# Rapport de rendu – défi 3 IBM

Équipe BAVARDS

6 décembre 2013

## 1 Introduction

Le problème suivant a été posé en défi de la nuit de l'info édition 2013. Initialement, il faut placer des photos d'homme et de femme dans un tableau $8 \times 8$. Cependant un certain nombre de contraintes viennent s'ajouter à ce problème, car il faut équilibrer chaque ligne de ce tableau. L'objectif est de maximiser le gain défini par le nombre de pourcentage différente pour toutes les diagonales possibles.

## 2 Programme linéaire

La solution optimale sera une matrice de décision $8 \times 8$ et chaque case représentera soit par 1 (une femme), soit 0 (un homme).

On pose les variables (de décision) suivante :

$$x_{i,j} = \begin{cases} 1 & \text{si la case } (i,j) \text{ est une femme} \\ 0 & \text{sinon} \end{cases}$$

$$y_{val} = \begin{cases} 1 & \text{si la valeur } val \text{ apparait au moins une fois} \\ 0 & \text{sinon} \end{cases}$$

$$t_{j,\ell} = \begin{cases} 1 & \text{si la colonne } j \text{ a exactement } \ell \text{ femmes} \\ 0 & \text{sinon} \end{cases}$$

$d_k$ est le pourcentage à la $k$-ième diagonale.

**Proposition 1.** *Le nombre de pourcentage différent $|V|$ est en $O(n^2)$.*

*Démonstration.* La taille des diagonales dans notre matrice peut prendre des valeurs entre 1 et $n$. Pour chaque diagonale de taille $t$, les valeurs possibles sont compris entre 1 et $t$. Donc au final, on a $1 + 2 + \ldots + n = \frac{n(n+1)}{2}$ valeurs possibles (même s'il y a des doublons). □

$$\text{maximiser } 1000 \sum_{val \in V} y_{val}$$

s.t.

$$\sum_{\ell=1}^{n} x_{\ell,j} - \sum_{\ell=1}^{n} x_{\ell,j'} = 0 \quad \forall 1 \leq j < j' \leq n$$

$$\sum_{\ell=0}^{n} t_{j,\ell} = 1 \quad \forall 1 \leq j \leq n$$

$$\sum_{j=1}^{n} t_{j,\ell} = 1 \quad \forall 0 \leq \ell \leq n$$

$$\sum_{j=1}^{n} x_{i,j} = \sum_{\ell=1}^{n} t_{i,\ell} \quad \forall 1 \leq i \leq n$$

$$d_i = \frac{\sum_{i'=0}^{n-i} x_{1+i',i+i'}}{n-i+1} \quad \forall i = 1, \ldots, n$$

$$d_{n+j-1} = \frac{\sum_{i'=0}^{n-j} x_{j+i',1+i'}}{n-j+1} \quad \forall j = 2, \ldots, n$$

$$d_{2n-1+i} = \frac{\sum_{i'=0}^{i-1} x_{1+i',i-i'}}{i} \quad \forall i = 1, \ldots, n$$

$$d_{3n-2+j} = \frac{\sum_{i'=0}^{n-j} x_{j+i',n-i'}}{n-j+1} \quad \forall j = 2, \ldots, n$$

$$\sum_{k=1 \mid d_k = val}^{4n-2} 1 \geq y_{val} \quad \forall val \in V$$

**Proposition 2.** *Le programme linéaire a $O(n^2)$ contraintes et $O(n^2)$ variables.*

*Démonstration.* La première et la dernière contrainte du programme linéaire sont au nombre de $O(n^2)$. En effet, $|V|$ est au nombre de $O(n^2)$ grâce à la Proposition 1, et de même pour la première contrainte. Toutes les autres contraintes sont au nombre $O(n)$. Donc on a au total $O(n^2)$ contraintes.

Les différentes variables sont $x, y, t, d$.

- Il y a $O(n^2)$ nombres de variables de type $x$
- Il y a $O(n^2)$ nombres de variables de type $y$
- Il y a $O(n^2)$ nombres de variables de type $t$
- Il y a $O(n)$ nombres de variables de type $d$

□

# 3 Résolution

Nous avons modélisé ce programme linéaire sur ILOG CPLEX©. Pour $n = 8$, on trouve cette affectation de photos hommes/femmes représentées par des 0/1.

