Approche égalitariste de l'affectation

LU3IN034

Léa Cohen-Solal
Lysa Rudy
année 2022-2023

Contents

1 Satisfaction moyenne **2**
1.1 Temps de calcul 3

2 Optimisation maximin **5**

3 Optimisation maximin et affectation optimale **9**

4 Optimisation avec f **13**
4.1 Différence entre maximin et f 14

5 Affectation optimale et regret **17**
5.1 Différence entre g et f 18

6 Temps d'exécution **20**

1 Satisfaction moyenne

Nous cherchons à maximiser la satisfaction moyenne des n agents $(a_1, ..., a_n)$ en leur affectant un unique objet parmi les n objets $(o_1, ..., o_n)$
On pose les variables suivantes :

$$\forall i, j \in \{1, ..., n\}\ x_{ij} = \begin{cases} 1 & \text{si l'objet } o_j \text{ est affecté à l'agent } a_i \\ 0 & \text{sinon} \end{cases}$$

$$uij = \text{l'utilité de l'objet j pour l'agent i}$$

Pour modéliser ce problème on cherche à maximiser la moyenne de la somme des utilités de chaque agent pour son affectation. La satisfaction de l'affectation de l'objet o_j pour l'agent a_i est $x_{ij}.u_{ij}$ avec u_{ij}=U[i][j] où U est la matrice des satisfactions. Le problème linéaire se modélise de la manière suivante :

$$\max \text{ w} = (\sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij}u_{ij}).\frac{1}{n}$$

Sous les contraintes :
$\sum_{j=1}^{n} x_{ij} = 1\ \forall i \in \{1, ..., n\}$ Un agent se voit attribuer un unique objet.
$\sum_{i=1}^{n} x_{ij} = 1\ \forall j \in \{1, ..., n\}$ Un objet est affecté à un unique agent.

Pour résoudre l'instance donnée, on cherche à résoudre :

$$\max \text{ w} = (\sum_{i=1}^{5}\sum_{j=1}^{5} x_{ij}u_{ij}).\frac{1}{5}$$

$\sum_{j=1}^{5} x_{ij} = 1\ \forall i \in \{1, ..., 5\}$
$\sum_{i=1}^{5} x_{ij} = 1\ \forall j \in \{1, ..., 5\}$

Nous avons utilisé pour cela un programme python appelant le Solver Gurobi. Le code source de ce programme linéaire se trouve dans les fichiers *PL.py* et *moyenne1 .py*.

Nous obtenons les résultats suivants :

```
Optimal solution found (tolerance 1.00e-04)
Best objective 1.080000000000e+01, best bound 1.080000000000e+01, gap 0.0000%

    Variable            X
-------------------------
         x12            1
         x21            1
         x33            1
         x44            1
         x55            1
```

L'affectation qui maximise la satisfaction moyenne est donc :

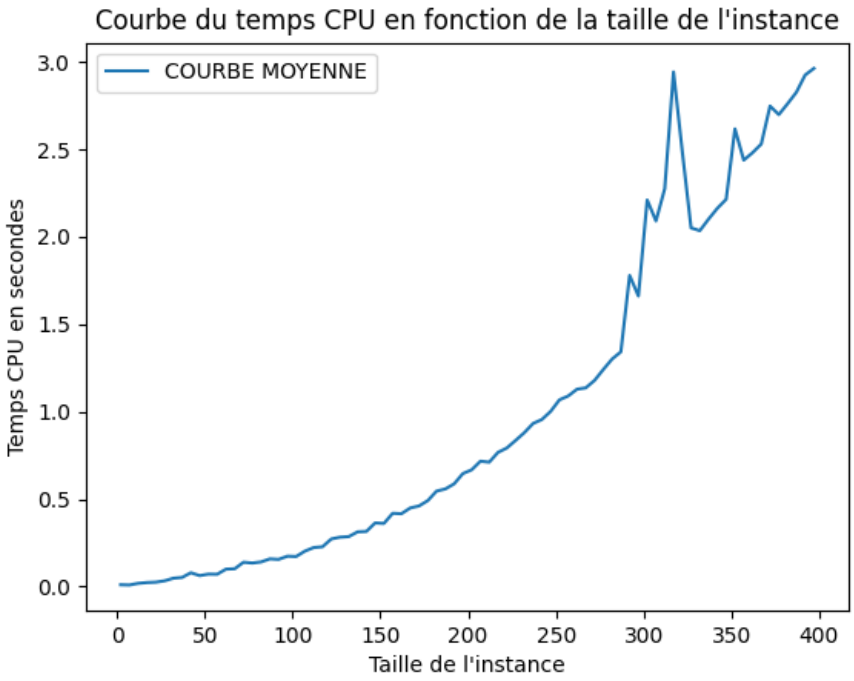
$\Pi(a_1) = o_2$, $\Pi(a_2) = o_1$, $\Pi(a_3) = o_3$, $\Pi(a_4) = o_4$, $\Pi(a_5) = o_5$,
$u(\Pi) = (20, 5, 11, 11, 7)$

Une satisfaction moyenne de 10,8.

