LU3IN034:

Approche égalitariste de l’affectation

Zein SAKKOUR

Philemon WEHBE

Question 1

On cherche à maximiser la satisfaction moyenne des agents, donc on cherchera d'abord à maximiser la satisfaction totale avant de diviser cette dernière par le nombre d'agents.

Soit $U = \begin{pmatrix} 12 & 20 & 6 & 5 & 8 \\ 5 & 12 & 6 & 8 & 5 \\ 8 & 5 & 11 & 5 & 6 \\ 6 & 8 & 6 & 11 & 5 \\ 5 & 6 & 8 & 7 & 7 \end{pmatrix}$. On utilisera cette matrice dans des exemples le long de ce rapport.

Posons $\forall i, j \in \{1, ..., n\},\ x_{i,j} = \begin{cases} 1 \text{ si } o_j \text{ est affecté à } a_i \\ 0 \text{ sinon} \end{cases}$.

On pose alors le PLNE (1) suivant:

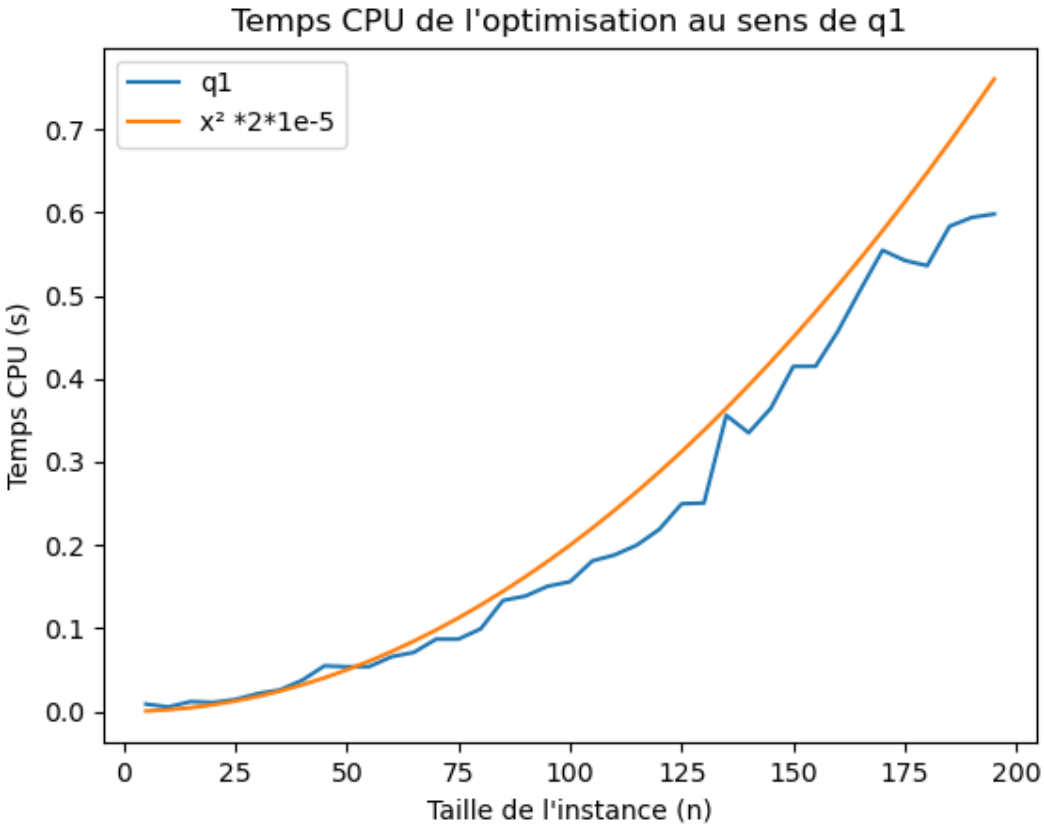
$\max \sum_{i,j} x_{i,j} U_{ij}$ s.c.

$\forall i \in \{1, \ldots, n\},\ \sum_j x_{i,j} = 1 \quad \text{(un élement par ligne)}$
$\forall j \in \{1, \ldots, n\},\ \sum_i x_{i,j} = 1 \quad \text{(un élement par colonne)}$
$x_{i,j} \in \{0, 1\}$

avec U_{ij} les coefficients de la matrice U.

N.B. On peut aussi remplacer les égalités de ces contraintes par des inégalités, le résultat sera le même.

Après avoir développé un code python (fourni) faisant appel au solveur Gurobi, la résolution de ce problème donne bien une satisfaction totale de 54, correspondant à une moyenne de satisfaction de 10.8. L'affectation obtenue est $\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 1 & 3 & 4 & 5 \end{pmatrix}$, comme dans l'énoncé.



Pour modéliser l'évolution du temps d'optimisation de cette méthode, on teste la fonction `q1` sur des instances de taille variant de 5 à 200, par pas de 5. Pour une même taille, on génère 10 matrices de notes aléatoires et on calcule leur affectation optimale respective selon le PLNE (1) puis on divise le temps CPU total par 10 afin d'obtenir une moyenne.

On remarque que l'évolution du temps de calcul en fonction de la taille de l'instance est quadratique : $O(n^2)$

Question 2

On modélise P_λ comme un problème de recherche de flot maximum dans un graphe.

