Table des matières

1 Achat des programmes :

1.1 Enoncé :

Une société de communication négocie des droits d'exploitation sur plusieurs années d'une chaîne de télévision à couverture nationale. Un contrat l'oblige à diffuser pendant cette période au moins 1800 heures de fictions françaises, 1800 heures de fictions européennes et 1200 heures de reportages.
Pour acheter les programmes, elle se fournit auprès de deux entreprises de production télévisuelle qui vendent des ensembles d'émissions « clé en main » appelés « lots ».
La société Damol propose des lots composés de la manière suivante :

- 90 heures de fictions françaises
- 120 heures de fictions européennes
- 40 heures de reportages

Chaque lot coûte 20 000 M€.
La société Martel propose des lots composés de la manière suivante :

- 90 heures de fictions françaises
- 60 heures de fictions européennes
- 100 heures de reportages

Chaque lot coûte 24 000 M€.
Questions :
1. Déterminer l'achat qui permet de minimiser le coût des programmes achetés par la société.
2. Faites un graphique permettant d'interpréter graphiquement la résolution optimale

1.2 Données :

	Damol	Martel
Fictions françaises	90	90
Fictions européennes	120	60
Reportages	40	100

Prix du lot de Damol	Prix du lot de Martel
20000	24000

1.3 Modèle :

Données:

- P_1 : prix du lot de la société Damol
- P_2 : prix du lot de la société Martel

- ff : nombre minimum d'heures de fictions françaises
- fe : nombre minimum d'heures de fictions européennes

- fr : nombre minimum d'heures de reportages
- df : nombre d'heures de fictions françaises dans le lot de la société Damol
- de : nombre d'heures de fictions européennes dans le lot de la société Damol
- dr : nombre d'heures de reportages dans le lot de la société Damol
- mf : nombre d'heures de fictions françaises dans le lot de la société Martel
- me : nombre d'heures de fictions européennes dans le lot de la société Martel
- mr : nombre d'heures de reportages dans le lot de la société Martel

Variables:

- X : nombre de lots achetés de la société Damol ;
- Y : nombre de lots achetés de la société Martel

Contraintes:

df.X + mf.Y ≥ ff ; de.X + me.Y ≥ fe ; fr.X + mr.Y ≥ dr

Objectif:

Minimiser (P_1.X + P_2.Y)

1.4 Solution:

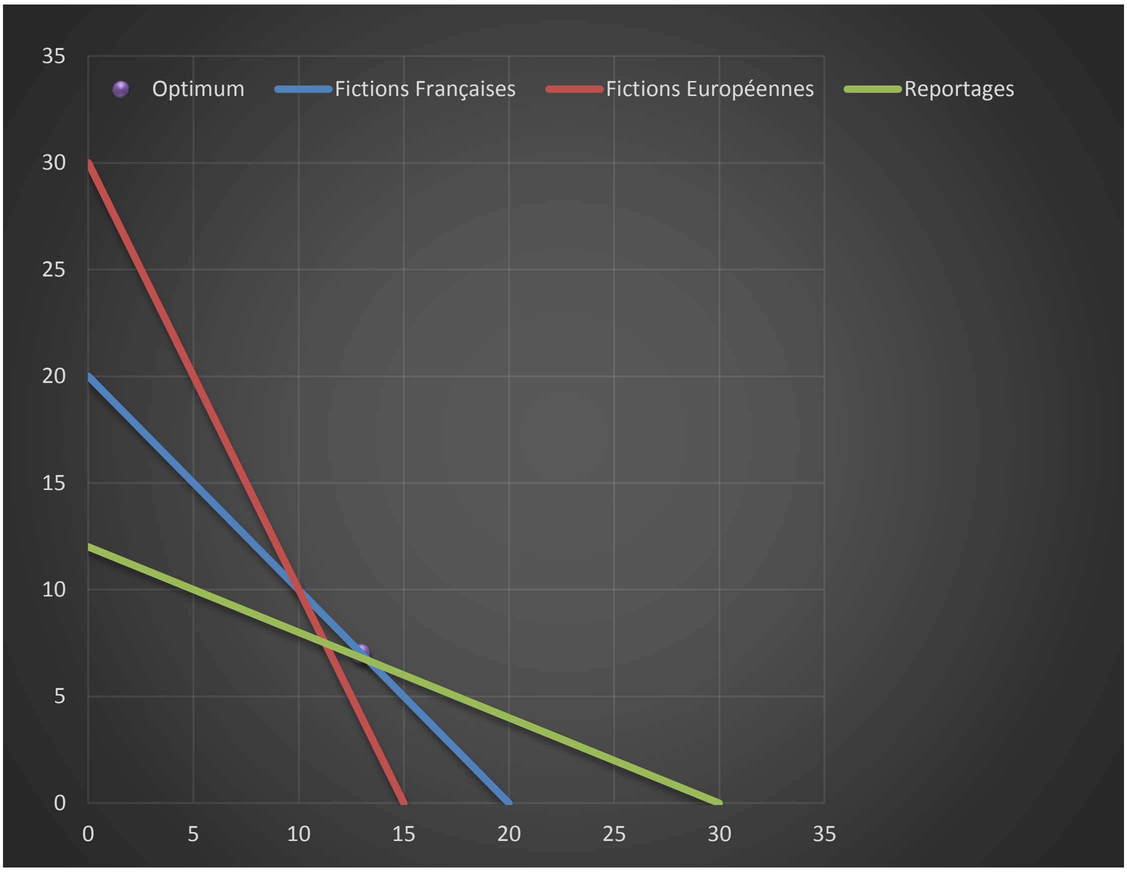
Après avoir établie le module en résout le problème en utilisant le simplexe PL et on obtient les résultats suivants:

Variables	Nombre des lots achetés de Damol	Nombre des lots achetés de Martel
	13	7

Objectif	428000

1.5 Graphes :

En utilisant Excel on peut obtenir un graphe de la forme suivante :



1.6 Mathematica :

Voir fichier (Exercice1.nb)

D'abord la commande suivante va nous permettre de résoudre le problème dans l'ensemble des réels :

$$\text{Minimize}[\{20000 * X + 24000 * Y, X + Y >= 20\&\&2 * X + Y >= 30\&\&2 * X + 5 * Y >= 60\}, \{X, Y\}]$$

On obtient alors la solution suivante :

$$\{\frac{1280000}{3}, \{X \to \frac{40}{3}, Y \to \frac{20}{3}\}\}$$

Ensuite on déclare que les variables X et Y sont réels :

$$\text{Minimize}[\{20000 * X + 24000 * Y, X + Y >= 20\&\&2 * X + Y >= 30\&\&2 * X + 5 * Y >= 60\&\&(X|Y) \in \text{Integers}\}, \{X, Y\}]$$

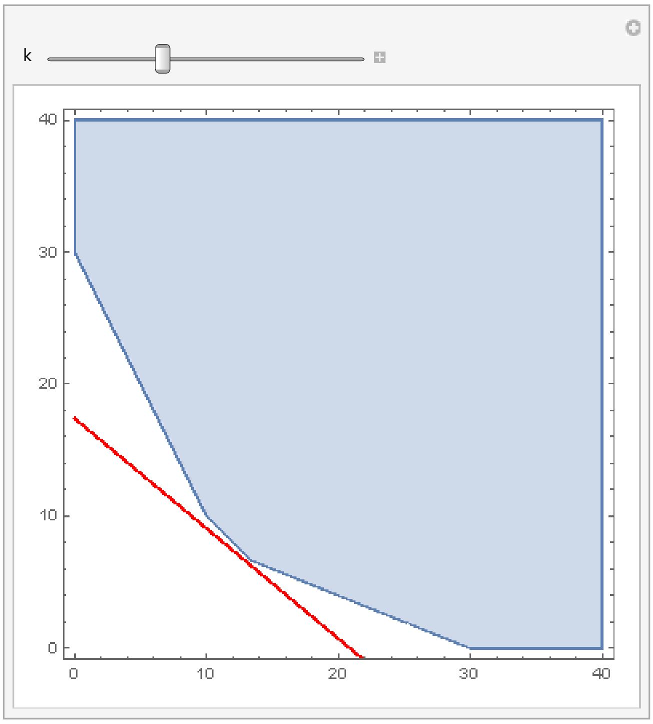
Et on obtient :

$$\{428000, \{X \to 13, Y \to 7\}\}$$

Finalement on a trouvé la même solution que les autres solveurs, on remarque que les solutions entières ne sont pas toujours la partie entière de la solution réelle. Dans la suite on va créer une animation qui nous permettra de regarder le graphe et l’évolution de l'objectif pour obtenir le minimum, j’ai simplifié les contraintes pour des raisons de clarté :

$$\mathrm{Manipulate}[\mathrm{Show}[\mathrm{RegionPlot}[x + y >= 20\&\&2 * x + y >= 30\&\&2 * x + 5y >= 60, \{x, 0{,}40\}, \{y, 0{,}40\}], \mathrm{Plot}[-5 * x/4 + k/1000\,, \{x, 0{,}40\}, \mathrm{PlotStyle} \rightarrow \{\mathrm{Red}, \mathrm{Thick}\}]], \{k, 0{,}50000\}]$$