1.1 Temps de calcul

Nous avons mesuré l'évolution du temps d'exécution en fonction de la taille de l'instance.
Nous avons fait varier la taille de l'instance de 2 à 100. Le code python se trouve dans le fichier *test.py*, fonction *courbe_moyenne*.

Nous obtenons les résultats suivants :



Nous remarquons que le temps d'exécution de cette fonction est relativement court et atteint seulement 0.2 secondes pour une instance de 100 agents et 3 secondes pour une instance de 400 agents.

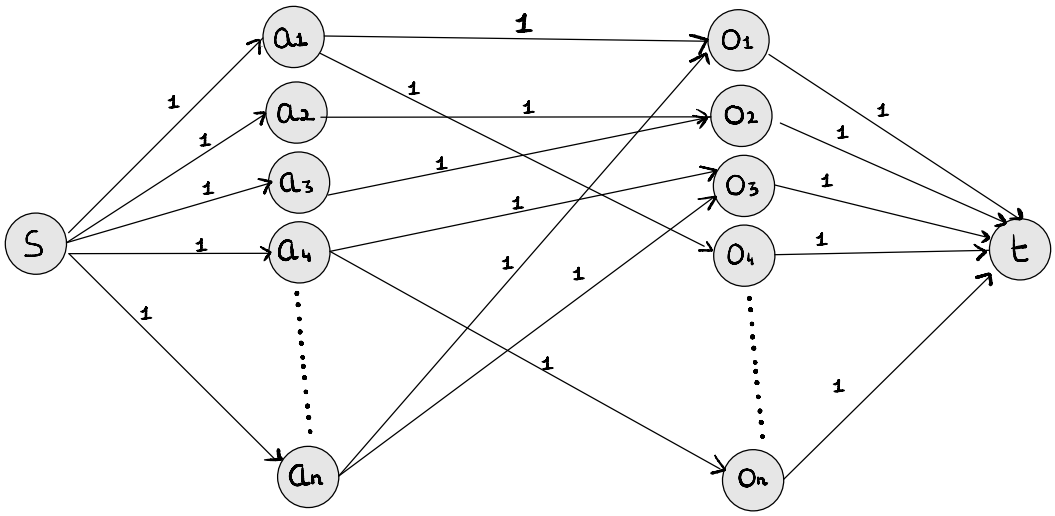
2 Optimisation maximin

On peut modéliser le problème P_λ avec un problème de flot maximum sur le graphe orienté biparti suivant :

Soit G = (S, A,s,t)

- S = $U \cup U'$ avec U = $\{a_i, i \in \{1, .., n\}\}$ et U' = $\{o_j, j \in \{1, .., n\}\}$
- A = $\{\{a_i, o_j\}, i, j \in \{1, .., n\} | u_{ij} \geq \lambda\}$
- s, l'entrée du graphe
- t, sortie du graphe (but à atteindre)