On se place dans le graphe $G = (V, E, c, s, t)$ où
$V = \{a_1, \dots, a_n, o_1, \dots, o_n\}$ et
$E = \{(a_i, o_j) \ \ ssi \ \ u_i(o_j) \geq \lambda\} \bigcup \{(s, a_i)\} \bigcup \{(o_i, t)\} \quad \forall i, j \in \{1, \dots, n\}$.

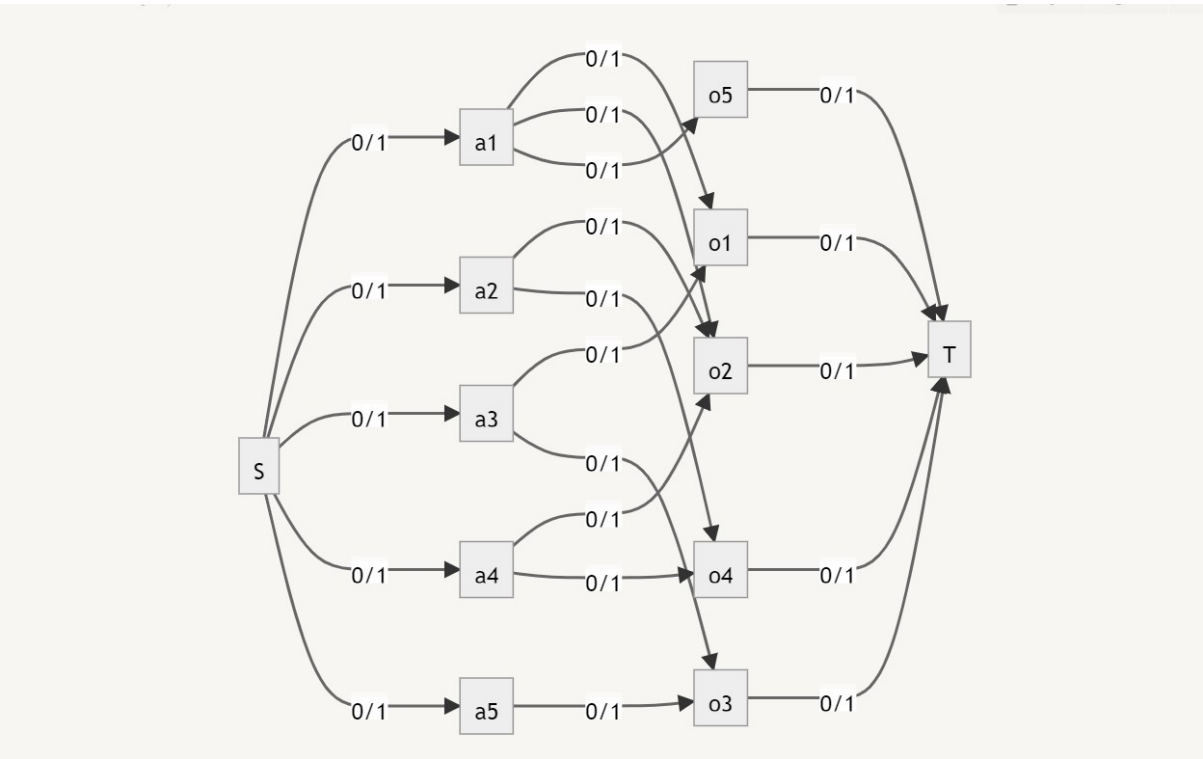
En effet, les arcs de G sont uniquement ceux dont l'utilité de l'objet o_j pour l'agent a_i est supérieure ou égale à λ. Tous ces arcs sont de capacité 1. Ainsi, on est sûr que chaque agent/objet ne pourra intervenir plus d'une fois, et chaque objet peut être attribué à au plus un agent.

On applique l'algorithme de Ford-Fulkerson sur le graphe, où on initialise tous les flots à 0, et on cherche un flot maximum. Si au bout de l'algorithme, la valeur du flot maximum est de n, le problème P_λ admet une affectation solution. Cette affectation peut être tirée en observant les arcs empruntés lors du choix des chemins augmentants.

