

- TP 5. Le problème d'affectation. -

Le but de ce TP est de résoudre le problème d'affectation (Cf cours, étude de cas 6) de t agents à t tâches. On se donne un tableau de rationnels $\text{assign}[t][t]$ dont chaque entrée $\text{assign}[i][j]$ représente la performance p_{ij} de l'agent i à réaliser la tâche j . Le problème est d'assigner une tâche à chaque agent afin de maximiser la somme des performances des agents à réaliser leurs tâches respectives.

La modélisation de ce problème se fait ainsi : Les variables de décision sont les x_{ij} qui représentent la part de la tâche j réalisée par l'agent i . La fonction à maximiser est la somme des $p_{ij}x_{ij}$. Les contraintes sont :

- Pour chaque agent i fixé, la somme des x_{ij} est au plus 1.
- Pour chaque tâche j fixée, la somme des x_{ij} est au plus 1.

Il y a donc $n = t^2$ variables de décision et $m = 2t$ contraintes, parmi lesquelles t proviennent des agents et t autres proviennent des tâches. Les variables d'écart provenant des agents seront notées a_1, \dots, a_t et celles provenant des tâches seront notées t_1, \dots, t_t .

Le but est d'utiliser l'algorithme du simplexe en une phase afin de résoudre ce problème. Illustrons sur le cas (facile!) suivant :

performance	tâche 1	tâche 2
agent 1	3	2
agent 2	3	5

On voit clairement que la performance maximale sera obtenue en assignant l'agent 1 à la tâche 1 et l'agent 2 à la tâche 2. Mais on va faire le calcul en utilisant le simplexe. La principale difficulté est d'initialiser le dictionnaire correspondant au programme linéaire. On veut obtenir :

```

Le dictionnaire initial est :
a_1 = 1  -x_11  -x_12
a_2 = 1           -x_21  -x_22
t_1 = 1  -x_11           -x_21
t_2 = 1           -x_12           -x_22
-----
z = 0  +3x_11  +2x_12  +3x_21  +5x_22

```

Ceci se fait par l'intermédiaire d'une fonction **void** `dicoinitassign(rat assign[t][t], rat dico[m+1][n+1])` qui initialise dico comme dictionnaire initial du problème d'assignation. L'affichage des variables se fera avec une fonction **void** `affichevar(int i)` qui au lieu d'afficher simplement `x_i` affichera `x_11`, `x_12`, `x_21`, `x_22`, `a_1`, `a_2`, `t_1`, `t_2`, pour i variant de 1 à $n + m$.

La suite ne pose pas de problème, il suffit d'exécuter le simplexe.

```

La variable entrante est x_11.
La variable sortante est a_1.
x_11 = 1  -a_1  -x_12
a_2 = 1           -x_21  -x_22
t_1 = 0  +a_1  +x_12  -x_21
t_2 = 1           -x_12           -x_22
-----
z = 3  -3a_1  -x_12  +3x_21  +5x_22

```

La variable entrante est x_22.

La variable sortante est a_2.

$$\begin{array}{rcccccc} x_{11} = & 1 & -a_1 & -x_{12} & & & \\ x_{22} = & 1 & & & -x_{21} & -a_2 & \\ t_1 = & 0 & +a_1 & +x_{12} & -x_{21} & & \\ t_2 = & 0 & & -x_{12} & +x_{21} & +a_2 & \\ \hline z = & 8 & -3a_1 & -x_{12} & -2x_{21} & -5a_2 & \end{array}$$

La solution optimale est : $x_{11}=1$ $x_{12}=0$ $x_{21}=0$ $x_{22}=1$.

La valeur de la fonction objectif en cette solution est : 8.

Le nombre de pivots effectués est : 2.

Vous pourrez modifier la conclusion en :

L'assignation est 11, 22, qui réalise une performance de 8.

Le certificat d'optimalité est obtenu en affectant les poids suivants sur les agents et les tâches : $a_1=3$, $a_2=5$, $t_1=0$, $t_2=0$.

Une fois votre algorithme réalisé, testez-le sur des valeurs aléatoires de $assign[t][t]$. Quelle taille t de problèmes peut-on atteindre? Comparer avec le simplexe sur des programmes aléatoires.