TME - Semaines 1 à 3

Yuxiang Zhang Antoine Lecomte

Partie 1 : Problème et affectation

Question 2

Nous avons différentes possibilités pour optimiser les performances de l'algorithme de Gale-Shapley en ce qui concerne les structures de données :

Nous utilisons plusieurs structures de données qui ont chacune des avantages pour les optimisations que nous cherchons à effectuer.

- **Structure** `deque` (liste doublement chaînée qui dispose des fonctions d'ajout/suppression en tête/queue `appendleft`, `popleft`, `append`, `pop`) : pour trouver un étudiant libre à chaque itération. Les étudiants sont récupérés dans l'ordre de leur numéro. Complexité constante $O(1)$.
- **Structure liste de Python**, de complexité $O(1)$ pour trouver le prochain parcours préféré par l'étudiant.
- **Structure liste de dictionnaires de Python**, où chaque dictionnaire comporte 11 associations (11 étudiants). Les clés sont les numéros des étudiants et les valeurs sont leur classement (qui démarre à 0 pour l'étudiant classé en tête). Il y a 9 dictionnaires dans la liste, car chaque dictionnaire représente le classement sur un Master.
- **Structure de heap inversé** : nous utilisons `heapq` en Python (file de priorité minimale), ce qui donne une complexité en $O(1)$ pour trouver l'étudiant classé en dernière position, car celui-ci est en tête du `heapq`.
- **Structures heap inversé + ensemble** (`set`, non ordonné) pour remplacer un étudiant par un autre dans l'affectation courante d'un parcours, avec une complexité $O(\log k)$ pour l'ajout et la suppression d'un étudiant. La structure de l'ensemble permet de vérifier directement la présence d'un étudiant en $O(1)$.

Question 3

Complexité temporelle globale :

- Initialisation des files, tableaux et dictionnaires : $O(m \times n)$ pour le classement des étudiants.
- Chaque étudiant fait au plus m propositions : $O(n \times m)$.
- Chaque insertion ou suppression dans un heap a un coût $O(\log k)$, où k est la capacité maximale d'un parcours.
- Dans le pire des cas (tous les parcours sont pleins et nécessitent des réaffectations), la complexité est $O(n \times m \times \log k)$.

Complexité spatiale globale :

- Classements des parcours : $O(m \times n)$.
- Affectations des étudiants : $O(m \times k)$, où k est la capacité maximale par parcours.
- File des étudiants libres : $O(n)$.

En résumé :

- Complexité temporelle : $O(n \times m \times \log k)$.
- Complexité spatiale : $O(m \times (n + k))$.

Question 4

Voici les structures de données utilisées pour l'optimisation des opérations de l'algorithme côté parcours :

- **Structure** `deque` pour maintenir la file des étudiants libres. Complexité en temps $O(1)$ pour l'ajout/retrait en tête ou queue. Espace : $O(n)$.
- **Structure liste** pour le classement des étudiants dans chaque parcours. Complexité d'accès $O(1)$. Espace : $O(m)$.
- **Structure liste de dictionnaires** pour le classement des étudiants par parcours. Recherche du rang en $O(1)$. Espace : $O(m \times n)$.
- **Structure heap inversé** (`heapq`) pour gérer les étudiants affectés à chaque parcours. Recherche du moins préféré en $O(1)$. Espace : $O(k)$.
- **Structures heap inversé + ensemble** (`set`) pour les suppressions/réaffectations. Ajout/suppression en $O(\log k)$, vérification de présence en $O(1)$.

Complexité temporelle globale :

- Classement des étudiants : $O(m \times n)$.

- Initialisation des `deque` et `heap` : $O(n)$ et $O(m)$.
- Propositions des étudiants : $O(m \times n)$.
- Retrait d'un étudiant d'un parcours : $O(k)$.

Complexité totale (pire cas) : $O(n \times m \times k)$.
Complexité spatiale globale :

- Classements des parcours : $O(m \times n)$.
- Affectations des étudiants : $O(m \times k)$.
- File des étudiants libres : $O(n)$.
- `Heaps` et `sets` : $O(m \times k)$.

Complexité spatiale totale : $O(m \times (n + k))$.