Le graphe est donc représenté de la manière suivante

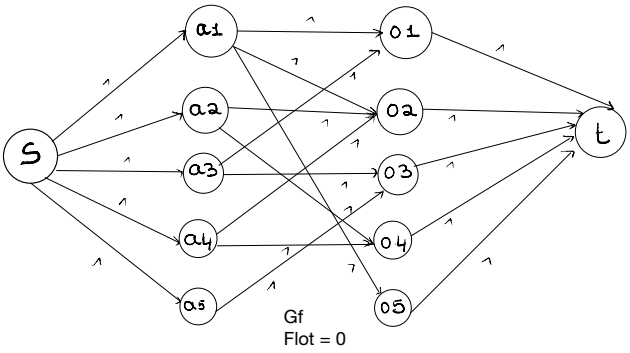


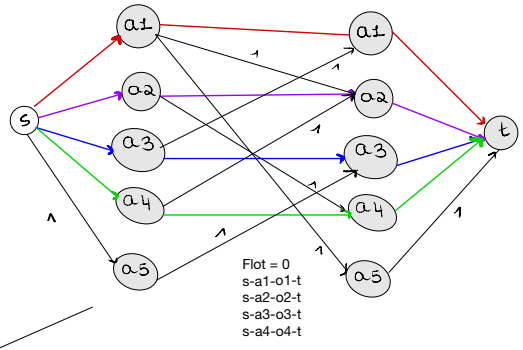
Tous les arcs ont une capacité de 1, en effet :
Les arcs $\{o_j, t\}$ doivent avoir une capacité de 1 car un objet o_j ne doit "laisser passer qu'un seul agent".
Les arcs $\{s, a_j\}$ ont une capacité de 1 car a_j ne doit s'associer qu'à un seul objet.
Nous remarquons rapidement que le flot de ce graphe ne peut pas dépasser $\sum_{i=1}^{n} f(o_i, s) \leq \sum_{i=1}^{n} c(o_i, s) = \sum_{i=1}^{n} 1 = n$ où c représente la capacité et f le flot.
Ce flot est atteint lorsque chaque o_j est affecté à un agent.
Comme tous les o_j sont affectés et que chaque agent a_i ne doit être associé qu'à un seul objet, nous avons bien les contraintes suivantes qui sont respectées :
- Un objet est affecté à un unique agent
- Un agent est associé à un unique objet
Il existe une affectation Π telle que $u_i(\Pi(a_i)) \geq \lambda, \forall i \in \{1, ..., n\}$ si et seulement si le flot maximum de ce graphe est n.

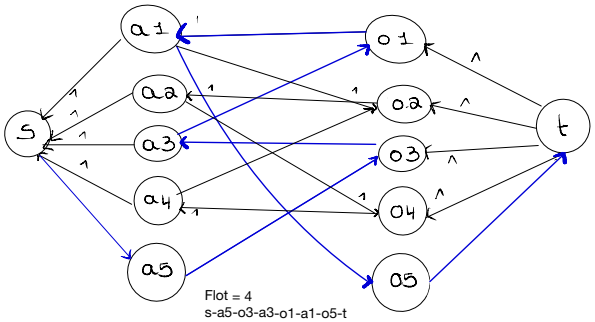
Résolvons le problème P_λ pour l'instance du sujet avec $\lambda = 8$

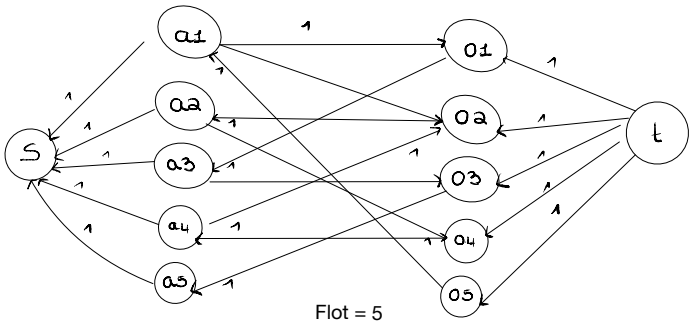
Soit G = (S,A,s,t,c)
S = $U \cup U'$ avec U = $\{a_1, a_2, a_3, a_4, a_5\}$ et U' = $\{o_1, o_2, o_3, o_4, o_5\}$
A = $\{\{s, a_1\}, \{s, a_2\}, \{s, a_3\}, \{s, a_4\}, \{s, a_5\}, \{a_1, o_1\},$
$\{a_1, o_2\}, \{a_1, o_3\}, \{a_2, o_2\}, \{a_2, o_4\}, \{a_3, o_1\}, \{a_3, o_3\},$
$\{a_4, o_2\}, \{a_4, o_2\}, \{a_5, o_3\}, \{o_1, t\}, \{o_2, t\}, \{o_3, t\}, \{o_4, t\}, \{o_5, t\}\}$

Calculons le flot maximum de ce graphe (que l'on a représenté comme un graphe résiduel)









Nous observons que le flot maximum est $5 = |\{o_1, ..., o_5\}|$.
Il existe donc bien une affectation Π telle que $\forall i \in \{1, ..., 5\}, u_i(\Pi(a_i)) \geq 8$ car, en effet, les cinq objets sont tous bien affectés à un unique agent (flot max $= 5$) et :

$$\forall i \in \{1, ..., 5\}, u_i(\Pi(a_i)) \geq 8.$$

Notons ici que les chemins du graphe ne sont pas forcement les paires de l'affectation Π .