Et de cette façon on construit l’animation présente dans le dossier : manipulate1.swf
(s’ouvre avec Google chrome if any)



1.7 Résolution avec Ampl :

Tout d'abord on construira un fichier de model, j'ai simplifié les contraintes et l'objectif en factorisant par 1000 dans l'objectif et on divisant les contraintes par des nombres simples par raison de simplicité et de compréhension :

1.7.1 Exercise1.mod

Dans cet exercice simple on n'aura pas besoin de faire un fichier « .dat », on considère deux variables X et Y correspondant nombre de lots acheté de la société Damol et Martel respectivement et on écrit le modèle simple sous forme standard on respectant la syntaxe :

```
var X;
var Y;

minimize Cost 20*X+24*Y;
s.t Francaise : X+Y     >= 20;
s.t Europienne: 2*X+Y   >= 30;
s.t Reportage : 2*X+5*Y >= 60;
```

Dans la suite on va créer un fichier « .run » qui est un script pour résoudre le problème. Puisque le problème est linéaire et facile à résoudre avec le simplexe alors on peut utiliser le solveur « cplex » déjà programmé dans AMPL :

1.7.2 Exercise1.run

```
reset;
model Exercise1.mod;
option solver cplex;
solve;
display X,Y,Cost;
```

Ainsi on tapant la commande `include Exercise1.run` on obtient la solution :

```
ampl: reset;
ampl: include Exercise1.run;
CPLEX 12.6.0.0: optimal integer solution; objective 428
3 MIP simplex iterations
0 branch-and-bound nodes
X = 13
Y = 7
Cost = 428

ampl:
```

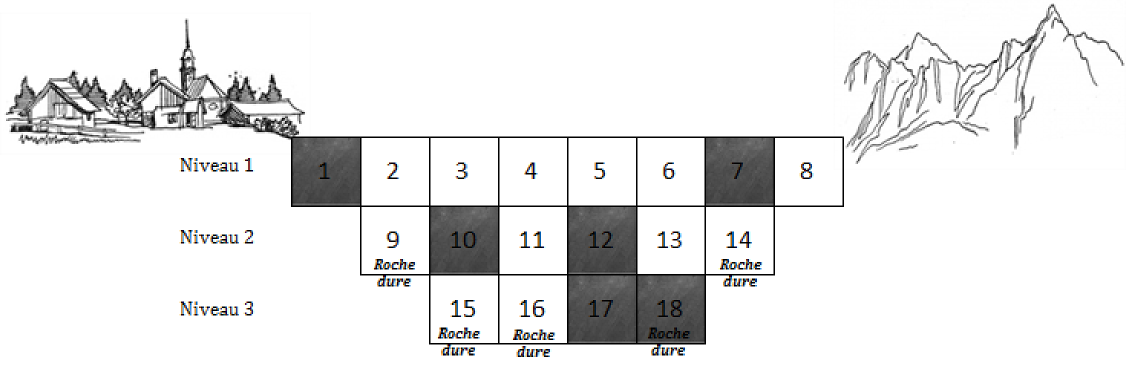
`Ceci veut dire que le cost total est 428 000 M€, car on a factorisé par 1000 au début.`

2 Mine :

2.1 Enoncé :

Une mine à ciel ouvert se présente comme indiqué sur le schéma présent en bas.
La mine est découpée en blocs appelés unités d'exploitation appelés blocs.
La mine doit être creusée en gradins afin de permettre aux camions d'y descendre. Elle est limitée à l'ouest par un village et à l'est pas une montagne.
18 blocs de 10000 tonnes ont été délimités. Pour extraire un bloc, il faut que les 3 blocs (au-dessus, au-dessus à gauche, au-dessus à droite) du niveau supérieur aient été extraits. Ceci permet de respecter les contraintes de pente. Seuls les blocs en gris contiennent de l'uranium et ont donc une valeur marchande.
Les coûts d'extraction et les valeurs marchandes sont indiqués dans le tableau ci-dessous (à noter que les blocs de roche dure sont coûteux à extraire).