Question 5

Affectations finales (Côté étudiants) :

- Étudiants 3, 5 : 0
- Étudiant 4 : 1
- Étudiant 9 : 2
- Étudiant 8 : 3
- Étudiant 10 : 4
- Étudiant 0 : 5
- Étudiant 1 : 6
- Étudiant 7 : 7
- Étudiants 2, 6 : 8

Affectations finales (Côté parcours) :

- Parcours 0 : 5, 3
- Parcours 1 : 4
- Parcours 2 : 9
- Parcours 3 : 8
- Parcours 4 : 10

- Parcours 5 : 1
- Parcours 6 : 0
- Parcours 7 : 7
- Parcours 8 : 6, 2

Question 6

À l'exécution :

Vérification de la stabilité de l'affectation (étudiants) :

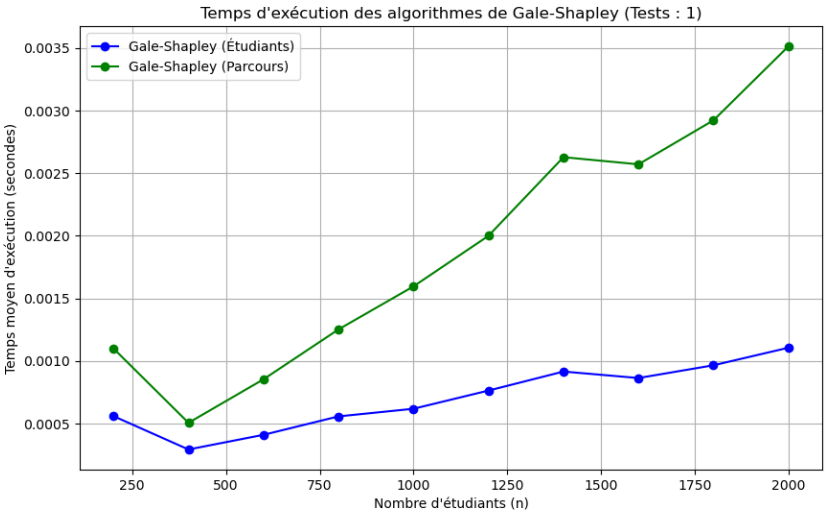
- Aucune paire instable trouvée. L'affectation est stable.

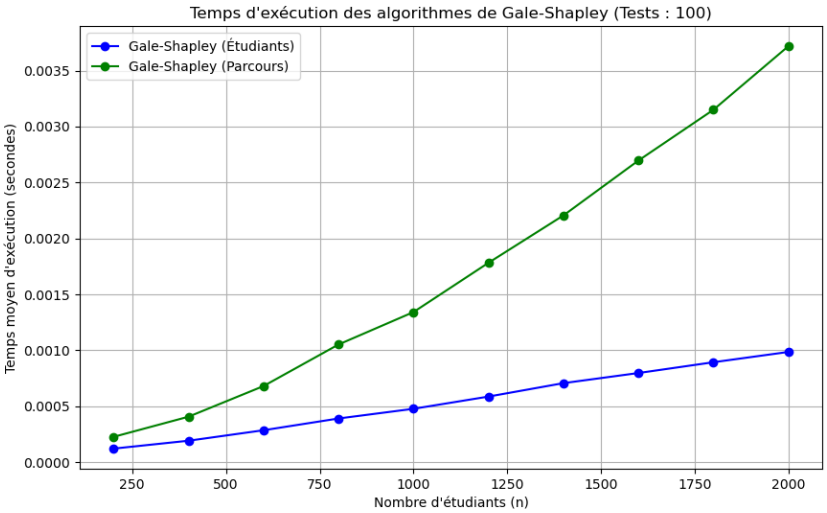
Vérification de la stabilité de l'affectation (parcours) :

- Aucune paire instable trouvée. L'affectation est stable.

Le test de vérification pour rechercher d'éventuelles paires instables échoue,... (pas de paires instables détectées)

Question 8





Partie 2 : Evolution du temps de calcul

Question 9

La complexité temporelle observée semble cohérente avec l'analyse théorique des algorithmes de Gale-Shapley. L'algorithme centré sur les étudiants est plus efficace, lorsque l'on fait des moyennes de temps d'exécution.

