit total :

66000

```
% min -f'*x <-> max f'*x
f = [-22 -20]';
% l'utilisation de m
A = [1 1;
      2 2;
      0 1];
% A*[x1 x2]' \leq b ; b disponibilité de
b = [6000; 6000; 2500];
% limites de x
lb = [0 0];
ub = [inf inf];
%% Appel linprog
% x = linprog(f,A,b,Aeq,beq,lb,ub)
% Aeq et beq [] "vide"
x = linprog(f,A,b,[],[],lb,ub);
disp("solution optimale :")
disp(x)
%% profit
disp("profit total :")
J=-f'*x;
disp(J)
```

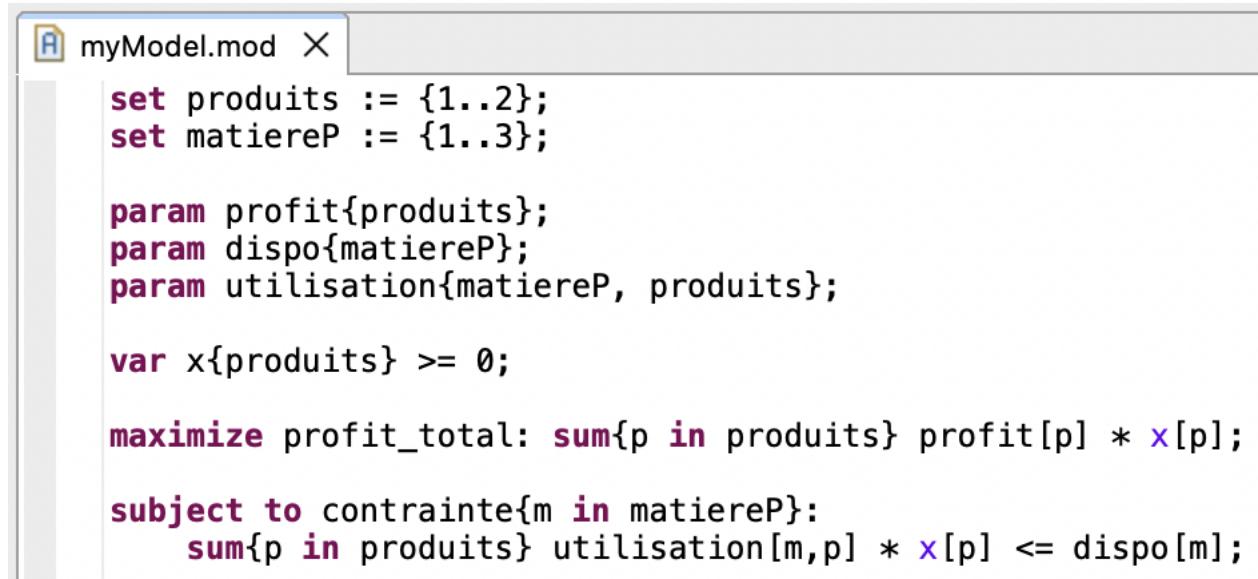
# Ordonnancement et Planification de la Production

- Installation du logiciel AMPL (Windows, Linux, macOS) pour résoudre les problèmes d'Ordonnancement et Planification :
  - <https://AMPL.com/start-free-now/> (AMPL Community Edition)



# Ordonnancement et Planification de la Production

- AMPL IDE :



```
myModel.mod X

set produits := {1..2};
set matiereP := {1..3};

param profit{produits};
param dispo{matiereP};
param utilisation{matiereP, produits};

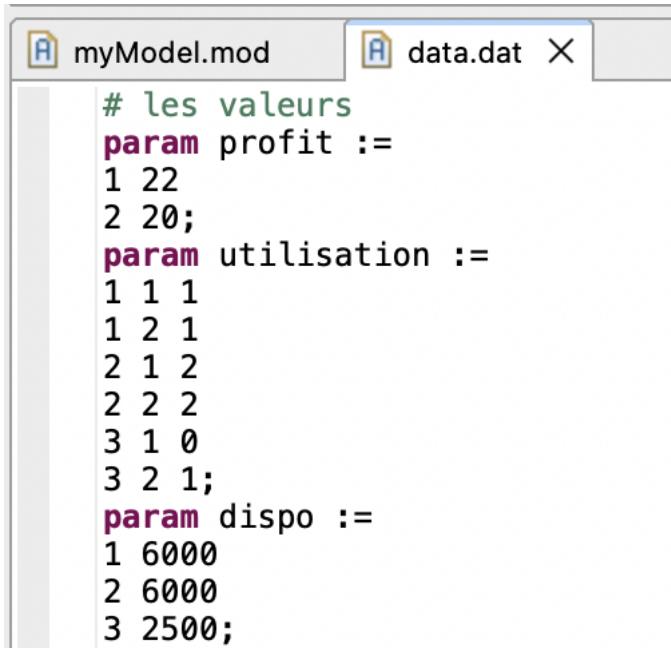
var x{produits} >= 0;

maximize profit_total: sum{p in produits} profit[p] * x[p];

subject to contrainte{m in matiereP}:
    sum{p in produits} utilisation[m,p] * x[p] <= dispo[m];
```

# Ordonnancement et Planification de la Production

- AMPL IDE :

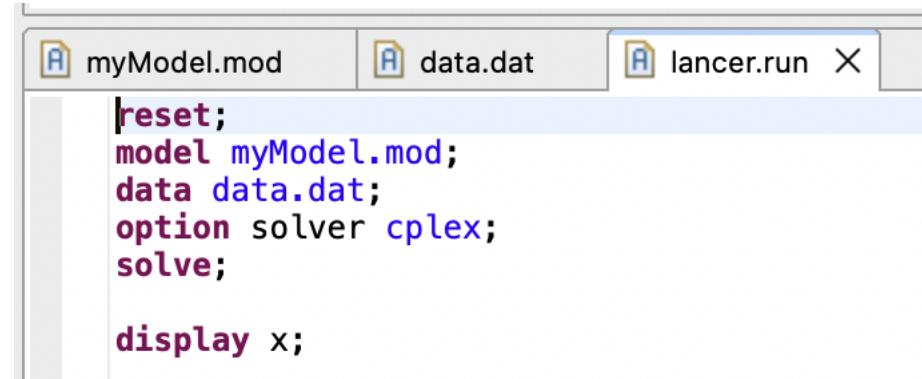


The screenshot shows an AMPL IDE interface with two tabs open: "myModel.mod" and "data.dat". The "myModel.mod" tab contains the following AMPL code:

```
# les valeurs
param profit :=  
1 22  
2 20;  
param utilisation :=  
1 1 1  
1 2 1  
2 1 2  
2 2 2  
3 1 0  
3 2 1;  
param dispo :=  
1 6000  
2 6000  
3 2500;
```