3 Optimisation maximin et affectation optimale

Sans utiliser la programmation linéaire et en se basant uniquement sur les calculs du flot, on peut trouver une affectation optimale au sens du maximin de la façon suivante :

1. Savoir si il existe une affectation optimale :

- Soit M la matrice composée des notes des agents comme présentée dans le sujet.
  On a $M[i, j]$ = utilité de l'objet j pour l'agent i, avec

$$m_{ij} \in [0, 20]$$

- On commence par résoudre le problème P_λ avec $\lambda = 1$, donc P_1.
  Si le problème P_1 admet une solution (c'est à dire qu'il existe une affectation Π telle que $\forall i \in \{1, ..., n\}, u_i(\Pi(a_i)) \geq 1$), alors on résout P_{1+1} et ainsi de suite.
- Si à partir d'un rang k ≤ 20 il n'existe plus de solution pour P_k, on s'arrête en sachant qu'il existe une solution au problème P_{k-1} qu'on note $P_{\lambda_{max}}$.

2. Trouver cette affectation :

Après avoir montré qu'il existe un certain λ_{max} et donc une affectation Π telle que $\forall i \in \{1, ..., n\}, u_i(\Pi(a_i)) \geq \lambda_{max}$, on retrace le graphe G = (S, A,s,t)

- S = $U \cup U'$ avec U = $\{a_i, i \in \{1, .., n\}\}$ et U' = $\{o_j, j \in \{1, .., n\}\}$
- A = $\{\{a_i, o_j\}, i, j \in \{1, .., n\} | u_{ij} \geq \lambda_{max}\}$
- s, l'entrée du graphe
- t, sortie du graphe (but à atteindre)

a. On commence par :

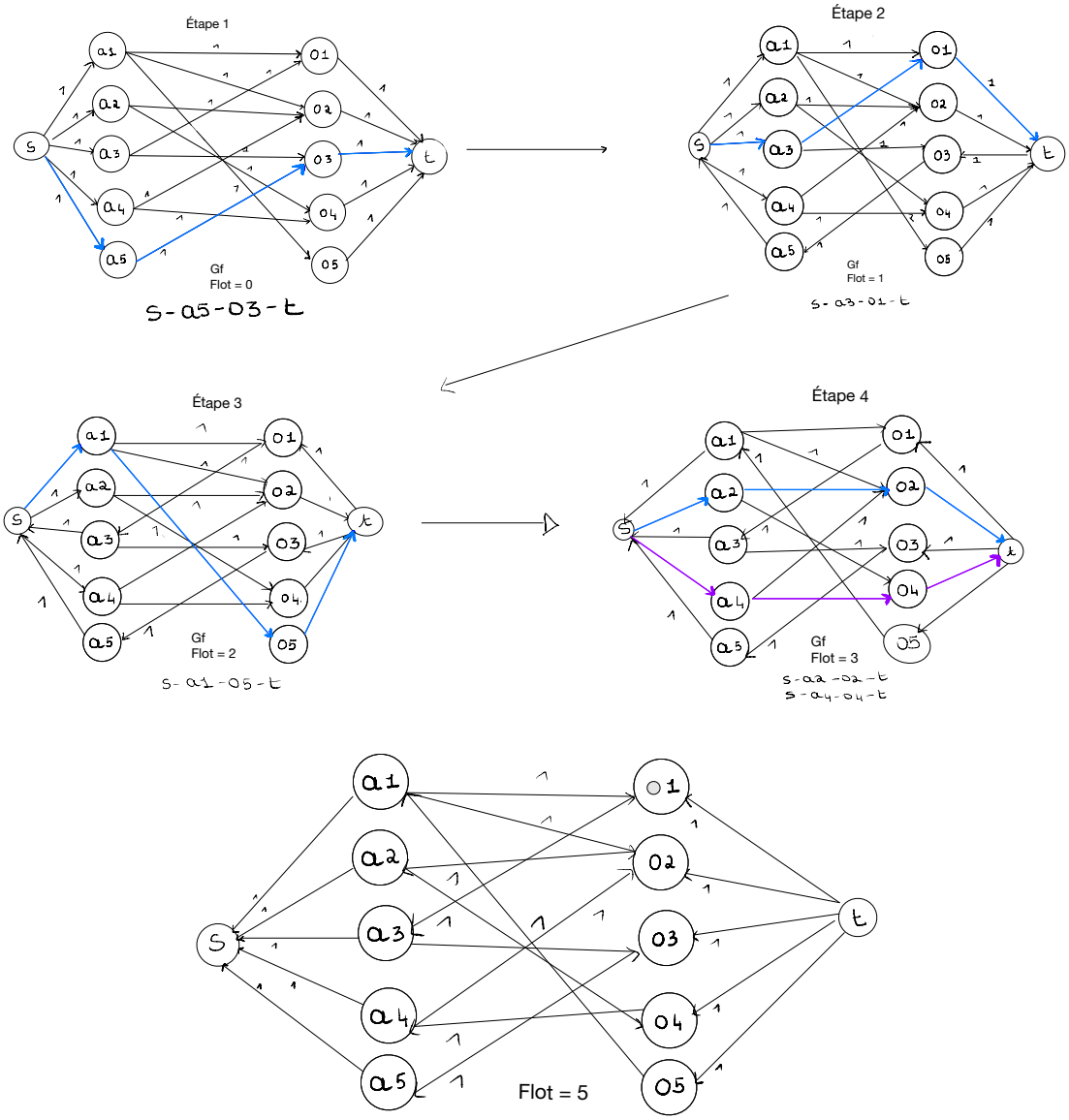
- le chemin $s - a_i - o_j - t$ tel qu'il n'existe qu'un seul chemin de s a t passant par a_i. En effet cela veut dire que pour l' agent a_i il n'existe qu'un seul objet o_j tel que $M[i, j] \geq \lambda_{max}$. (Ce a_i n'est pas unique)