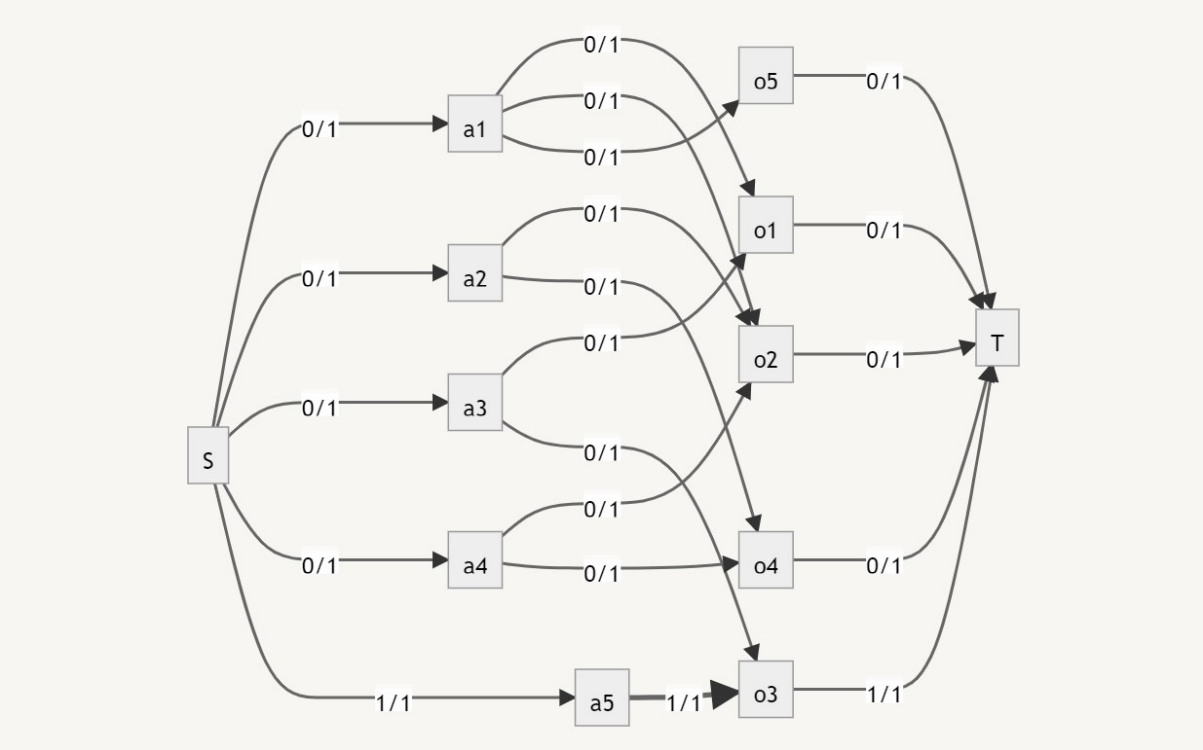
Par contre, si le flot maximum n'est pas de valeur n, alors P_λ n'a pas de solution car il existe au moins un objet non affecté.

Pour $\lambda = 8$, le graphe ci-dessous représente le réseau choisi en utilisant la matrice U de la question 1).

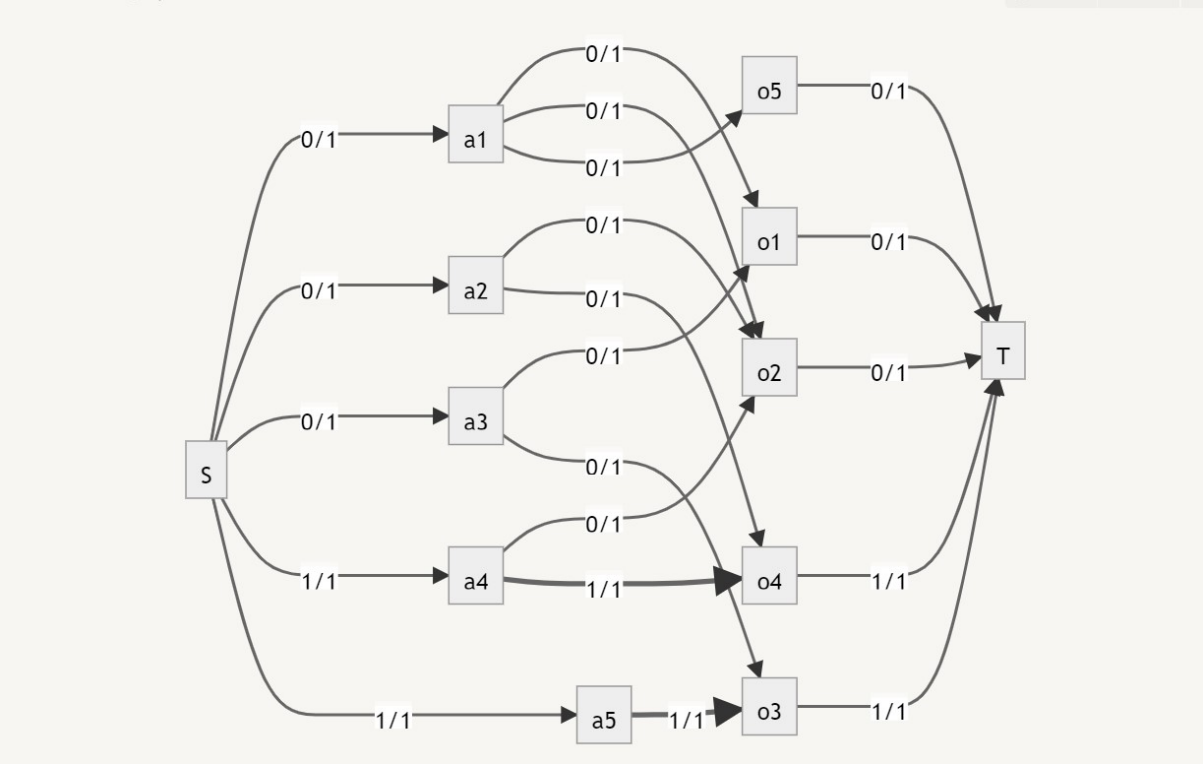
Si le flot maximum est de valeur $n = 5$ dans ce cas, il existe donc une permutation σ de $\{1, 2, 3, 4, 5\}$ telle que $\pi(a_i) = o_{\sigma(i)}$ et $u_i(\pi(a_i)) \geq \lambda$.