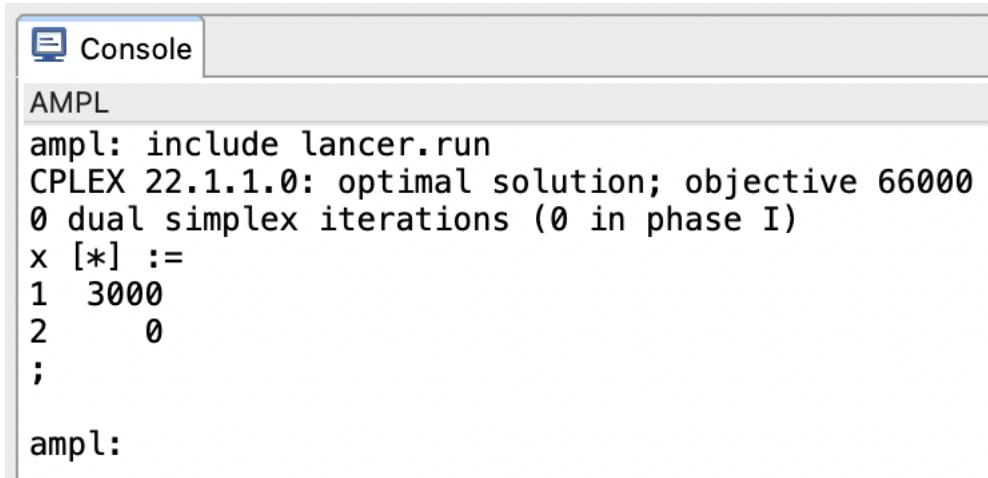
# Ordonnancement et Planification de la Production

- AMPL IDE :



```
reset;
model myModel.mod;
data data.dat;
option solver cplex;
solve;

display x;
```



Console

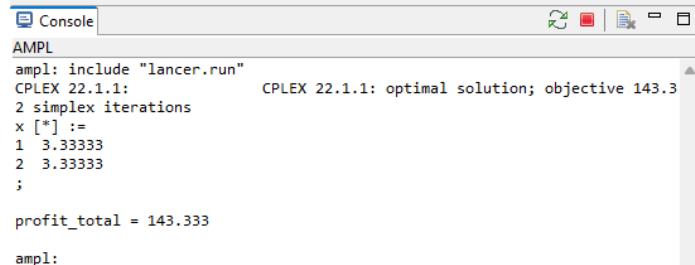
AMPL

```
ampl: include lancer.run
CPLEX 22.1.1.0: optimal solution; objective 66000
0 dual simplex iterations (0 in phase I)
x [*] :=
1 3000
2 0
;

ampl:
```

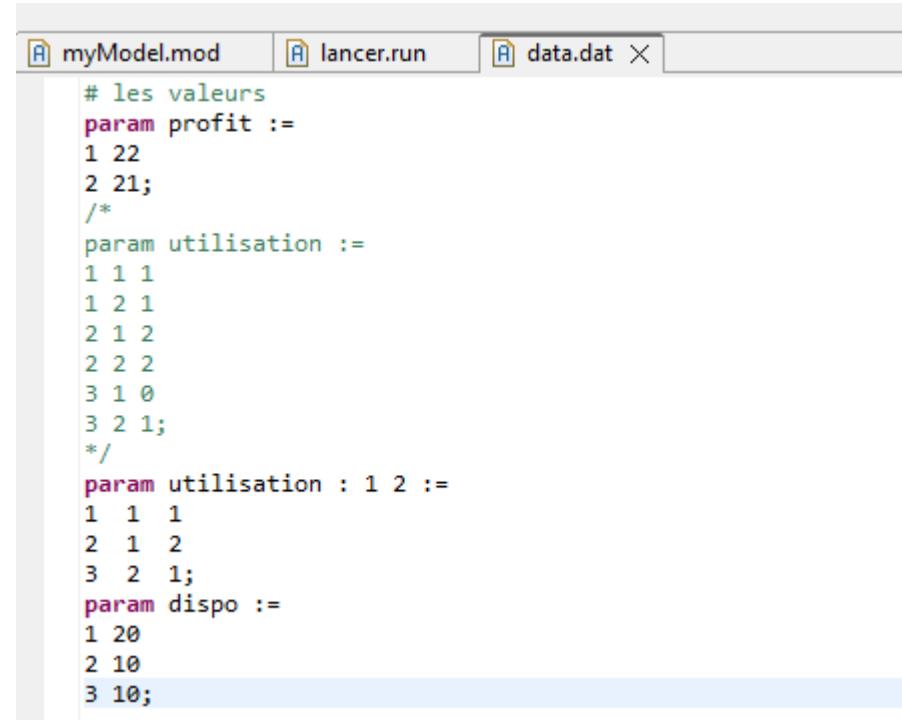
# Ordonnancement et Planification de la Production

- AMPL IDE :



The screenshot shows the AMPL IDE interface. On the left, there's a 'Console' window displaying the following output:

```
AMPL
ampl: include "lancer.run"
CPLEX 22.1.1: optimal solution; objective 143.3
2 simplex iterations
x [*] :=
1 3.33333
2 3.33333
;
profit_total = 143.333
ampl:
```



The screenshot shows the AMPL IDE interface with three tabs open: 'myModel.mod', 'lancer.run', and 'data.dat'. The 'myModel.mod' tab is active and displays the following AMPL code:

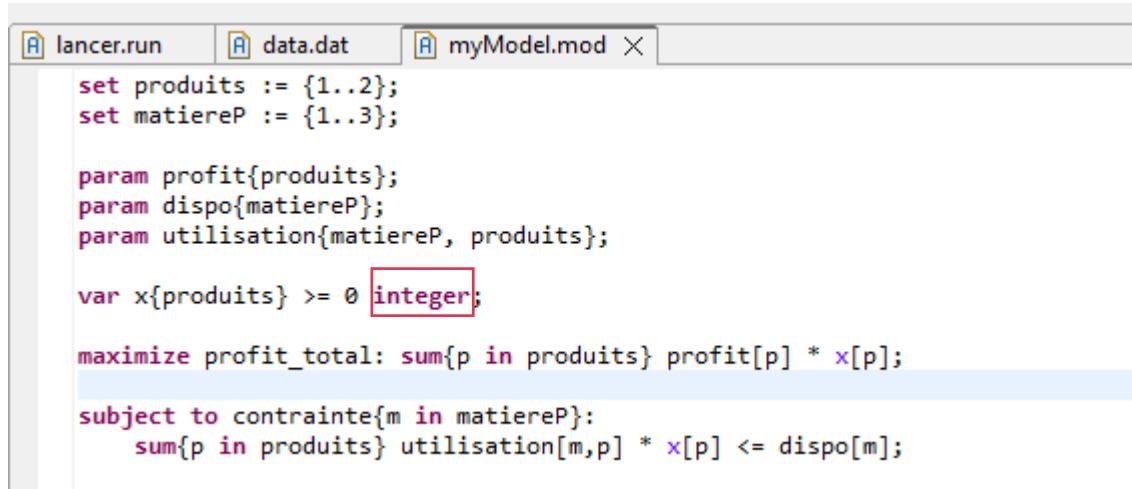
```
# les valeurs
param profit :=  
1 22  
2 21;  
/*  
param utilisation :=  
1 1 1  
1 2 1  
2 1 2  
2 2 2  
3 1 0  
3 2 1;*/  
param utilisation : 1 2 :=  
1 1 1  
2 1 2  
3 2 1;  
param dispo :=  
1 20  
2 10  
3 10;
```