- le chemin $s - a_i - o_j - t$ tel qu'il n'existe qu'un seul chemin de s à t passant par o_j. En effet cela veut dire que l'objet o_j, ne peut être affecté qu'à l'agent a_i. (ce o_j n'est pas unique)

b. Après avoir tracé ces chemins, nous allons avoir des arêtes $\{a_i, o_j\}$ saturées par les chemins précédents. Donc on reprend les nouveaux agents $a_{i'}$ et objets $o_{j'}$ qui n'ont qu'une seule possibilité pour arriver à t et ainsi de suite jusqu'à avoir un flot maximum de n.

3. Trouver cette affectation pour l'instance du sujet

D'après Q2 il existe une solution pour le problème P_8. On peut directement dire qu'il n'existe pas de solution pour le problème P_9 car $max\{M[5, j], \forall j \in \{1, ..., 5\}\} = 8 \leq 9$.
Trouvons cette affectation.



Dans la première étape, on choisit le chemin $s - a_5 - o_3 - t$ car l'agent a_5 ne peut passer que par o_3.
Dans la deuxième étape on choisit le chemin $s - a_3 - o_1 - t$ car comme a_5 est déjà passé par o_3, a_3 ne peut passer que par o_1.
Dans la troisième étape on choisit le chemin $s - a_1 - o_5 - t$ car c'est le seul chemin passant par o_5.

Dans la dernière étape on choisit les chemins $s - a_2 - o_2 - t$ et $s - a_4 - o_4 - t$ (on aurait pu aussi choisir $s - a_2 - o_4 - t$ et $s - a_4 - o_2 - t$). On a donc l'affectation suivante :
$\Pi(a_1) = o_5$, $\Pi(a_2) = o_2$, $\Pi(a_3) = o_1$, $\Pi(a_4) = o_4$, $\Pi(a_5) = o_3$,
$u(\Pi) = (8, 12, 8, 11, 8)$
Notons ici que l'on aurait pu faire les 2 étapes en même temps (savoir si il existe une affectation optimale et la trouver) en traçant nos chemins de manière stratégique dans le graphe.

4 Optimisation avec f

Pour approcher le problème d'une manière différente, on pose

$$f(\Pi) = min\{u_i(\Pi(a_i))\} + \epsilon * \sum_i u_i(\Pi(a_i))$$

Pour maximiser cette fonction on prends les variables suivantes :

- $\forall i, j \in \{1, ..., n\}$, $x_{ij} = \begin{cases} 1 & \text{si l'objet } o_j \text{ est affecté à l'agent } a_i \\ 0 & \text{sinon} \end{cases}$
- l_i : variable temporaire pour accéder à la matrice.
- u_{ij} = l'utilité de l'objet j pour l'agent i

Le problème linéaire se modélise de la manière suivante :

$$\max \text{ w} = mini + \epsilon * \sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij} u_{ij}$$