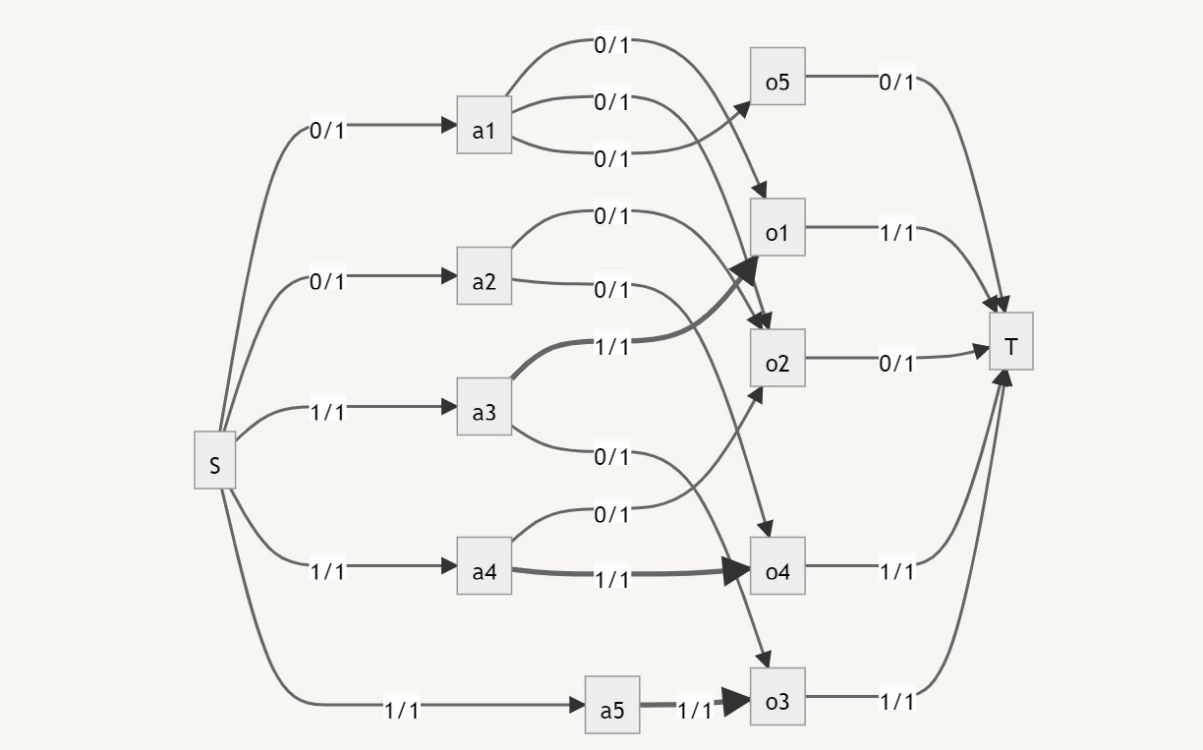
$s -> a_5 -> o_3 -> t$ est un chemin augmentant : on augmente le flot de 1.



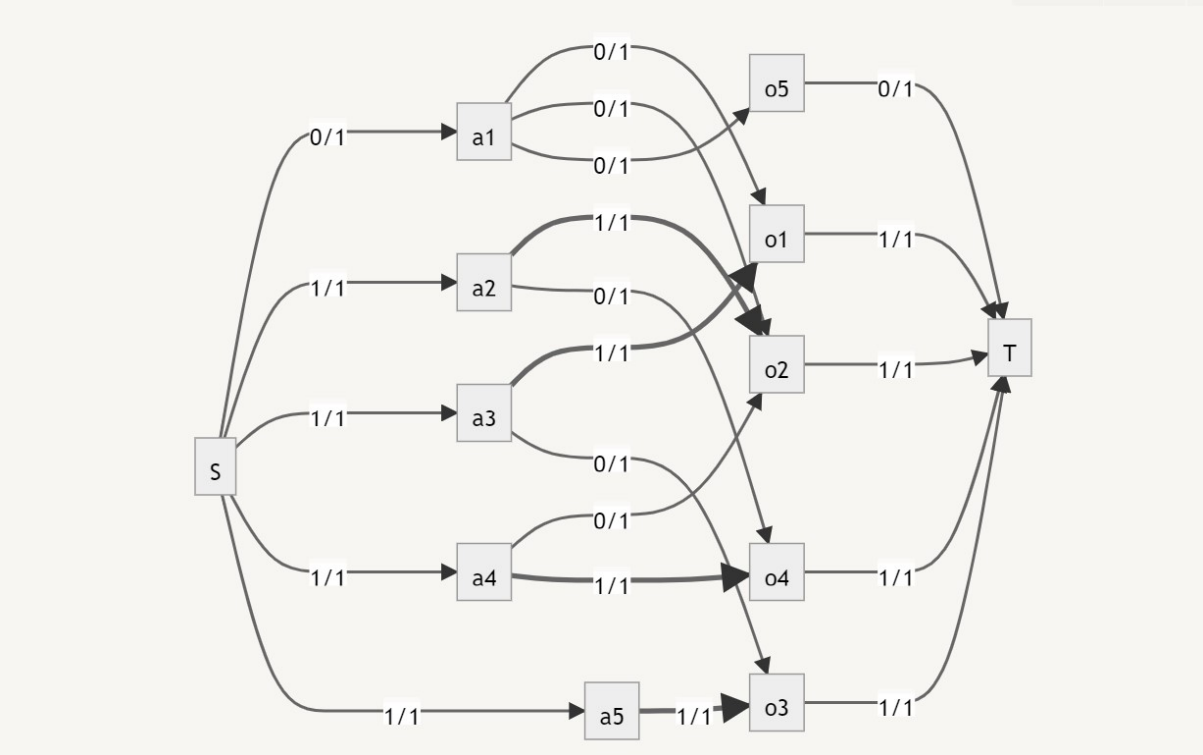
$s->a_4->o_4->t$ est un chemin augmentant : on augmente le flot de 1.