13



# Ordonnancement et Planification de la Production

- AMPL IDE :



```
set produits := {1..2};
set matiereP := {1..3};

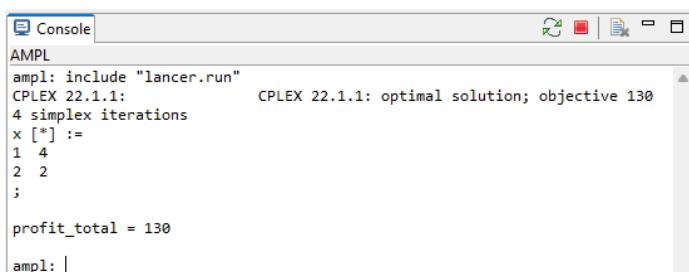
param profit{produits};
param dispo{matiereP};
param utilisation{matiereP, produits};

var x{produits} >= 0 integer;

maximize profit_total: sum{p in produits} profit[p] * x[p];

subject to contrainte{m in matiereP}:
    sum{p in produits} utilisation[m,p] * x[p] <= dispo[m];
```

14



```
Console
AMPL
ampl: include "lancer.run"
CPLEX 22.1.1:          CPLEX 22.1.1: optimal solution; objective 130
4 simplex iterations
x [*] :=
1 4
2 2
;
profit_total = 130
ampl: |
```

14

# Ordonnancement et Planification de la Production

- Programmation linéaire mixte
  - Exemple : minimisation du « load » d'une machine

- $x(i,j) = \begin{cases} 1, & \text{si la tâche } i \text{ est exécutée à la position } j \\ 0, & \text{sinon} \end{cases}$
- $i = 1, \dots, n = 7$  and  $j = 1, \dots, m$

Tâche	1	2	3	4	5	6	7
$p_i$	3	6	6	5	4	8	9

- $\min LOAD = \sum_i \sum_j p_i x_{ij}$
- $\sum_j x_{ij} \leq 1, \forall i$  : chaque tâche (ligne\_i) a une position (si la tâche est affectée à une position  $\leq 1$ )
- $\sum_i x_{ij} = 1, \forall j$  : chaque position (colonne\_j) a une seule tâche

15



# Ordonnancement et Planification de la Production

- Programmation entière mixte

$$\min_x f^T x \text{ subject to } \begin{cases} x(\text{intcon}) \text{ are integers} \\ A \cdot x \leq b \\ Aeq \cdot x = beq \\ lb \leq x \leq ub. \end{cases}$$

- Donner les paramètres du problème pour m=4 :

16



# Ordonnancement et Planification de la Production

- Programmation entière mixte

$$\min_x f^T x \text{ subject to} \begin{cases} x(\text{intcon}) \text{ are integers} \\ A \cdot x \leq b \\ Aeq \cdot x = beq \\ lb \leq x \leq ub. \end{cases}$$

- Donner les paramètres du problème m=4 :

ordonnancement :

2        5        4        1

fval

18

```
m=4; % nb de positions  
% pour montrer le résultat  
ord = 1:7;  
% vecteur de poids  
p = [3; 6; 6; 5; 4; 8; 9];  
% fonction objectif : \sum_i \sum_j p_i*x_ij  
f = kron(p, ones(m, 1));  
f=f(:);  
% contraintes \sum_j x_ij <= 1 pour tout i  
A=[kron(eye(7), ones(1,m))];  
b=ones(7,1);  
% contraintes \sum_i x_ij = 1 pour tout j  
Aeq=[kron(ones(1,7), eye(m))];  
beq=ones(m,1);  
% x_ij \in {0,1}  
lb=zeros(7*m,1);  
ub=ones(7*m,1);  
% variables entières  
intcon=1:(7*m);  
% solveur  
[x, fval] = intlinprog(f, intcon, A, b, Aeq, beq, lb, ub);  
% ordonnancement  
aux=reshape(x, [m,7])';  
disp("ordonnancement :")  
disp(ord*aux)  
disp("fval")  
disp(fval)
```

Tâche	1	2	3	4	5	6	7
$p_i$	3	6	6	5	4	8	9

# Ordonnancement et Planification de la Production

- Programmation entière mixte

$$\min_x f^T x \text{ subject to} \begin{cases} x(\text{intcon}) \text{ are integers} \\ A \cdot x \leq b \\ Aeq \cdot x = beq \\ lb \leq x \leq ub. \end{cases}$$

- Donner les paramètres du problème m=7 :

ordonnancement :