Sous les contraintes :

- $\sum_{j=1}^{n} x_{ij} = 1 \ \forall i \in \{1, ..., n\}$: Un agent se voit attribuer un unique objet.
- $\sum_{i=1}^{n} x_{ij} = 1 \ \forall j \in \{1, ..., n\}$: Un objet est affecté à un unique agent.
- $l_i = u_{ij}$ si $x_{ij} = 1$, $\forall i, j \in \{1, ..., n\}$
- mini = $\min\{l_i \ , \ \forall i \in \{1, ..., n\}\}$: mini prend la valeur de l'utilité de l'objet pour l'agent le moins satisfait.

En effet étant donné que chaque agent est plus ou moins satisfait de son objet (la satisfaction étant représentée par la matrice utilité), il existe un agent qui est le moins satisfait de tous. On va chercher ici à maximiser la satisfaction de l'agent le moins satisfait. De plus la valeur de epsilon donnera une importance ou non à la satisfaction totale pour le choix de l'affectation.

Le problème revient ici à maximiser la satisfaction de l'agent le moins satisfait.

Le code de ce programme linéaire se trouve dans le fichier $f2.py$.
Sur l'instance donnée nous obtenons les résultats ci-dessous :

```
    Variable             X
-------------------------
          l1             8
         x15             1
          l2            12
         x22             1
          l3             8
         x31             1
          l4            11
         x44             1
          l5             8
         x53             1
        mini             8
Obj:  8.00000000000001
```

$\Pi(a_1) = o_5$, $\Pi(a_2) = o_2$, $\Pi(a_3) = o_1$, $\Pi(a_4) = o_4$, $\Pi(a_5) = o_3$
$u(\Pi) = (8, 12, 8, 11, 8)$

4.1 Différence entre maximin et f

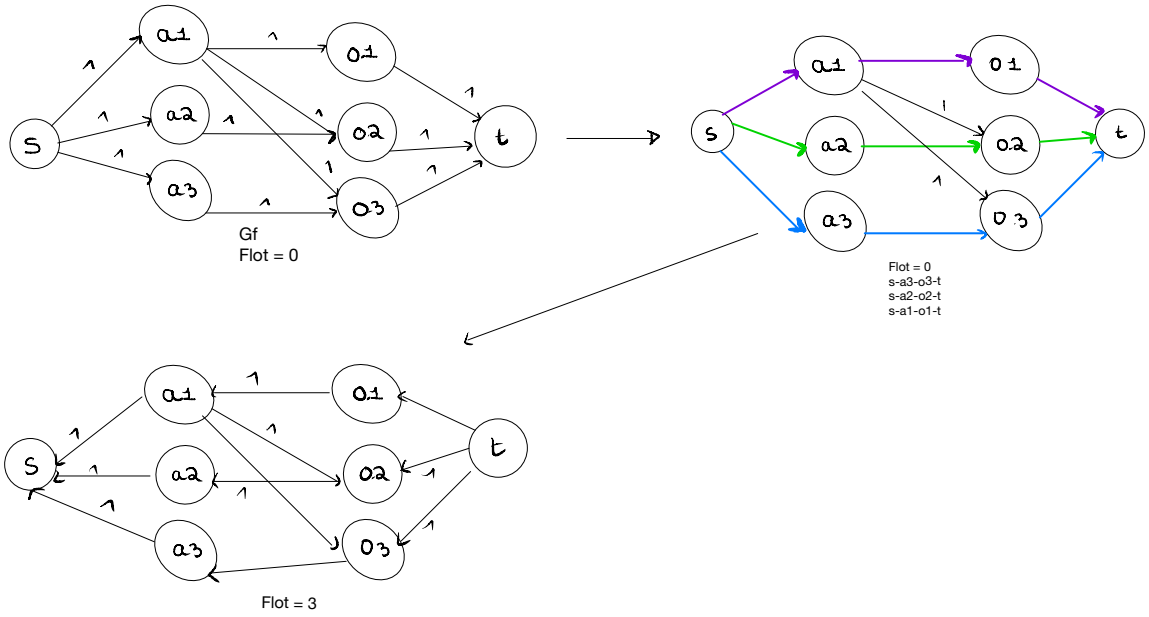
Trouvons une instance qui admet une affectation optimale au sens de maximin mais pas au sens de f

Soit l'instance suivante :

$$A = \begin{pmatrix} 1 & 3 & 1000 \\ 0 & 20 & 0 \\ 0 & 0 & 3 \end{pmatrix}$$

composée donc de 3 agents et 3 objets.
Trouvons une affectation optimale au sens de maximin.