L'exécution a pu être fait, cependant la résolution s'est arrêtée car la limite mémoire a été atteinte.

```
 4784295 2070746    23000,0000    58    19000,0000    23000,0000 78954237   21,05%
 4797654 2075684    21870,8333    53    19000,0000    23000,0000 79184157   21,05%
 4812000 2080802    22923,0769    71    19000,0000    23000,0000 79399069   21,05%
 4825453 2085585    22600,0000    24    19000,0000    23000,0000 79640570   21,05%
Elapsed time = 1622,24 sec. (1399750,42 ticks, tree = 22061,56 MB, solutions = 8)
Nodefile size = 21931,45 MB (12984,40 MB after compression)
 4840042 2091101    22733,3333    48    19000,0000    23000,0000 79876396   21,05%
 4854102 2096912    23000,0000    52    19000,0000    23000,0000 80119881   21,05%
 4868590 2102161    23000,0000    55    19000,0000    23000,0000 80355264   21,05%
 4883943 2107254    23000,0000    51    19000,0000    23000,0000 80589557   21,05%
 4898229 2112391    23000,0000    42    19000,0000    23000,0000 80823874   21,05%
 4913691 2117397    23000,0000    63    19000,0000    23000,0000 81061753   21,05%
 4926879 2122703    21028,5714    47    19000,0000    23000,0000 81294460   21,05%
 4942399 2127818    21750,0000    14    19000,0000    23000,0000 81534196   21,05%
 4952820 2131296    22470,0000    44    19000,0000    23000,0000 81719684   21,05%
```

```
 4966959 2136622    22550,0000    28    19000,0000    23000,0000 81950771   21,05%
Elapsed time = 1771,02 sec. (1438825,88 ticks, tree = 22598,17 MB, solutions = 8)
Nodefile size = 22468,49 MB (13312,92 MB after compression)
```

La meilleure solution entière à l'arrêt a été récupérée dont la valeur de la fonction objective est égale 19000.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |

FIGURE 1 – Solution réalisable pour notre problème

Dans la Figure 1, le gain pour notre problème est 19 000. En effet, les différents pourcentage pour les différentes colonnes sont :

```
0
0,1666666667
0,2
0,25
0,2857142857
0,3333333333
0,375
0,4
0,4285714286
0,5
0,5714285714
0,6
0,625
0,6666666667
0,7142857143
0,75
0,8
0,8333333333
1
```

# 4 Conclusion

Notre programme linéaire peut s'appliquer à une grille plus large. $n$ étant une donnée du problème. On peut imaginer à avoir une matrice plus grande. Cependant, les différentes valeurs de pourcentage seront plus nombreuses ce qui peut poser certains problèmes d'arrondies. Il faudrait explorer le fait de pouvoir stocker une liste de fractions qui correspondrait aux différentes valeurs de pourcentage de femmes dans chaque diagonale.

# Annexe

## Code source du programme linéaire

```
/*********************************************************************************
 *
 * Programme lineaire pour le defi nuit de l'info.
 *
 *********************************************************************************/

/*********************************************************************************
 *
 * DATA
 *
 *********************************************************************************/


int     n       = ...;
range   r_n  = 1..n;

int nV = ...;
range r_V = 1..nV;
float v[r_V] = ...; //ensemble des pourcentages possibles
int tmp=4*n-2;
range r_k= 1..tmp;
range z_n= 0..n-1;

dvar boolean x[r_n][r_n];
dvar boolean y[r_V];
dvar boolean t[r_n][z_n];
dvar float+ d[r_k];

/*********************************************************************************
 *
 * MODEL
 *
 *********************************************************************************/

// Objective
maximize 1000 * (sum (i in r_V) y[i]);
subject to {

   // contrainte (a)
   forall (j in r_n)
     forall(k in j..n )
        sum (l in r_n) x[l][j] - sum (l in r_n) x[l][k] == 0;

   // contrainte (b)
   forall (j in r_n)
       sum (l in z_n) t[j][l]<=1;

    // contrainte (c)
```

```
   forall (l in z_n)
       sum (j in r_n) t[j][l]<=1;

   // contrainte (d)
   forall (i in r_n)
       sum (j in r_n) x[i][j] - sum(k in z_n) (k*t[i][k]) == 0;

//somme diagonale
   forall(j in r_n)
      d[j]== (sum (l in 0 .. (n - j))  x[1+l][j+l]) /(n-j+1);


   forall(j in 2 .. n)
      d[n+j-1]== sum (l in 0 .. (n - j)) x[ j+l][ 1+l] /(n-j+1);

   forall(i in r_n)
      d[(2*n)-1 +i]== sum (l in 0 .. ( i-1)) x[ 1+l][i-l] /(i) ;


   forall(j in 2 .. n)
      d[(3*n)-2+j]== sum (l in 0 .. (n - j)) x[ j+l][ n-l] /(n-j+1) ;

   forall (val in r_V)
      sum (k in r_k) (d[k] == v[val]) >= y[val];
};



/*********************************************************************************
 *
 * SCRIPT
 *
 *********************************************************************************/


main {

   thisOplModel.generate();
   cplex.nodefileind = 2;
   cplex.tilim = 3600;
   cplex.solve();

   for (var i in thisOplModel.r_n)
   {
    for (var j in thisOplModel.r_n) {

      write("|",thisOplModel.x[i][j]);

    }
    writeln("|");
   }
}
```