2      7      6      5      1      3      4

fval

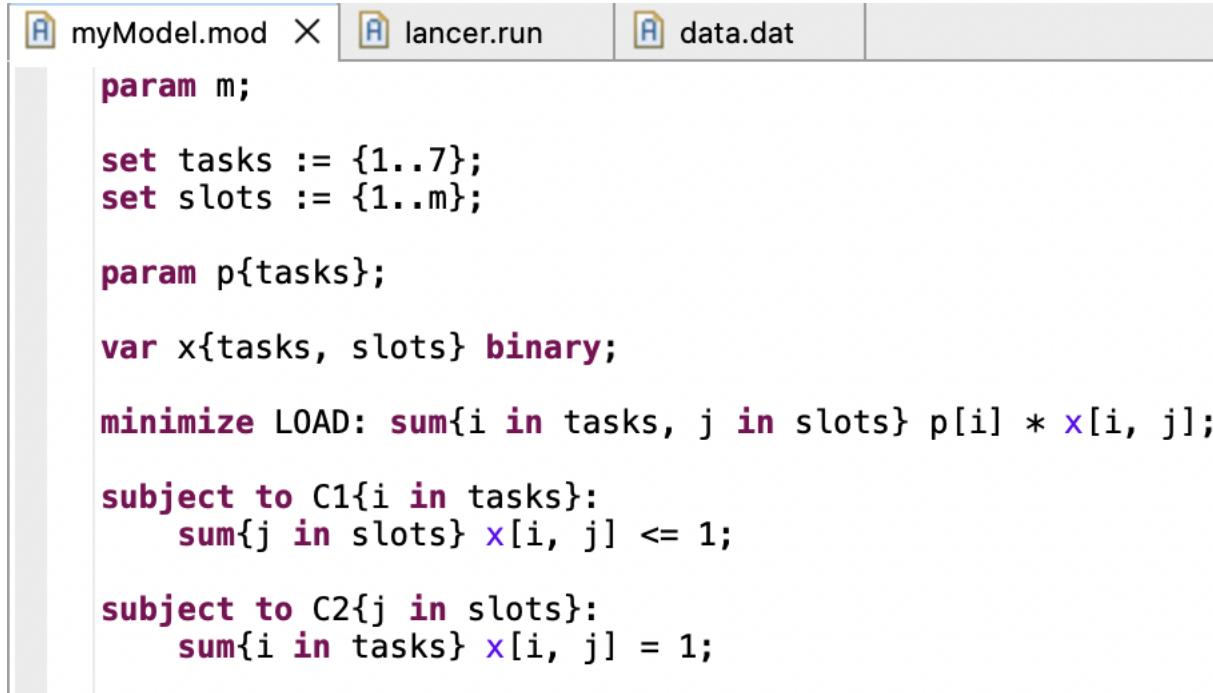
41

Peu importe l'ordre

```
m=4; % nb de positions  
% pour montrer le resultat  
ord = 1:7;  
% vecteur de poids  
p = [3; 6; 6; 5; 4; 8; 9];  
% fonction objectif : \sum_i \sum_j p_i*x_ij  
f = kron(p, ones(m, 1));  
f=f(:);  
% contraintes \sum_j x_ij <= 1 pour tout i  
A=[kron(eye(7), ones(1,m))];  
b=ones(7,1);  
% contraintes \sum_i x_ij = 1 pour tout j  
Aeq=[kron(ones(1,7), eye(m))];  
beq=ones(m,1);  
% x_ij \in {0,1}  
lb=zeros(7*m,1);  
ub=ones(7*m,1);  
% variables entieres  
intcon=1:(7*m);  
% solveur  
[x, fval] = intlinprog(f, intcon, A, b, Aeq, beq, lb, ub);  
% ordonnancement  
aux=reshape(x, [m,7])';  
disp("ordonnancement :")  
disp(ord*aux)  
disp("fval")  
disp(fval)  
disp(f'*x)
```

# Ordonnancement et Planification de la Production

- AMPL IDE :



The screenshot shows an AMPL IDE interface with three tabs at the top: "myModel.mod" (selected), "lancer.run", and "data.dat". The main area displays the following AMPL model code:

```
param m;

set tasks := {1..7};
set slots := {1..m};

param p{tasks};

var x{tasks, slots} binary;

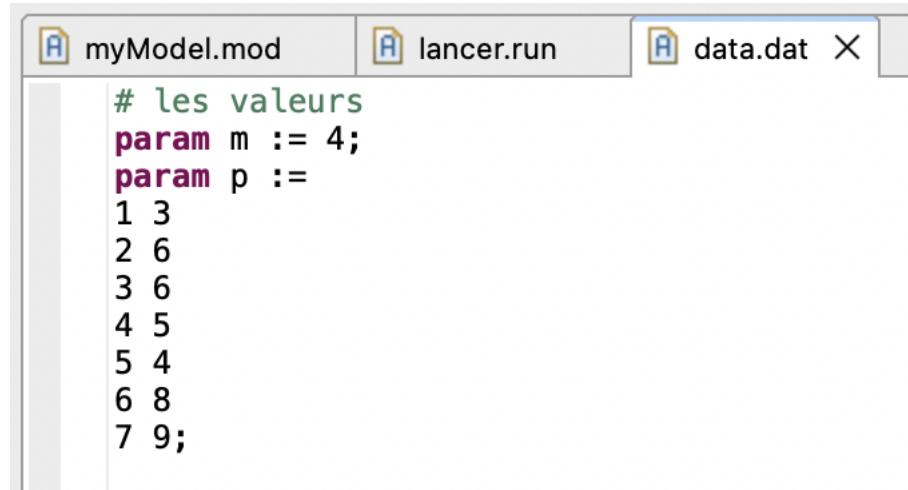
minimize LOAD: sum{i in tasks, j in slots} p[i] * x[i, j];

subject to C1{i in tasks}:
    sum{j in slots} x[i, j] <= 1;

subject to C2{j in slots}:
    sum{i in tasks} x[i, j] = 1;
```

# Ordonnancement et Planification de la Production

- AMPL IDE :



```
# les valeurs
param m := 4;
param p :=
1 3
2 6
3 6
4 5
5 4
6 8
7 9;
```

# Ordonnancement et Planification de la Production

- AMPL IDE :



```
reset;
model myModel.mod;
data data.dat;
option solver cplex;
solve;

display x;

printf "Fonction objectif (load) : %f\n", LOAD;
printf "Ordonnancement :\n";
for {i in tasks} {
    for {j in slots : x[i, j] = 1} {
        printf "Tache %d dans la position : ", i;
        printf "%d ", j;
        printf "\n";
    }
}
printf "\n";
```



# Ordonnancement et Planification de la Production

- AMPL IDE :

Console

AMPL

```
ampl: include lancer.run
CPLEX 22.1.1.0: optimal integer solution; objective 18
6 MIP simplex iterations
0 branch-and-bound nodes
x [*,*]
:   1   2   3   4      :=
1   1   0   0   0
2   0   0   0   0
3   0   0   1   0
4   0   0   0   1
5   0   1   0   0
6   0   0   0   0
7   0   0   0   0
;
```

Fonction objectif (load) : 18.000000

Ordonnancement :

```
Tache 1 dans la position : 1
Tache 3 dans la position : 3
Tache 4 dans la position : 4
Tache 5 dans la position : 2
```

# Ordonnancement et Planification de la Production

- AMPL IDE :

Console

AMPL

```
ampl: include lancer.run
CPLEX 22.1.1.0: optimal integer solution; objective 18
6 MIP simplex iterations
0 branch-and-bound nodes
x [*,*]
:   1   2   3   4      :=
1   1   0   0   0
2   0   0   0   0
3   0   0   1   0
4   0   0   0   1
5   0   1   0   0
6   0   0   0   0
7   0   0   0   0
;
```

Fonction objectif (load) : 18.000000

Ordonnancement :

Tache 1 dans la position : 1  
Tache 3 dans la position : 3  
Tache 4 dans la position : 4  
Tache 5 dans la position : 2