La courbe bleue concernant Gale-Shapley côté étudiant montre une croissance linéaire modérée du temps d'exécution avec le nombre d'étudiants. Le temps moyen d'exécution reste relativement bas, même pour 2000 étudiants. La croissance linéaire est cohérente avec la complexité O(n * m * log k). Comme m (nombre de parcours) et k (capacité des parcours) restent constants, le facteur dominant est O(n).

La courbe verte concernant Gale-Shapley côté parcours croît plus rapidement et à partir de n=600 étudiants, la différence devient significative et la courbe semble avoir une croissance pluôt en forme de parabole. L'algorithme côté parcours devient rapidement moins performant à mesure que le nombre d'étudiants augmente et la courbe montre une croissance plus rapide : cohérent avec O(m * n * k).

Question 10

Jusqu'à 100 tests effectués, nous obtenons des courbes quasi-linéaires, et cela en fonction du nombre d'étudiants impliqués. Les courbes obtenues sont toutes conformes à nos attentes. (Petit détail : lorsqu'on effectue un seul test, on remarque que le temps d'exécution pour 200 étudiants est anormale-

ment long, c'est le délai au lancement du programme qui est responsable de cette valeur sur le graphique, car cela ne se répète pas lorsque le programme fait de multiples tests et en fait une moyenne.)

Nous pouvons noter que les temps d'exécutions pour Gale-Shapley côté étudiant vont de 0.2 milliseconde approximativement pour 200 étudiants à 1 milliseconde en moyenne pour 2000 étudiants. Pour l'algorithme côté parcours, nous obtenons environ 0.25 milliseconde comme temps d'exécution avec 200 étudiants et jusqu'à 3.7 millisecondes en moyenne avec 2000 étudiants. Donc, nous voyons très clairement que l'algorithme de Gale-Shapley côté étudiant est plus efficace que celui côté parcours, au fur et à mesure que l'on augmente le nombre d'étudiants. De plus, le nombre m de parcours est seulement de 9, donc les deux algorithmes restent très performants. Avec davantage de parcours considérés (et donc des capacités k pour chaque parcours nettement supérieures), nous obtiendrions des différences bien plus significatives entre les temps d'exécution des deux versions de l'algorithme, car cela augmenterait évidemment le nombre de changement de parcours pour les étudiants, ce qui va allonger davantage les temps d'exécution pour l'algorithme côté parcours que celui côté étudiant.

Partie 3 : Equité et PL(NE)

Question 11

Dans cette étude, nous avons utilisé les fichiers `PrefEtu.txt` et `PrefSpe.txt` pour $n = 11$ et $k = 3$. Notre objectif est de générer un fichier `.lp` correspondant au Programme Linéaire en Nombres Entiers (PLNE) et de le résoudre à l'aide du solveur Gurobi.

1 Problème et Modélisation en PLNE

1.1 Définitions et Notations

- S : ensemble des étudiants.
- P : ensemble des parcours.
- $x_{i,j}$: variable binaire indiquant si l'étudiant i est affecté au parcours j :

$$x_{i,j} = \begin{cases} 1, & \text{si l'étudiant } i \text{ est affecté au parcours } j \\ 0, & \text{sinon} \end{cases}$$

- $U_{i,j}$: score de Borda attribué par l'étudiant i au parcours j.
- m : nombre total de parcours et k le nombre maximal de choix considérés.

1.2 Formulation PLNE

Contraintes :

1. **Chaque étudiant est affecté à un seul parcours** :

$$\sum_{j \in P} x_{i,j} = 1, \quad \forall i \in S$$

2. **Respect des capacités des parcours** :

$$\sum_{i \in S} x_{i,j} \leq C_j, \quad \forall j \in P$$

3. **Affectation dans les k premiers choix** :

$$\sum_{j \in P : U_{i,j} \geq m-k} x_{i,j} = 1, \quad \forall i \in S$$

4. **Variables binaires** :

$$x_{i,j} \in \{0, 1\}, \quad \forall i \in S, \forall j \in P$$

1.3 Génération du fichier LP avec Python

Le modèle PLNE est généré à l'aide de la fonction `generate_lp_file_k`, qui prend en entrée les préférences des étudiants et des parcours, ainsi que les capacités des parcours et le nombre maximal de choix k pour chaque étudiant.