Resolvons P_1 :



Il s'agit de la seule affectation possible car :
a_3 est obligé de prendre o_3, a_2 est obligé de prendre o_2, donc a_1 se retrouve avec o_1.
On voit rapidement que $\lambda_{max} = 1$ car si on essayait avec $\lambda = 3$, on ne pourrait affecter l'objet o_1 à personne. On a l'affectation Π telle que $\Pi(a_1) = o_1$, $\Pi(a_2) = o_2$, $\Pi(a_3) = o_3$.
En revanche une affectation optimale au sens de f avec epsilon = 0.05 nous donne :

```
Optimal solution found (tolerance 1.00e-04)
Best objective 5.100000000000e+00, best bound 5.100000000000e+00, gap 0.0000%

    Variable           X
-------------------------
          l1        1000
         x13           1
          l2          20
         x22           1
         x31           1
Obj:  5.1
```

On a l'affectation Π' telle que $\Pi'(a_1) = o_3$, $\Pi'(a_2) = o_2$, $\Pi'(a_3) = o_1 \neq \Pi$.
En effet le choix de l'affectation optimale selon f prend aussi en compte la satisfaction totale .
Si l'affectation d'un objet o_j à un agent a_i permet de faire augmenter considérablement la satisfaction totale (dépend de epsilon) alors on privilégiera cette affectation même si les autres agents ne sont pas du tout satisfaits. C'est le cas ici de o_3 pour a_1.
Trouver une affectation optimale au sens de maximin revient à trouver une affectation optimale selon f avec epsilon = 0. L'affectation optimale selon f permet donc de donner une affectation égalitaire en donnant plus ou moins d'importance à la satisfaction totale. Plus on augmente epsilon, plus on favorisera une affectation qui maximise la satisfaction totale. Plus epsilon tend vers 0, plus on favorisera une affectation complètement égalitaire qui ne prend presque pas en compte (de l'ordre de epsilon) la satisfaction totale. Il est donc plus intéressant de choisir une affectation optimale selon f car elle permet de donner une importance ou non (par l'intermédiaire de la variable epsilon) à la satisfaction totale.

5 Affectation optimale et regret

On veut minimiser la fonction

$$g(\Pi) = max\{regret[i] \forall i \in \{1, ..., n\}\}$$

Pour minimiser cette fonction on prends les variables suivantes :

- $\forall i, j \in \{1, ..., n\}$, $x_{ij} = \begin{cases} 1 & \text{si l'objet } o_j \text{ est affecté à l'agent } a_i \\ 0 & \text{sinon} \end{cases}$
- u_{ij} = l'utilité de l'objet j pour l'agent i
- $\forall i \in \{1, ..., n\}$, $regret[i]$ = regret de l'agent i

Le problème linéaire se modélise de la manière suivante :

$$\min \text{ w} = maxi$$

Sous les contraintes :

- $\sum_{j=1}^{n} x_{ij} = 1 \ \forall i \in \{1, ..., n\}$: Un agent se voit attribuer un unique objet.
- $\sum_{i=1}^{n} x_{ij} = 1 \ \forall j \in \{1, ..., n\}$: Un objet est affecté à un unique agent.
- Regret[i] = $max(u_{ij}, \forall j \in \{1, ..., n\}) - u_i(o_j)$
- $maxi = max(regret[i], \forall i \in \{1, ..., n\})$

Le code de cette fonction se trouve dans le fichier *g3.py* et les fonctions auxiliaires dans le fichier *PL.py*

5.1 Différence entre g et f

Soit l'instance suivante :

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 5 \\ 3 & 4 & 10000 & 5 \\ 4 & 5 & 6 & 7 \end{pmatrix}$$

- Une affectation optimale de A au sens de g est :

```
Best objective 2.999999999993e+00, best bound 2.999999999993e+00, gap 0.0000%

    Variable           X
-------------------------
         x14           1
          r2           2
         x22           1
         x33           1
          r4           3
         x41           1
        maxi           3
Obj:  2.999999999992724
```

  $\Pi(a_1) = o_4$, $\Pi(a_2) = o_2$, $\Pi(a_3) = o_3$, $\Pi(a_4) = o_1$, $u(\Pi) = (4, 3, 10000, 4)$

  avec une satisfaction moyenne de 2502.