1. Quels sont les blocs à extraire pour maximiser le bénéfice total ?



Vue en coupe de la mine

2.2 Données :

Pour simplifier la propagation des formules dans les cellules d'Excel on représente les données avec la même forme que la mine. En effet pour les contraintes et les variables il est très difficile de manipuler à chaque fois les cellules si on écrit toutes les valeurs dans une seule ligne, par contre ici on a une vision très claire des données chaque nombre dans les tableaux suivants correspond au cout d'extraction, présence d'uranium ou valeurs marchandises du bloc existant dans la cellule correspondante :

Coûts extraction C_i							
100	100	100	100	100	100	100	100
	1000	200	200	200	200	1000	
		1000	1000	300	1000		

Présence d'uranium							
oui	non	non	non	non	non	oui	non
	non	oui	non	oui	non	non	
		non	non	oui	oui		

Valeurs marchandes P_i							
200	0	0	0	0	0	300	0
	0	500	0	200	0	0	
		0	0	1000	1200		

2.3 Modèle :

Données :

C_i : Coût d'extraction par tonne du bloc i, 1 ≤ i ≤ 18

P_i : Valeur marchande par tonne du bloc i, 1 ≤ i ≤ 18

Variables :

X_i : variable binaire égale à 1 si on entracte le bloc i, 0 sinon, 1 ≤ i ≤ 18

Contraintes : la seule contrainte qui existe est :« Pour extraire un bloc, il faut que les 3 blocs (au-dessus, au-dessus à gauche, au-dessus à droite) du niveau supérieur aient été extraits », cette contrainte s'exprime facilement avec les opération booléennes '*'et '-' , par exemple pour extraire X9 il faut extraire X1 , X2 et X3 c'est exactement ce que est traduit par la première équation : Si X9=1 alors nécessairement X1=X2=X3=1 :

- $X1 + X2 + X3 - 3*X9 \geq 0$
- $X2 + X3 + X4 - 3*X10 \geq 0$
- $X3 + X4 + X5 - 3*X11 \geq 0$
- $X4 + X5 + X6 - 3*X12 \geq 0$
- $X5 + X6 + X7 - 3*X13 \geq 0$
- $X6 + X7 + X8 - 3*X14 \geq 0$
- $X9 + X10 + X11 - 3*X15 \geq 0$
- $X10 + X11 + X12 - 3*X16 \geq 0$
- $X11 + X12 + X13 - 3*X17 \geq 0$
- $X12 + X13 + X14 - 3*X18 \geq 0$

Objectif :

Maximiser $\sum_1^{18}(Xi * Pi - Ci * Xi)$

2.4 Résultats :

On remarque que les seuls blocs à extraire sont X1 et X7, en fait si on veut extraire ceux qui sont à l'intérieur on va dépenser plus. Le modèle retenue nous permet de faire une présentation facile et simple des résultats, et ceci nous éviter surtout de regarder quelle cellule correspond à quel Xi !

Objectif
400000

Solution

1	1	1	1	1	1	1	0
	0	1	1	1	1	0	
		0	0	1	0		

3 Ordonnancement de tâches :

3.1 Enoncé :

Une entreprise de peintures doit fabriquer chaque semaine 5 lots de peintures. Chaque lot est fabriqué en une seule fois dans une cuve à l'aide d'un mélangeur. Les durées de fabrication des peintures sont données dans le tableau ci-dessous.
Pour fabriquer une couleur, il faut d'abord nettoyer la cuve mais les temps de nettoyage dépendent de la couleur à fabriquer et de celle qu'il faut nettoyer. Ainsi, passer du noir au blanc est très coûteux car il faut une cuve très propre pour faire du blanc tandis que passer du jaune au noir est moins coûteux.
Plus généralement, passer d'une fabrication à une autre implique des temps de réglage qui peuvent être très importants (quelques fois plusieurs jours !) donc il est utile d'optimiser les enchaînements.
La matrice n×n DN donne les temps de nettoyage.
Comment minimiser la durée totale de fabrication des n lots de peinture sachant que de semaine en semaine l'ordre sera le même et qu'il faut terminer la fabrication de manière à ce que la cuve soit prête à fabriquer la couleur qui commence le cycle en semaine suivante (prendre la couleur 1 comme couleur à fabriquer en premier) ?
Le modèle, assez difficile, est fourni.