# Ordonnancement et Planification de la Production

- AMPL IDE :

```
Console  
AMPL  
ampl: include lancer.run  
CPLEX 22.1.1.0: optimal integer solution; objective 41  
14 MIP simplex iterations  
0 branch-and-bound nodes  
x [*,*]  
: 1 2 3 4 5 6 7 :=  
1 1 0 0 0 0 0 0  
2 0 1 0 0 0 0 0  
3 0 0 1 0 0 0 0  
4 0 0 0 1 0 0 0  
5 0 0 0 0 1 0 0  
6 0 0 0 0 0 1 0  
7 0 0 0 0 0 0 1  
;  
  
Fonction objectif (load) : 41.000000  
Ordonnancement :  
Tache 1 dans la position : 1  
Tache 2 dans la position : 2  
Tache 3 dans la position : 3  
Tache 4 dans la position : 4  
Tache 5 dans la position : 5  
Tache 6 dans la position : 6  
Tache 7 dans la position : 7
```

# Ordonnancement et Planification de la Production

- Exercice :
- Utiliser AMLP ou PuLP python, etc... Matlab : Yalmip, etc...
- 6 tâches
- 3 machines : chaque machine possède 3 positions
- Machine 1 et 2 ont la même « vitesse »/(poids inversé) :  $v_1 = v_2 = 1$  ; pour Machine 3 :  $v_3 = 2$
- Minimiser le « makespan » le plus élevé parmi toutes les machines : le « makespan » d'une machine dans un contexte de planification fait référence au temps total nécessaire à la machine pour terminer toutes les tâches.
- Chaque machine a son propre « makespan » :  $t_m = \frac{1}{v_m} \sum_j \sum_i p_j x_{jim}$ , avec  $x_{jim} = 1$  si tâche j est exécutée dans la position i de la machine m ;  $p_j$  le temps de traitement de la tâche j

Tâche	1	2	3	4	5	6
$p_i$	6	6	7	7	9	10

- $\min C$  avec  $C = \max_m t_m \rightarrow C \geq t_m, \forall m$

# Ordonnancement et Planification de la Production

- Exercice :

```
myModel.mod X lancer.run data.dat

set tasks := {1..6};
set slots := {1..3};
set machines := {1..3};

param p{tasks};
param v{machines};

var x{tasks, slots, machines} binary;

var opt;

minimize OPT: opt;

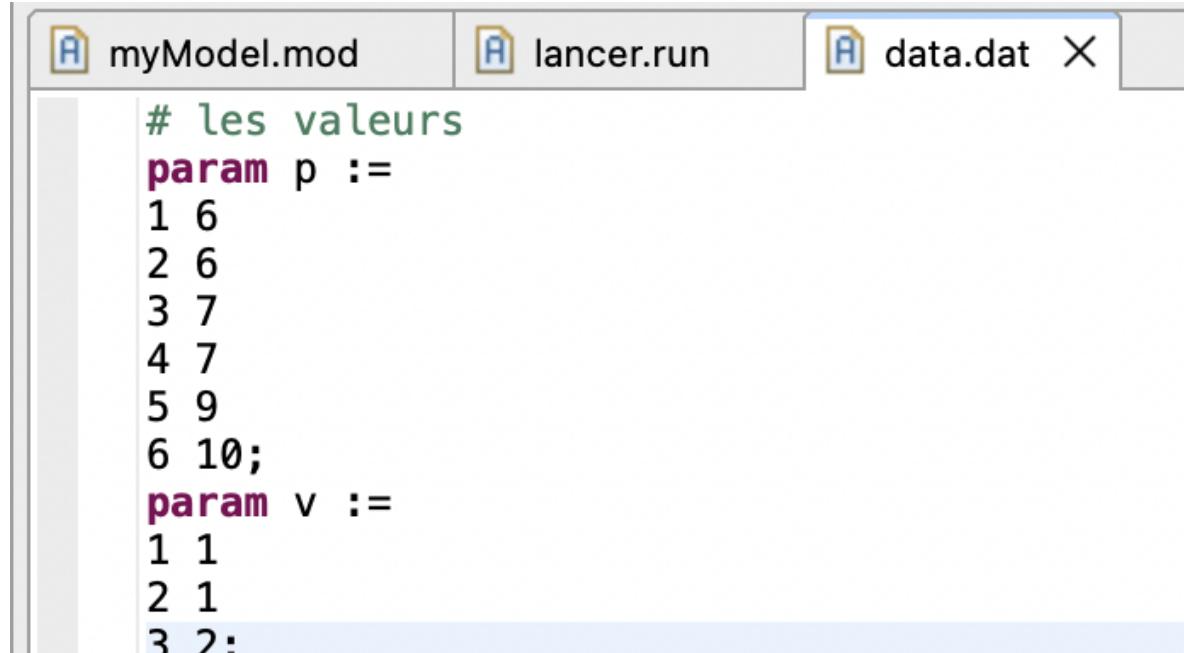
subject to C1{i in slots, m in machines}:
    sum{j in tasks} x[j, i, m] <= 1;

subject to C2{j in tasks}:
    sum{m in machines, i in slots} x[j, i, m] = 1;

subject to C3{m in machines}:
    opt >= sum{j in tasks, i in slots} ( p[j] / v[m] ) * x[j, i, m];
```

# Ordonnancement et Planification de la Production

- Exercice :



The screenshot shows a software interface with three tabs at the top: "myModel.mod", "lancer.run", and "data.dat". The "myModel.mod" tab is active, displaying the following code:

```
# les valeurs
param p :=
1 6
2 6
3 7
4 7
5 9
6 10;
param v :=
1 1
2 1
3 2;
```

# Ordonnancement et Planification de la Production

- Exercice :

```
myModel.mod    lancer.run X

reset;
model myModel.mod;
data data.dat;
option solver cplex;
solve;

display x;

printf "Ordonnancement :\n";
for {m in machines} {
    for {j in tasks} {
        for {i in slots : x[j, i, m] = 1} {
            printf "Tache %d dans la position : ", j;
            printf "%d ", i;
            printf "de la machine %d", m;
            printf "\n";
        }
    }
}

for {m in machines} {
    display sum{j in tasks, i in slots} ( p[j] / v[m] ) * x[j, i, m];
}
```

# Ordonnancement et Planification de la Production

- Exercice :

Ordonnancement :

Tache 6 dans la position : 1 de la machine 1

Tache 1 dans la position : 3 de la machine 2

Tache 2 dans la position : 2 de la machine 2

Tache 3 dans la position : 3 de la machine 3

Tache 4 dans la position : 1 de la machine 3

Tache 5 dans la position : 2 de la machine 3

$\sum_{j \in \text{tasks}, i \in \text{slots}} p[j]/v[m] * x[j, i, m] = 10$

$\sum_{j \in \text{tasks}, i \in \text{slots}} p[j]/v[m] * x[j, i, m] = 12$

$\sum_{j \in \text{tasks}, i \in \text{slots}} p[j]/v[m] * x[j, i, m] = 11.5$

Faire avec mêmes vitesses

29

Machine	P1	P2	P3
1	T6	-	-
2	-	T2	T1
3	T4	T5	T3

# Ordonnancement et Planification de la Production

- Exercice :

Ordonnancement :