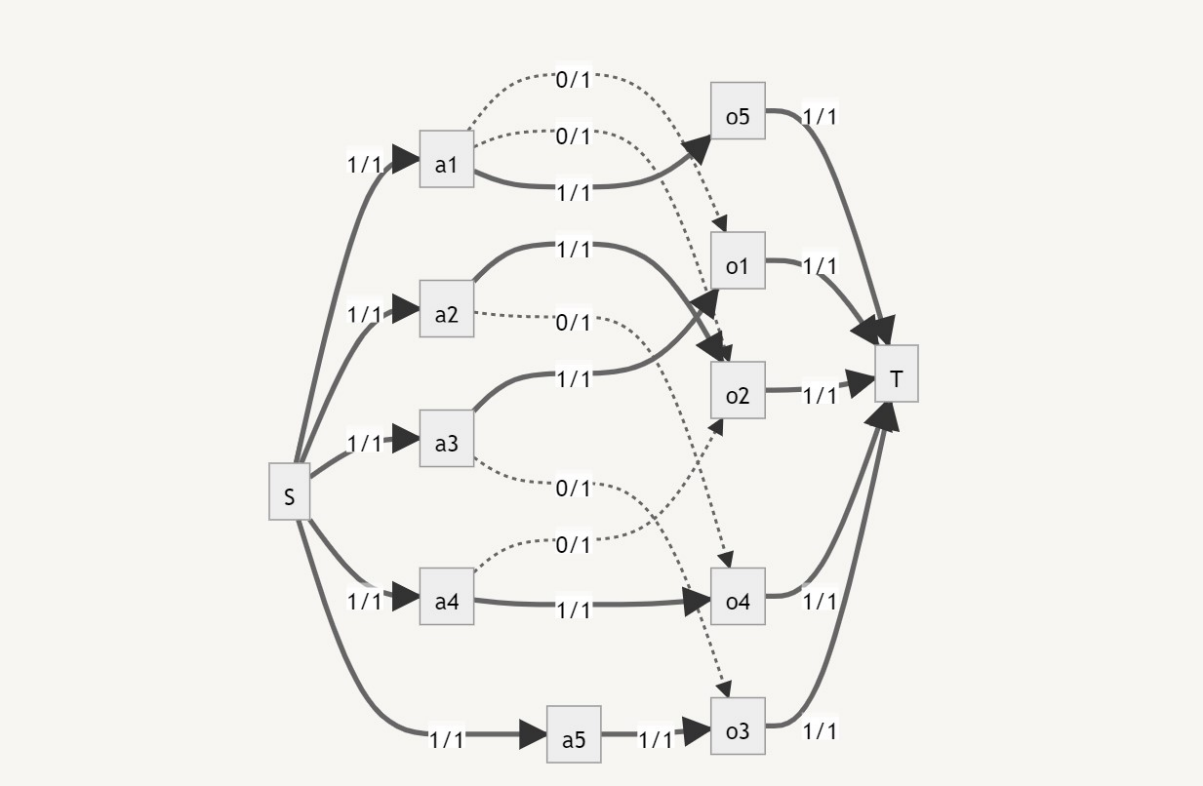
$s->a_3->o_1->t$ est un chemin augmentant : on augmente le flot de 1.



$s->a_2->o_2->t$ est un chemin augmentant : on augmente le flot de 1.



$s->a_1->o_5->t$ est un chemin augmentant : on augmente le flot de 1.



On obtient un flot maximum F de valeur $|F| = n = 5$ qui convient (en coupant le graphe entre les agents et les objets). Donc il existe une affectation vérifiant $u_i(\pi(a_i)) \geq \lambda$ pour tout i. On voit que l'affectation $\pi(a_1) = o_5,\ \pi(a_2) = o_2,\ \pi(a_3) = o_1,\ \pi(a_4) = o_4,\ \pi(a_5) = o_3$ vérifie bien P_λ.

Question 3

On remarque que les problèmes P_λ suivent une relation d'ordre. D'abord, on pose $I_\lambda = [\![\min_{i,j\in\{1,\dots n\}}\{u_{ij}\},\ \min_{i\in\{1,\dots,n\}}\{\max_{j\in\{1,\dots,n\}}\{u_{i,j}\}\}]\!]$ (qui est bien ordonné) où n est la taille de la matrice U (un P_λ ne peut pas avoir de solution si une ligne ne contient pas d'utilité supérieure à λ).