1. Expliquer pourquoi un algorithme naïf énumérant tous les ordonnancements est inapplicable en pratique ? (se placer bien sûr dans le cas général).
2. Expliquer précisément pourquoi la dernière famille de contrainte interdit les sous-cycles.
3. Implémenter ce modèle avec un solveur.

3.2 Modèle (donné)

Modèle

Données

n= *nombre de lots*

$$DF_i = durée\ de\ fabrication\ du\ lot\ i$$

$$DN_{ij} = durée\ de\ nettoyage\ entre\ les\ lots\ i\ et\ j$$

Inconnues

$$X_{ij} = 1\ si\ le\ lot\ j\ succède\ au\ lot\ i,\ inconnues\ binaires$$

$$R_i = rang\ du\ lot\ i\ dans\ la\ fabrication,\ inconnues\ réelles\ \geq 0$$

Contraintes

$$\forall i = 1 \dots n: \sum_{j=1, j\neq i}^{n} X_{ij} = 1 \quad chaque\ lot\ a\ un\ successeur\ dans\ la\ fabrication$$

$$\forall j = 1 \dots n: \sum_{i=1, i\neq j}^{n} X_{ij} = 1 \quad chaque\ lot\ a\ un\ prédécesseur\ dans\ la\ fabrication$$

$$R_1 = 1 \qquad le\ lot\ 1\ est\ fabriqué\ en\ premier$$

$$\forall i = 1 \dots n,\ \forall j = 2 \dots n,\ i \neq j:\ R_j \geq R_i + 1 - n\left(1 - x_{ij}\right)\ pour\ éviter\ les\ sous-cycles$$

Objectif *(minimiser la durée totale de la fabrication et des temps de nettoyage)*

$$Min\left(\sum_{i=1}^{n} \sum_{j=1,\ j\neq i}^{n} \left(DF_i + DN_{ij}\right)X_{ij}\right)$$

3.3 Réponses aux questions

1. Un algorithme naïf énumérant tous les ordonnancements est inapplicable en pratique, en effet ce dernier à une complexité factorielle en n !, et ce genre des algorithmes sont inutile car le temps nécessaire pour faire un tel calcul est gigantesque, par exemple si (n=100) ce qui est très possible et on utilisant un calculateur (processeur, …) avec une vitesse de 10^8 ops/sec alors on obtient le résultat après 10^{142} années !!!! .
2. La dernière famille des opérations garantit bien que les cycles sont interdites : supposons par exemple qu'il existe un cycle $(i_1, i_2, \ldots, i_m)$ avec $i_1 = i_m$ et alors$x_{i_k i_{k+1}} = 1$ et par suite on aura :
   $R_j \geq R_i + 1 - n(1 - x_{ij})$ $pour\ i = i_k et\ j = i_{k+1}$

   Et on faisant une relation de récurrence simple ceci nous conduit à

   $$R_{i_1} \geq R_{i_1} + m$$

   Ce qui est impossible.
   Signalons également que la relation n'a aucune signification autre que les valeurs des rangs de j est supérieure à la valeur du rang de i lorsque j succède i, dans le cas contraire c'est à dire lorsque j ne succède pas i ou formellement $x_{ij} = 0$ alors la formule devient :

   $$R_j \geq R_i + 1 - n$$

   Qui est toujours vrai puisque tous les R_i $sont\ positifs\ inférieurs$ à n .

3.4 Donnes

On les considère comme ils sont donnés dans l'énoncé :

n
5

Durée de fabrication des lots DF(i)					
Lots	**1**	**2**	**3**	**4**	**5**
Durée	40	35	45	32	50

Temps de nettoyage entre lots DN(i,j)					
Lots	**1**	**2**	**3**	**4**	**5**
1	0	11	7	13	11
2	5	0	13	15	15
3	13	15	0	23	11
4	9	13	5	0	3
5	3	7	7	7	0

3.5 Solution :

La manière de présenter les choses est importante dans cet exercice aussi, en effet puisque on est face à des variables de la formes x_{ij} alors naturellement on utilise une forme matricielle, ceci nous évite de confondre les variables et les cellules correspondante et surtout de ne pas taper toutes les contraintes à la main (qui est fastidieux). Ainsi comme dans le fichier Excel je retiens une représentation des variables sous forme de tableau dont les cases hachées sont inutiles.