Tache 1 dans la position : 2 de la machine 1

Tache 4 dans la position : 1 de la machine 1

Tache 3 dans la position : 1 de la machine 2

Tache 5 dans la position : 2 de la machine 2

Tache 2 dans la position : 1 de la machine 3

Tache 6 dans la position : 2 de la machine 3

$\sum_{j \in \text{tasks}, i \in \text{slots}} p[j]/v[m] * x[j, i, m] = 13$

$\sum_{j \in \text{tasks}, i \in \text{slots}} p[j]/v[m] * x[j, i, m] = 16$

$\sum_{j \in \text{tasks}, i \in \text{slots}} p[j]/v[m] * x[j, i, m] = 16$

Faire avec  $p_j = 1 \forall j$

30



Machine	P1	P2	P3
1	T4	T1	-
2	T3	T5	-
3	T2	T6	-

# Ordonnancement et Planification de la Production

- Exercice :

Ordonnancement :

Tache 1 dans la position : 1 de la machine 1

Tache 4 dans la position : 2 de la machine 1

Tache 2 dans la position : 1 de la machine 2

Tache 5 dans la position : 2 de la machine 2

Tache 3 dans la position : 1 de la machine 3

Tache 6 dans la position : 2 de la machine 3

$\sum_{j \in \text{tasks}, i \in \text{slots}} p[j]/v[m] * x[j, i, m] = 2$

$\sum_{j \in \text{tasks}, i \in \text{slots}} p[j]/v[m] * x[j, i, m] = 2$

$\sum_{j \in \text{tasks}, i \in \text{slots}} p[j]/v[m] * x[j, i, m] = 2$

Solution triviale !

31

Machine	P1	P2	P3
1	T1	T4	-
2	T2	T5	-
3	T3	T6	-

# Ordonnancement et Planification de la Production

- Exercice :
- Maximiser le nombre de tâches qui sont complétées sans retard
- $\max \sum_i x_i$
- $x_i = 1$  si la tâche  $i$  est complétée sans retard (s'il n'est pas possible de la compléter sans retard, alors elle ne sera pas exécutée et nous passerons à la tâche suivante.)
- Chaque tâche a une durée  $p_i$
- Chaque tâche doit respecter une date limite («due time»)  $d_i$
- $\sum_{i=1}^j p_i x_i \leq d_j, \forall j$ , le terme  $\sum_{i=1}^j p_i x_i$  est la date de réalisation de la tâche  $j$

32

Tâche	1	2	3	4	5	6
$p_i$	1	5	4	2	6	2
$d_i$	2	6	10	12	18	20

ON TIME ?

Si  $d_6 = 19$  ?

Considérer à présent qu'un poids est associé à chaque tâche  $w_6 = 10$  et  $w_i = 1, i \neq 6$   
La tâche 5 sera complétée sans retard ?



# Ordonnancement et Planification de la Production

- Exercice :

myModel.mod X

```
set S := {1..6};

param p{S};
param d{S};
param w{S};

var x{S} binary;

maximize ONTIME: sum{j in S} w[j] * x[j];

subject to C1{j in S}:
    sum{i in {1..j}} p[i] * x[i] <= d[j];
```

33

myModel.mod \*data2.dat X

```
param p :=  
1 1  
2 5  
3 4  
4 2  
5 6  
6 2;  
param d :=  
1 2  
2 6  
3 10  
4 12  
5 18  
6 19;  
param w :=  
1 1  
2 1  
3 1  
4 1  
5 1  
6 1;
```

# Ordonnancement et Planification de la Production

- Exercice :

myModel.mod X

```
set S := {1..6};

param p{S};
param d{S};
param w{S};

var x{S} binary;

maximize ONTIME: sum{j in S} w[j] * x[j];

subject to C1{j in S}:
    sum{i in {1..j}} p[i] * x[i] <= d[j];
```

myModel.mod X data2.dat X lancer.run

```
param p :=
1 1
2 5
3 4
4 2
5 6
6 2;
param d :=
1 2
2 6
3 10
4 12
5 18
6 20;
param w :=
1 1
2 1
3 1
4 1
5 1
6 1;
```

# Ordonnancement et Planification de la Production

- Exercice :

myModel.mod X

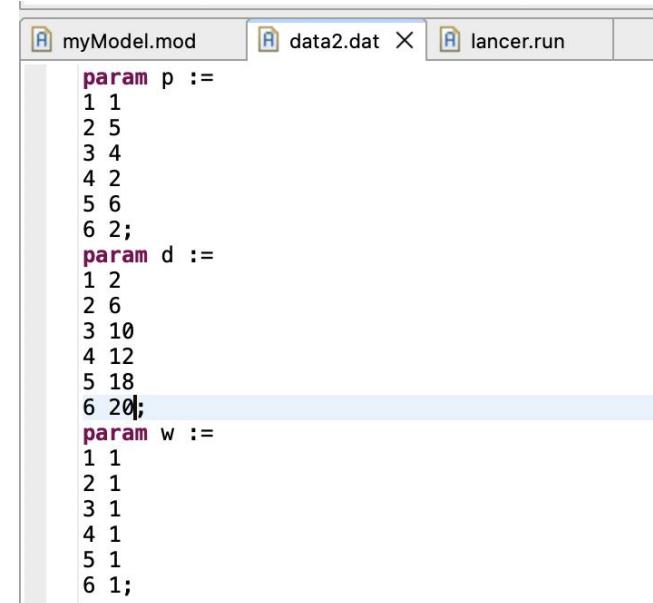
```
set S := {1..6};

param p{S};
param d{S};
param w{S};

var x{S} binary;

maximize ONTIME: sum{j in S} w[j] * x[j];

subject to C1{j in S}:
    sum{i in {1..j}} p[i] * x[i] <= d[j];
```



The screenshot shows a software interface for modeling and solving optimization problems. The main window displays the contents of the `myModel.mod` file, which contains the MIP model described above. To the right of the code editor, there are tabs for `data2.dat` and `lancer.run`, both of which are currently closed. The status bar at the bottom right of the interface shows the number "35".

```
param p :=  
1 1  
2 5  
3 4  
4 2  
5 6  
6 2;  
param d :=  
1 2  
2 6  
3 10  
4 12  
5 18  
6 20;  
param w :=  
1 1  
2 1  
3 1  
4 1  
5 1  
6 1;
```