Pour trouver le plus grand λ vérifiant $u_i(\pi(a_i)) \geq \lambda$ on applique un algorithme de recherche dichotomique sur I_λ avec le test de la question 2 (on peut commencer par tester la borne supérieure de l'intervalle pour sauver quelques itérations, puis appliquer l'algorithme classique en cas d'échec).

Lorsqu'on obtient le plus grand λ vérifiant $P_{\lambda_{max}}$, on résout ce dernier par la méthode de la question 2). En observant les arcs traversés par un flot de valeur 1, on peut extraire une affectation maximin optimale (comme le flot maximum n'est pas unique, plusieurs affectations pourraient convenir).

La question 2) montre que P_8 admet une solution, donc $\lambda_{max} \geq 8$. On remarque ensuite que P_9 n'a pas de solution (le sommet a_5 n'a pas de successeur dans le graphe posé pour P_9), on peut en déduire que $\lambda_{max} = 8$. Une affectation maximin optimale est

$\pi(a_1) = o_5,\ \pi(a_2) = o_2,\ \pi(a_3) = o_1,\ \pi(a_4) = o_4,\ \pi(a_5) = o_3,$

qui vérifie bien $u_i(\pi(a_i)) = u_i(o_{\sigma(i)}) \geq \lambda$, avec $\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 2 & 1 & 4 & 3 \end{pmatrix}$ la permutation associée à π.

En revanche, la satisfaction totale n'est pas forcément optimale, comme on cherche à équilibrer l'affectation. Pour $\lambda = 8$, l'affectation trouvée à la main donne une satisfaction totale de 47 (moyenne de 9.4), mais elle peut valoir 40 (moyenne de 8) en posant $\pi'(a_1) = o_5,\ \pi'(a_2) = o_4,\ \pi'(a_3) = o_1,\ \pi'(a_4) = o_2,\ \pi'(a_5) = o_3$ qui reste maximin optimale et que l'on pouvait obtenir en choisissant de différents chemins augmentants. En effet, les 2 moyennes de satisfaction sont inférieures à celle de la question 1).

Question 4

On cherche à maximiser $f(\pi) = \min_i\{u_i(\pi(a_i))\} + \varepsilon \sum_i u_i(\pi(a_i))$.

Pour cela, on pose le PLNE (2) suivant;

$\max\ m + \varepsilon \sum_{i,j} x_{i,j} U_{ij}$ s.c

$\forall i \in \{1, \dots, n\},\ \sum_j x_{i,j} = 1$
$\forall j \in \{1, \dots, n\},\ \sum_i x_{i,j} = 1$
$\forall i \in \{1, \dots, n\},\ t \leq$
$\sum_j x_{i,j} U_{ij} \quad (\text{contrainte pour que t soit un minimum})$
$x_{i,j} \in \{0, 1\},\ t \in \mathbb{N}$

En effet, on a $u_i(\pi(a_i)) = \sum_j x_{i,j} U_{ij}$ et comme $t \leq \min\{u_i(\pi(a_i))\}$ pour être un minimum, étant donné qu'il n'y a pas de contrainte "min" sur Gurobi (et en PL), on pose la contrainte $t <= \sum_j x_{i,j} U_{ij}$ pour tous les éléments de U.

Après implémentation en python, encore grâce à Gurobi, en résolvant ce problème sur la matrice U de la question 1), on observe:

Pour $\varepsilon = 0.01$, l'affectation $\pi'(a_1) = o_5,\ \pi'(a_2) = o_4,\ \pi'(a_3) = o_1,\ \pi'(a_4) = o_2,\ \pi'(a_5) = o_3$ est maximin optimale (comme on l'a vu à la fin de la question 3) mais n'est pas optimale au sens de f, comme $f(\pi') = 8.40$ alors que la fonction objectif prend une valeur maximale de $8.47 = f(\pi)$ pour
$\pi(a_1) = o_5,\ \pi(a_2) = o_2,\ \pi(a_3) = o_1,\ \pi(a_4) = o_4,\ \pi(a_5) = o_3.$

Pour $\varepsilon = \frac{1}{2}$, l'affectation π est maximin optimale (question 3) mais n'est pas maximale au sens de f. En effet, $f(\pi) = 31.5$ alors que la fonction objectif est maximale pour $f(\Pi) = 33.5$ avec $\Pi(a_1) = o_1,\ \Pi(a_2) = o_2,\ \Pi(a_3) = o_3,\ \Pi(a_4) = o_4,\ \Pi(a_5) = o_5$.

On cherche donc à maximiser f plutôt que de passer par le maximin car pour ε assez petit (par exemple $\varepsilon \leq \frac{1}{20 \times n}$ pour U une matrice de taille n contenant des notes variant de 0 à 20), $\varepsilon \sum_i u_i(\pi(a_i)) \leq 1$ et $\max f$ donnera d'une part un λ_{max} maximin-optimal (partie entière de la fonction objectif après optimisation), ainsi que la satisfaction totale maximale associée à ce λ_{max} (en divisant la partie décimale du résultat par ε).