Après avoir résolu le problème avec GRL Non linéaire on obtient l'ordre de fabrication suivant (déjà la résolution prend environ 10 secondes) :

$\longrightarrow$

Lot 1	Lot 4	Lot 3	Lot 2	Lot 5

Objectif	243

Ordres				
0	0	0	1	0
1	0	0	0	0
0	0	0	0	1
0	0	1	0	0
0	1	0	0	0

Rang				
1	5	3	2	4

3.6 Résolution ampl :

3.6.1 Data

```
data;
param n:=5;
param durre:=1      40
             2            45
             3            35
             4            32
             5            50     ;
param DNettoyage(tr):1          2          3          4          5 :=
            1      0          11         7          13         1
            2      5          0          13         15         15
            3      13         15         0          23         11
            4      9          13         5          0          3
            5      3          7          7          7          0;
```

3.6.2 Model

```
param n>0;

param durre{1..n}>=0;
param DNettoyage{1..n,1..n}>=0;
var Succ{1..n,1..n} , binary;
var Rang {1..n}>=0;


minimize LaDurreTotale :sum {i in 1..n, j in 1..n}
      (durre[i] + DNettoyage[i,j])*Succ[i,j];
  s.t. InitRang: Rang[1]=1;
  s.t. Initials{p in 1..n}: Succ[p,p]=0;
  s.t. Successeur{i in 1..n}: sum{j in 1..n} Succ[i,j]=1;
  s.t. Precesseur{j in 1..n}: sum{i in 1..n} Succ[i,j]=1;
  s.t. Noncycles{i in 1..n,j in 2..n: j<i or i<j}: Rang[j]>= Rang[i]+1-n*(1-
Succ[i,j]);
```

3.6.3 Script

```
reset;
model ordo_taches.mod;
data  ordo_taches.dat;
solve;
display Rang,Succ,LaDurreTotale;
```

4 Chargement Wagons :

4.1 Enoncé :

Trois wagons de chemins de fer de charge utile limitée à 100 quintaux sont réservés pour transporter 16 caisses. Les caisses et leurs poids en quintaux sont donnés dans le tableau à droite.
1. Comment affecter les caisses aux wagons en respectant les charges utiles et en minimisant la charge du wagon le plus chargé ?
2. Est-ce que la phrase : "Si la somme des poids des caisses est plus petite que la somme des capacités des wagons alors le modèle a une solution" est correcte ? Si non, donnez un exemple qui prouve qu'elle est incorrecte.
Présentez les résultats dans 2 tableaux :
Un tableau indiquant les wagons dans lesquels les caisses sont chargées.
Un tableau présentant les charges des wagons

4.2 Données

K	100

Caisse	1	2	3	4	5	6	7
Poids	28	6	8	17	16	5	13

8	9	10	11	12	13	14	15	16
21	19	22	14	13	25	9	21	25

4.3 Modèle :

Modèle :

Données:

- PM : Poids utile maximum des chariots
- P_i : Poids par quintaux de la caisse i, $1 \le i \le 16$

Variables:

pour tout $1 \le i \le 3, 1 \le j \le 16$ X_{ij} : Variable binaire égale à 1 si le chariot i porte la caisse j, 0 sinon.

Contraintes:

- $pour\ 1 \le i \le 3 \sum_1^{16}(X_{ij} * P_j) \le \mathrm{PM}$
- pour $1 \le j \le 16$ $X_{1j} + X_{2j} + X_{3j} = 1$

Objectif:

$Min(Max(\sum_1^{16}(X_{ij} * P_j))\ )\ 1 \le i \le 3\ ]\ )$

4.4 Solution :

De la même façon que les exercices qui précèdent, on utilise une représentation matricielle des variables ce qui facilitent l'écriture des contraintes et la résolution du problème :

Et on obtient dans le tableau des contraintes :

Objectif	88

Caisse n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
chariot 1	1	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0
chariot 2	0	0	0	0	0	0	0	0	1	0	0	1	0	1	1	1
chariot 3	0	1	0	0	0	0	0	1	0	1	1	0	1	0	0	0

Le chariot	charge	caisses
Chariot 1	87	1,3,4,5,6,7
Chariot 2	87	9,12,14,15,16
Chariot 3	88	2,8,10,11,13

On remarque que si la somme totale des poids des caisses est inférieure à la capacité totale des chariots cela n'implique pas que le transport peut s'effectuer. En effet, si par exemple toutes des caisses ont un poids égale 1 sauf une qui a un poids égal à 105 alors la somme totale des poids des caisses est 115 inférieures à 300. Alors que, la caisse à 105 ne peut être transportée dans aucun chariot car leur capacité ne la supporte pas.