# Ordonnancement et Planification de la Production

- Exercice :

```
myModel.mod    *data2.dat    lancer.run X
reset;
model myModel.mod;
data data2.dat;
option solver cplex;
solve;

display x;

printf "Fonction objectif (ONTIME) : %f\n", ONTIME;
printf "Ordonnancement :\n";
for {j in S} {
    printf "Tache %d est %s à jour \n", j, if x[j] == 1 then "" else "NOT";
}
for {j in S : x[j] = 1} {
    printf "tache %d :", j;
    display sum{i in {1..j}} p[i] * x[i];
}
display d;
```

```
Fonction objectif (ONTIME) : 6.000000
Ordonnancement :
Tache 1 est à jour
Tache 2 est à jour
Tache 3 est à jour
Tache 4 est à jour
Tache 5 est à jour
Tache 6 est à jour
tache 1 :sum{i in 1 .. j} p[i]*x[i] = 1
tache 2 :sum{i in 1 .. j} p[i]*x[i] = 6
tache 3 :sum{i in 1 .. j} p[i]*x[i] = 10
tache 4 :sum{i in 1 .. j} p[i]*x[i] = 12
tache 5 :sum{i in 1 .. j} p[i]*x[i] = 18
tache 6 :sum{i in 1 .. j} p[i]*x[i] = 20
d [*] :=
1 2
2 6
3 10
4 12
5 18
6 20
;
```

# Ordonnancement et Planification de la Production

- Exercice :

```
myModel.mod    *data2.dat    lancer.run X
reset;
model myModel.mod;
data data2.dat;
option solver cplex;
solve;

display x;

printf "Fonction objectif (ONTIME) : %f\n", ONTIME;
printf "Ordonnancement :\n";
for {j in S} {
    printf "Tache %d est %s à jour \n", j, if x[j] == 1 then "" else "NOT";
}
for {j in S : x[j] = 1} {
    printf "tache %d :", j;
    display sum{i in {1..j}} p[i] * x[i];
}
display d;
```

AMPL

Ordonnancement :

Tache 1 est à jour

Tache 2 est à jour

Tache 3 est à jour

Tache 4 est à jour

Tache 5 est à jour

Tache 6 est NOT à jour

tache 1 : $\sum_{i \in 1..j} p[i]*x[i] = 1$

tache 2 : $\sum_{i \in 1..j} p[i]*x[i] = 6$

tache 3 : $\sum_{i \in 1..j} p[i]*x[i] = 10$

tache 4 : $\sum_{i \in 1..j} p[i]*x[i] = 12$

tache 5 : $\sum_{i \in 1..j} p[i]*x[i] = 18$

d [\*] :=

1 2

2 6

3 10

4 12

5 18

6 19

;

# Ordonnancement et Planification de la Production

- Exercice :

```
set S := {1..6};

param p{S};
param d{S};
param w{S};

var x{S} binary;

maximize ONTIME: sum{j in S} w[j] * x[j];

subject to C1{j in S}:
    sum{i in {1..j}} p[i] * x[i] <= d[j];
```

```
param p :=
1 1
2 5
3 4
4 2
5 6
6 2;
param d :=
1 2
2 6
3 10
4 12
5 18
6 19;
param w :=
1 1
2 1
3 1
4 1
5 1|
6 10;
```

38

# Ordonnancement et Planification de la Production

- Exercice :

```
reset;
model myModel.mod;
data data2.dat;
option solver cplex;
solve;

display x;

printf "Fonction objectif (ONTIME) : %f\n", ONTIME;
printf "Ordonnancement :\n";
for {j in S} {
  printf "Tache %d est %s à jour \n", j, if x[j] == 1 then "" else "NOT";
}
for {j in S} {
  printf "tache %d :", j;
  if x[j] == 0 then {
    printf "devrait être executée à ";
    display p[j]+sum{i in {1..j}} p[i] * x[i];
  }else{
    printf "est executée à ";
    display sum{i in {1..j}} p[i] * x[i];
  }
}
display d;
```

```
Fonction objectif (ONTIME) : 14.000000
Ordonnancement :
Tache 1 est à jour
Tache 2 est à jour
Tache 3 est à jour
Tache 4 est à jour
Tache 5 est NOT à jour
Tache 6 est à jour
tache 1 :est executée à sum{i in 1 .. j} p[i]*x[i] = 1

tache 2 :est executée à sum{i in 1 .. j} p[i]*x[i] = 6

tache 3 :est executée à sum{i in 1 .. j} p[i]*x[i] = 10

tache 4 :est executée à sum{i in 1 .. j} p[i]*x[i] = 12

tache 5 :devrait être executée à p[j] + sum{i in 1 .. j} p[i]*x[i] = 18

tache 6 :est executée à sum{i in 1 .. j} p[i]*x[i] = 14

d [*] :=
1 2
2 6
3 10
4 12
5 18
6 19
;
```

# Ordonnancement et Planification de la Production

- Exercice :

```
Fonction objectif (ONTIME) : 32.000000
Ordonnancement :
Tache 1 est à jour
Tache 2 est à jour
Tache 3 est NOT à jour
Tache 4 est NOT à jour
Tache 5 est à jour
Tache 6 est à jour
tache 1 :est executée à sum{i in 1 .. j} p[i]*x[i] = 19
tache 2 :est executée à sum{i in 1 .. j} p[i]*x[i] = 48
tache 3 :devrait être executée à p[j] + sum{i in 1 .. j} p[i]*x[i] = 109
tache 4 :devrait être executée à p[j] + sum{i in 1 .. j} p[i]*x[i] = 120
tache 5 :est executée à sum{i in 1 .. j} p[i]*x[i] = 54
tache 6 :est executée à sum{i in 1 .. j} p[i]*x[i] = 67

d [*] :=
1   60
2   75
3   78
4  101
5  102
6  127
;
```

```
# les valeurs
param p :=
1 19
2 29
3 61
4 72
5 6
6 13;
param d :=
1 60
2 75
3 78
4 101
5 102
6 127;
param w :=
1 8
2 11
3 7
4 5
5 7
6 6;
```

# Ordonnancement et Planification de la Production

- Exercice :
- Considérons un atelier de production composé de n tâches à effectuer sur une seule machine.
- Chaque tâche  $i$  a une durée de traitement  $p_i$  et une date d'échéance  $d_i$  à laquelle elle doit être terminée.
- L'objectif est de minimiser le total des retards cumulés pour toutes les tâches/positions.
- Soit  $C_i$  la date de complétion de la tâche  $i$  et  $T_i$  le retard associé à la tâche  $i$ , défini comme  $T_i = \max(0, C_i - d_i)$ 
  - $T_i = \max(0, C_i - d_i) \rightarrow T_i \geq 0$  et  $T_i \geq C_i - d_i$
- Formulez le problème d'optimisation pour minimiser le total des retards.