Cette méthode permet donc d'obtenir l'affectation maximin optimale donnant une satisfaction totale maximale, donc moyenne maximale, par rapport aux affectations maximin optimales possibles pour ce même λ_{max}.

Question 5

On cherche maintenant à minimiser $g(\pi) = \max_i(\ \max_j\{u_j(o_j)\} - u_i(\pi(a_i))\ )$

On pose donc le PLNE (3):

$\min\ r$ s.c

$\forall i \in \{1, \dots, n\},\ \sum_j x_{i,j} = 1$
$\forall j \in \{1, \dots, n\},\ \sum_i x_{i,j} = 1$
$\forall i, j \in \{1, \dots, n\},\ r \geq U_{ij} - \sum_k U_{ik} x_{i,k}$ $\quad (\text{contrainte pour que r soit un maximum})$
$x_{i,j} \in \{0, 1\},\ r \in \mathbb{N}$

En effet, on a $u(\pi(a_i)) = \sum_k U_{ik} x_{i,k}$ et comme $r \geq \max_j\{U_{ij}\} - u_i(\pi(a_i))$ pour être un maximum, étant donné qu'il n'y a pas de contrainte "max" sur Gurobi (et en PL), on pose la contrainte $r \geq U_{ij} - \sum_k U_{ik} x_{i,k}$ pour tous les éléments de U.

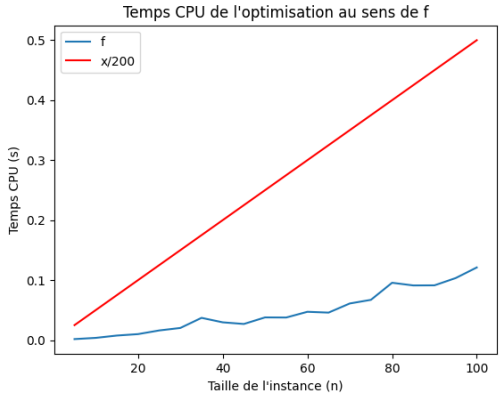
En appliquant l'implémentation en python sur la matrice U, on obtient une affectation $\pi_g(a_1) = o_2,\ \pi_g(a_2) = o_4,\ \pi_g(a_3) = o_5,\ \pi_g(a_4) = o_1,\ \pi_g(a_5) = o_3$ optimale au sens de g, de regret maximal 5, mais pas optimale au sens de f. En effet, $f(\pi_g) = 6.48 \leq 8.47$ pour ε qui vaut 0.01.

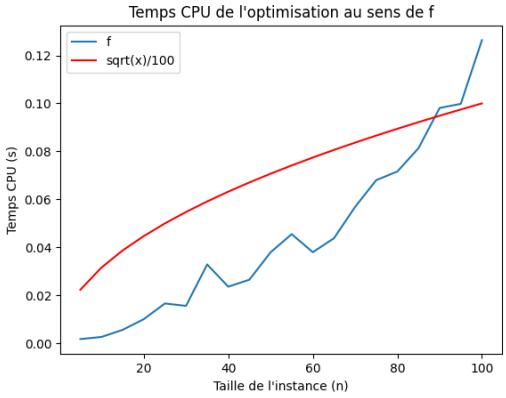
Question 6

Pour étudier le temps de calcul d'une affectation optimale au sens de f et g respectivement, on teste les fonctions respectives `q4` et `q5` sur des instances de taille variant de 5 à 200 par pas de 5. Pour une même taille, on génère 10 matrices d'utilité contenant des notes entières

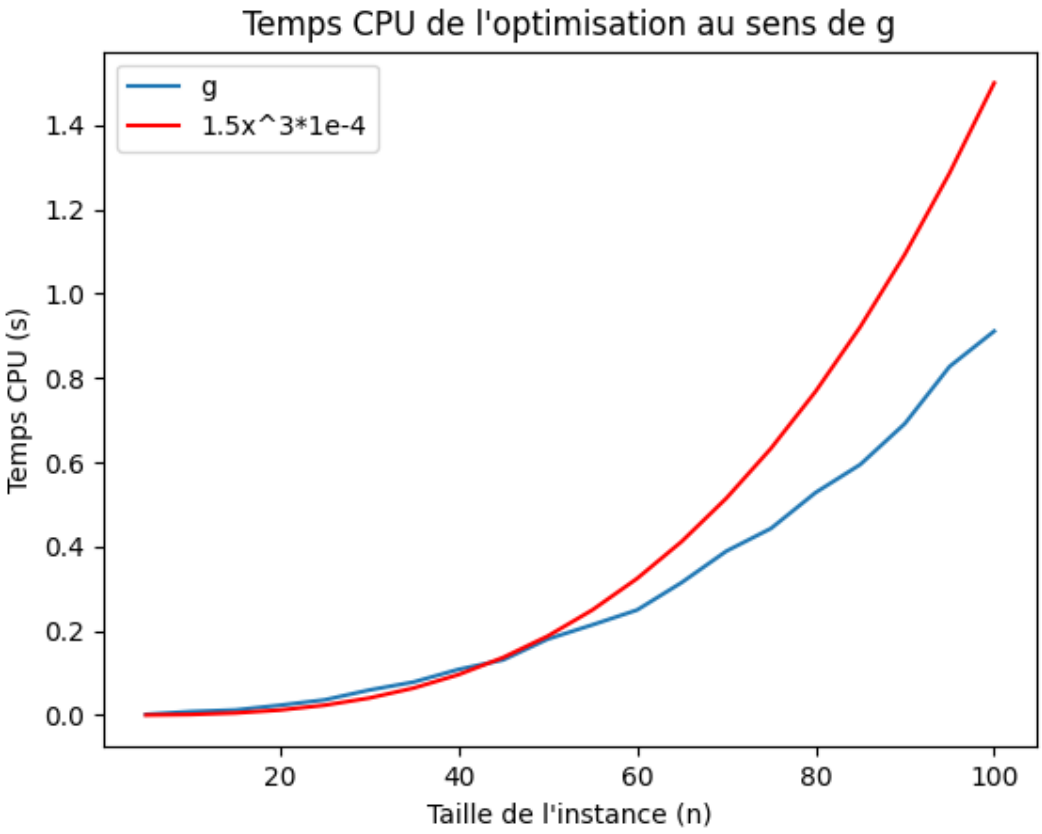
aléatoires comprises entre 0 et 20 et on calcule une affectation optimale au sens de f (c'est-à-dire le PLNE (2)) pour chacune d'elles, et on divise le temps de calcul total par 10, afin d'obtenir une moyenne de temps d'optimisation. On répète le même processus pour g et le PLNE (3).