5 Planification des travailleurs

5.1 Enoncé :

L'objectif de cette application est de planifier les emplois du temps de travailleurs. Ils doivent tous travailler 5 jours parmi les 7 de la semaine. Ils doivent donc disposer de 2 jours de repos, consécutifs ou pas. Une contrainte impose que le nombre de travailleurs disposant de 2 journées de repos non consécutives ne dépasse pas un certain pourcentage du nombre de travailleurs total (tester 10%, 20%, 30%, 40%).
L'entreprise demande à ce que le nombre de travailleurs présents chaque jour de la semaine soit au moins égal à une valeur déterminée (tableau ci-contre à droite).
Le taux horaire du salaire en semaine sera différent de celui du weekend.

1. Calculer le planning qui minimise la paie des travailleurs en respectant les contraintes.

Faites clairement apparaitre les résultats (inspirez-vous de la copie d'écran - attention, cette partie ne comporte que des formules Excel et aucune recopie manuelle).

Jours	Travailleurs
Lundi	20
Mardi	15
Mercredi	25
Jeudi	20
Vendredi	25
Samedi	15
Dimanche	10

Taux de salaire horaire en semaine					8,00 €
Taux de salaire horaire pour le weekend					12,00 €
Pourcentage maximum des travailleurs disposant de jours de repos non consécutifs par rapport au nombre total de travailleurs					10,00%

5.2 Modèle :

<u>Données :</u>

- p1 : Le taux de salaire horaire en semaine
- p2 : le taux de salaire horaire en WE
- r : Le pourcentage maximum des travailleurs disposant de jours de repos non consécutifs
- n_i : nombre de travailleurs requis le jour i, $1 \leq i \leq 7$

<u>Variables :</u>

- n : Le nombre total des travailleurs
- X_i : nombre de travailleurs présents le jour i, $1 \leq i \leq 7$
- $Y_{i,j}$: nombre de travailleurs qui ne travaillent pas le jour i et le jour j , $1 \leq i < j \leq 7$

<u>Contraintes :</u>

<u>Contrainte 1 :</u> Le nombre minimum de travailleurs par jour :

- $X_i \geq n_i$pour $1 \leq i \leq 7$

<u>Contrainte 2</u> : Le pourcentage maximum est r :

- $\sum_{i,j>i+1} Y_{ij} \leq r * n$

<u>Contrainte 3 :</u> Les travailleurs travaillent 5 jours et se reposent 2 jours :

- $\sum_{j=1,j\neq i}^{7} Y_{ij} + X_i = n$ pour tout $1 \leq i \leq 7$

<u>Objectif :</u>

$$Min(p1 * 8 * \sum_{i=1}^{5} X_i + p2 * 8 * (X_6 + X_7))$$

5.3 Solution :

Comme dans les exercices précédents on représente les variables Y_{ij} dans un tableau dans (la moitié est inutile) puisque i>j toujours, on considère aussi que le nombre de travailleurs est variables ce qui va nous permettre de trouver la meilleur solution comme ci-dessous :

Ainsi la résolution nous permet de trouver les résultats suivants :

Nombre Travailleurs	
	40

Objerctif	9 280,00 €

Dans le tableau suivant on représente le nombre de travailleurs en repos pour deux jours donnés:

Yij	Lundi	Mardi	Mercredi	Jeudi	Vendredi	Samedi	dimanche
Lundi		18	0	0	0	0	1
Mardi			7	0	0	0	0
Mercredi				5	0	0	3
Jeudi					15	0	0
Vendredi						0	0
Samedi							25
Dimanche							

Et dans le tableau suivant on représente le nombre de travailleurs présent chaque jour:

Lundi	21
Mardi	15
Mercredi	25
Jeudi	20
Vendredi	25
Samedi	15
dimanche	11

6 Packing circle

6.1 Discussion et modèle :

Nous allons examiner k cercles de même rayons r> 0. Nous supposerons que les centres des cercles ont coordonne x(i) et y(i) pour i = 1, ..., k: ce sont des variables à déterminer. En outre, dans les cas avec les milieux identiques, nous allons optimiser leur rayon; Ainsi, lorsqu'on expose ces hypothèses et on considère des cercles identiques sur le carré unité qui est, par hypothèse, centrée autour de l'origine, le modèle d'optimisation le plus simple est formulé comme suit:

$$\max(r)$$

$$|x(i)| + r \leq \frac{1}{2} \qquad for\ i\ in\ 1..k$$

$$|y(i)| + r \leq \frac{1}{2} \qquad for\ j\ in\ 1..k$$

$$2r \leq \sqrt{\big(x(i) - x(j)\big)^2 + \big(y(i) - y(j)\big)^2} \ \ for\ i\ in\ 1..k \quad for\ j\ in\ 1..k\ i < j$$