41

Tâche	1	2	3	4	5
$p_i$	40	78	73	11	22
$d_i$	54	66	143	145	149

- Exemple : pour la position  $k=2$  on affecte la tâche 1 sachant que la tâche 3 a été affectée à  $k=1$ 
  - le retard cumulé est  $T_1 = \max(0, p_3 + p_1 - d_1)$

# Ordonnancement et Planification de la Production

- Exercice :
- $n$  tâches avec  $p_i$  et  $d_i$  la durée et la date limite (« due time ») de la tâche  $i$
- $x_{ik} = 1$  si la tâche  $i$  est affectée à la position  $k$ 
  - $x_{ki} = 1$  car il tâche et position sont dans le même ensemble  $\{1, \dots, n\}$
- Une tâche pour chaque position
- $\min \sum_k t_k$ ,
- Il faut calculer les dates de complétion cumulées (jusqu'à  $k$ ) :
  - $C_k = \sum_i p_i (\sum_{u=1}^k x_{iu})$ 
    - exemple : pour la position  $k=2$  avec 3 tâches  $x_{i,1:2} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \\ 1 & 0 \end{pmatrix} \rightarrow p_3 + p_1$
- Il faut calculer les « due times »
  - $d_k = \sum_i d_i x_{ik}$ 
    - exemple : pour la position  $k=2$  avec 3 tâches  $x_{i,2} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \rightarrow d_1$
- $\forall k, t_k = \max(0, C_k - d_k)$ 
  - $\sum_i p_i (\sum_{u=1}^k x_{iu}) - \sum_i d_i x_{ik} \leq t_k$  et  $0 \leq t_k$

42

# Ordonnancement et Planification de la Production

- Exercice :

```
*myModel.mod X lancer.run data.dat
set S := {1..5};

param p{S};
param d{S};

var x{S, S} binary;
var t{S} >= 0;

minimize TARDINESS: sum{k in S} t[k];

subject to C1{i in S}:
  sum{k in S} x[i, k] == 1;

subject to C2{k in S}:
  sum{i in S} x[i, k] == 1;

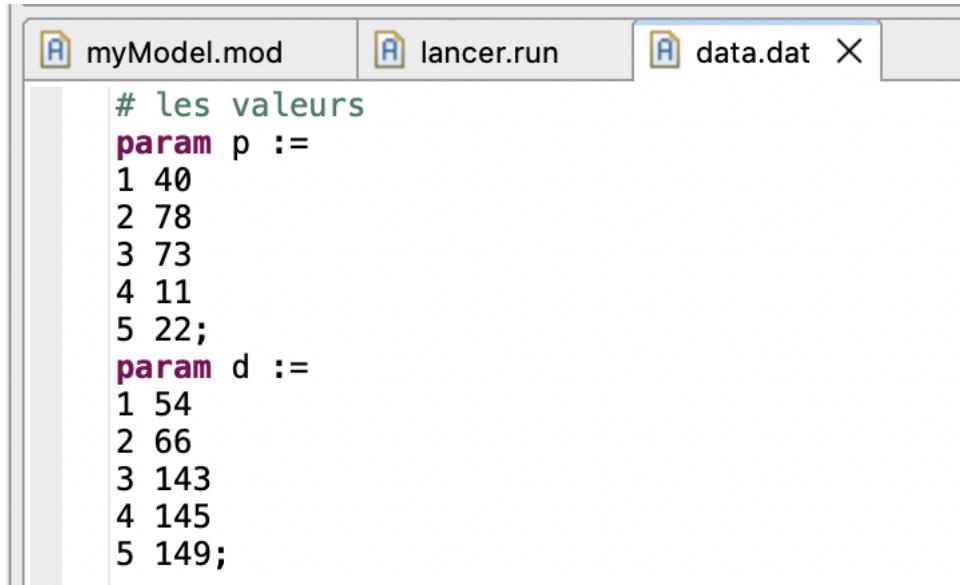
subject to C3{k in S}:
  sum{i in S} p[i] * sum{u in {1..k}} x[i,u] - sum{i in S} d[i] * x[i,k] <= t[k];
```

43



# Ordonnancement et Planification de la Production

- Exercice :



The screenshot shows a software interface with three tabs at the top: "myModel.mod", "lancer.run", and "data.dat". The "data.dat" tab is currently selected. The content of the "data.dat" tab is as follows:

```
# les valeurs
param p :=  
1 40  
2 78  
3 73  
4 11  
5 22;  
param d :=  
1 54  
2 66  
3 143  
4 145  
5 149;
```

# Ordonnancement et Planification de la Production

- Exercice :



```
myModel.mod lancer.run X data.dat

reset;
model myModel.mod;
data data.dat;
option solver cplex;
solve;

display x;
display t;

printf "Fonction objectif (TARDINESS) : %f\n", TARDINESS;
printf "Ordonnancement :\n";
for {i in S} {
    printf "Tache %d dans la position : ", i;
    for {j in S : x[i, j] = 1} {
        printf "%d ", j;
    }
    printf "\n";
}

for {k in S} {
    printf "complétude pour la position %d", k;
    display sum{i in S} p[i] * sum{u in {1..k}} x[i,u];
}

for {k in S} {
    printf "lateness pour la position %d", k;
    display sum{i in S} p[i] * sum{u in {1..k}} x[i,u] - sum{i in S} d[i] * x[i,k];
}
```

45

# Ordonnancement et Planification de la Production

- Exercice :

```
t [*] :=
1 0
2 52
3 0
4 2
5 81
;

Fonction objectif (load) : 135.000000
Ordonnancement :
Tache 1 dans la position : 1
Tache 2 dans la position : 2
Tache 3 dans la position : 5
Tache 4 dans la position : 3
Tache 5 dans la position : 4
completitude pour la position 1sum{i in S} p[i]*(sum{u in 1 .. k} x[i,u]) = 40
completitude pour la position 2sum{i in S} p[i]*(sum{u in 1 .. k} x[i,u]) = 118
completitude pour la position 3sum{i in S} p[i]*(sum{u in 1 .. k} x[i,u]) = 129
completitude pour la position 4sum{i in S} p[i]*(sum{u in 1 .. k} x[i,u]) = 151
completitude pour la position 5sum{i in S} p[i]*(sum{u in 1 .. k} x[i,u]) = 224
lateness pour la position 1sum{i in S} p[i]*(sum{u in 1 .. k} x[i,u]) - sum{i in S} d[i]*x[i,k] = -14
lateness pour la position 2sum{i in S} p[i]*(sum{u in 1 .. k} x[i,u]) - sum{i in S} d[i]*x[i,k] = 52
lateness pour la position 3sum{i in S} p[i]*(sum{u in 1 .. k} x[i,u]) - sum{i in S} d[i]*x[i,k] = -16
lateness pour la position 4sum{i in S} p[i]*(sum{u in 1 .. k} x[i,u]) - sum{i in S} d[i]*x[i,k] = 2
lateness pour la position 5sum{i in S} p[i]*(sum{u in 1 .. k} x[i,u]) - sum{i in S} d[i]*x[i,k] = 81
```

**Maintenant, c'est  
à vous !**

47



## En détails

- Travail en autonomie (binôme)
- Faire la liste d'exercices
- Choisir l'outil : AMPL, Matlab (yalmip), PuLP, etc...

48