En récoltant les données nécessaires de cette procédure, on peut modéliser l'évolution du temps d'optimisation selon f ainsi que selon g, afin de les comparer.

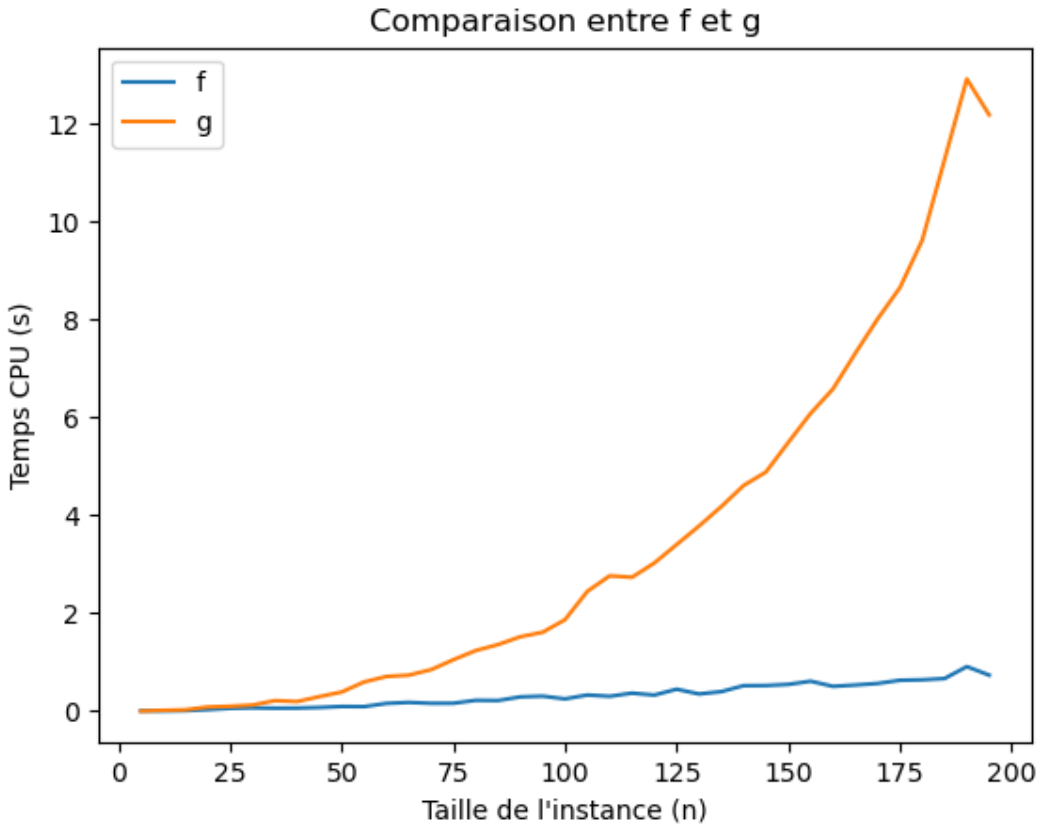




En effet, l'évolution du temps de calcul d'une affectation optimale selon f donc selon le PLNE (2) semble être meilleure qu'une évolution linéaire, mais pas vraiment de l'ordre de $\sqrt{n}$ non plus.



On remarque qu'au contraire, g ou le PLNE (3) suit plutôt une évolution cubique : $O(n^3)$.



Le temps de calcul d'une affectation optimale selon g augmente bien plus rapidement que celui selon f. En effet, g est soumis à plus de contraintes que f: la 3e contrainte de g comporte n^2 sous-contraintes alors que celle de f en comporte n. De plus, comme les 3e sous-contraintes de g sont plus complexes que celles de f, il est normal de voir une évolution plus importante du temps de calcul de la fonction `q5` par rapport à `q4`.