On remarque que ce problème n'est pas linéaire et n'est pas convexe, le nombre de variables est 2k, le nombre de contraintes convexes est 2k (les deux première lignes) et le nombre de contraintes non convexes (quadratiques) est k(k-1)/2.

En utilisant la méthode de copier le modèle et le coller on obtient une erreur :

```
ampl: reset;
ampl: include packing_circle.run;
CPLEX 12.6.0.0: QP Hessian is not positive semi-definite.
No basis.
r = 0

:   x   y    :=
1   0   0
2   0   0
3   0   0
4   0   0

;
```

Cette erreur veut dire tout simplement que le problème ne peut pas être résolu avec les paramètres utilisé car il est non linéaire et même si quadratique il n'est pas convexe (l'héssienne est non positive semi définie). Ainsi AMPL nous confirme le fait que le problème est non convexe, essayons alors autre chose.

En cherchant sur internet j'ai trouvé une autre formalisation plus utilisé, qui est la suivante :

$$\max(r)$$

$$r \leq x(i) \leq 1 - r \qquad for\ i\ in\ 1..k$$

$$r \leq y(i) \leq 1 - r \qquad for\ j\ in\ 1..k$$

$$2r \leq \sqrt{\left(x(i) - x(j)\right)^2 + \left(y(i) - y(j)\right)^2} \ for\ i\ in\ 1..k \ \ for\ j\ in\ 1..k\ i < j$$

Le modèle reste le même c'est juste le carrée il n'est plus centré en (0,0), dans ce cas on utilisera le solveur MINOS qui est un solveur intégrer dans AMPL pour résoudre des problèmes d'optimisation mathématiques linéaires et non linéaires. Il peut être utilisé pour la programmation linéaire, la programmation quadratique, et plus générale des fonctions et des contraintes objectives, et pour trouver un point faisable pour un ensemble de linéaires ou non linéaires égalités et inégalités.

L'idée est la suivante :

- Fixer une solution négative
- Résoudre le problème avec des valeurs aléatoire et on trouvera une solution intermédiaire « maximum local »
- Update solution si le maximum local >= solution
- Répéter les étapes précédentes plusieurs fois (200 fois).

Voilà les résultats:

6.1.1 File .mod

```
param N=2;

var r:=Uniform(0,1), >=0, <=1;
var x{j in 1..N} := Uniform(0,1), >= 0, <=1;
var y{j in 1..N} := Uniform(0,1), >= 0, <=1;


maximize rayon: r;
s.t. Non_inter{i in 1..N,j in 1..N :i<j}: 4*r**2<=(x[i]-x[j])**2+(y[i]-y[j])**2;
s.t. min1{i in 1..N}: x[i]>=r;
s.t. min2{i in 1..N}: y[i]>=r;
s.t. max1{i in 1..N}: x[i]<=1-r;
s.t. max2{i in 1..N}: y[i]<=1-r;
```

6.1.2 File .run

```
reset; option randseed 0;
model Packing_circle.mod;
option solver minos;
param sol;
let sol:=-10000;
for {i in 1..200}
{
 reset data x;
 reset data y;
 reset data r;
 solve;
 if (r>sol) then
        {
        let sol:=r;
        }
}
display x, y, r;
printf "Solution trouve= %f\n", sol;
```

6.1.3 Résultat sous forme de tableau :

N=2	N=3	N=4	N=5	N=6	N=7	N=8	N=9	N=10	N=11
0.29289	0.25433	0.2500	0.2071	0.20710	0.17445	0.170541	0.166	0.1482	0.142399

On remarque que toutes les valeurs se correspondent aux valeurs trouvé sur le site : http://hydra.nat.uni-magdeburg.de/packing/csq/d1.html

On sait très bien que cette méthode ne sera plus valable lorsque n est très grand en effet il fodra faire beaucoup plus d'opération et le nombre de variables est très grand et le nombre de maximum local aussi ce qui rend les calculs fastidieux.