- Une affectation optimale de A au sens de f avec epsilon = 10^{-16} est
  $\Pi(a_1) = o_4$, $\Pi(a_2) = o_3$, $\Pi(a_3) = o_2$, $\Pi(a_4) = o_1$
  $u(\Pi) = (4, 4, 4, 4)$, avec une satisfaction moyenne de 4 .

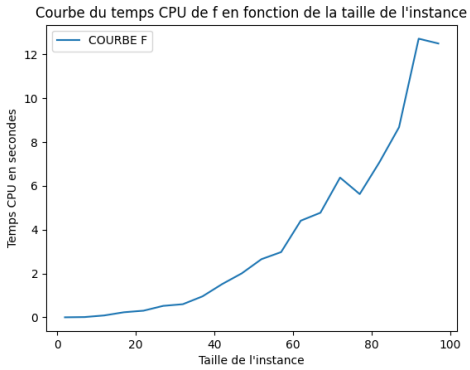
```
Best objective 4.000000000000e+00, best bound 4.000000000000e+00, gap 0.0000%

    Variable           X
-------------------------
          l1           4
         x14           1
          l2           4
         x23           1
          l3           4
         x32           1
          l4           4
         x41           1
        mini           4
Obj:  4.000000000000002
```

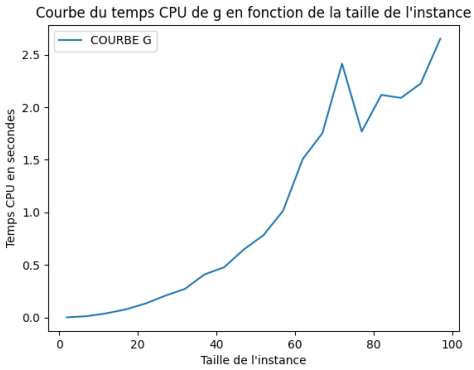
Elle est donc différente qu'au sens de g. En effet regret$[a_3] \geq 9995$, donc comme g essaye de minimiser le plus grand regret, qui sera ici celui de a_3 largement plus grand que le regret des autres agents, il va naturellement affecter o_3 à a_3. En revanche f ne va pas affecter o_3 à a_3 car, epsilon étant très petit, f va choisir une affectation très égalitaire, et va donc éviter de privilégier un agent à un autre.
Les méthodes f et g permettent d'éviter de décevoir les agents d'une manière ou d'une autre et se valent à première vue.
Ici, l'affectation selon g permet d'avoir une plus grande satisfaction moyenne que celle de f car epsilon est très petit.

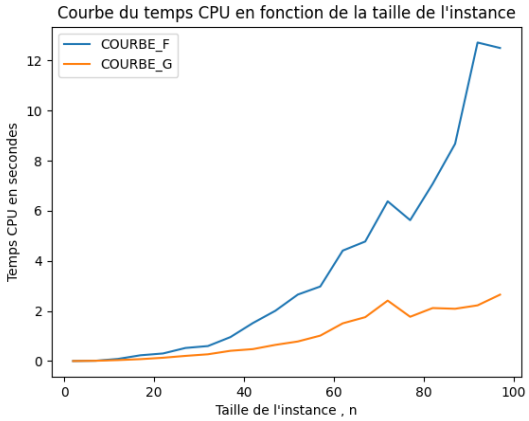
6 Temps d'exécution

Nous avons mesuré le temps de calcul d'une affectation optimale au sens de f et au sens de g.
Pour cela nous avons crée une fonction *alloue_matrice*(n) qui crée une matrice de taille $n * n$ avec des entiers aléatoires dans [0,20].

Le code de cette fonction et du calcul des temps d'exécutions se trouvent dans les fichiers *PL.py* (alloue_matrice) et *test.py*
Voici ci dessous l'évolution du temps CPU pour le calcul d'une affectation optimale selon f et selon g :







Le choix d'une affectation optimale prend environ 13 secondes au sens de f, et 2.5 secondes au sens de g pour une instance de 100 agents. Le dernier graphique montre bien une différence de temps entre f et g. Cependant une affectation au sens de f pour une instance de 100 agents ne prendra pas toujours 13 secondes. En effet il est possible de "bien tomber" des le début et de trouver rapidement une bonne affectation respectant toutes les contraintes et donc d'avoir un temps d'exécution plus court. Il y a une petite variable "chance" qui permet d'avoir un temps d'exécution plus ou moins long.