Ce fichier LP est ensuite résolu à l'aide de Gurobi pour trouver une solution qui respecte les contraintes d'affectation, de capacité et de choix parmi les k premiers choix des étudiants.

1.4 Résolution avec Gurobi

Une fois le fichier LP généré, il peut être résolu avec Gurobi pour déterminer l'affectation optimale des étudiants aux parcours, en respectant les contraintes mentionnées.

1.5 Analyse des Résultats

Une fois la solution obtenue, nous analyserons l'affectation des étudiants et la capacité de chaque parcours. Nous examinerons également l'utilité moyenne et minimale des étudiants pour évaluer la qualité de la solution.

2 Génération du fichier `.lp`

Nous avons écrit une fonction permettant de générer un fichier `.lp` représentant notre problème d'affectation. La méthode suit les étapes suivantes :

1. Définition des variables $x_{i,j}$, représentant l'affectation de l'étudiant i au parcours j.
2. Ajout des contraintes :
   - Chaque étudiant est affecté à un seul parcours.
   - Respect des capacités des parcours.
   - Le choix des étudiants respecte les k premiers choix.
3. Maximisation de la fonction objective : maximiser les scores de Borda.

Question 12

Nous avons utilisé la fonction `generate_lp_file_k`. Après l'exécution de cette fonction, nous avons trouvé les contraintes dans le fichier `affectation_k.lp`.

Ensuite, nous avons exécuté la commande suivante :

```
gurobi_cl ResultFile=solution.sol affectation_k.lp
```

Les résultats obtenus sont les suivants :

```
Model is infeasible
Best objective -, best bound -, gap -
Unable to retrieve attribute 'X'
```

Cela signifie qu'il n'y a pas de solution pour $k = 3$.

Question 13

À l'aide de la fonction `find_min_k`, nous avons trouvé la valeur minimale de k permettant à tous les étudiants d'obtenir un de leurs k premiers choix dans les affectations pour les Masters. Echec à k=4, et affectation stable à partir de k=5, donc il est possible d'obtenir au moins une affectation stable à partir de ce kmin.

Maximisation de l'utilité minimale des étudiants

Nous cherchons à maximiser l'utilité minimale des étudiants pour un k donné. Le problème est formulé sous forme d'un Programme Linéaire en Nombres Entiers (PLNE) avec les contraintes suivantes :

1. **Affectation unique de chaque étudiant** : Chaque étudiant i doit être affecté à un seul parcours parmi ses k premiers choix. Cela se traduit par la contrainte suivante :

$$\sum_{j \in P_i^k} x_{i,j} = 1, \quad \forall i \in S$$

où P_i^k est l'ensemble des k premiers choix de l'étudiant i, et $x_{i,j}$ est une variable binaire indiquant si l'étudiant i est affecté au parcours j.

2. **Respect des capacités des parcours** : La capacité de chaque parcours j doit être respectée, c'est-à-dire que le nombre d'étudiants affectés à ce parcours ne doit pas dépasser sa capacité. La contrainte est formulée comme suit :

$$\sum_{i \in S} x_{i,j} \leq C_j, \quad \forall j \in P$$

où C_j est la capacité du parcours j.

3. **Maximisation de l'utilité minimale** : L'objectif est de maximiser l'utilité minimale des étudiants. La contrainte relative à l'utilité minimale est la suivante :

$$\sum_{j \in P_i^k} U_{i,j} \cdot x_{i,j} \geq U_{\min}, \quad \forall i \in S$$

où $U_{i,j}$ est le score de Borda combiné pour l'étudiant i et le parcours j, et $U_{\min}$ est la variable représentant l'utilité minimale que nous cherchons à maximiser.

4. **Objectif** : L'objectif du problème est de maximiser l'utilité minimale $U_{\min}$.

$$\text{Maximiser } U_{\min}$$

Nous avons utilisé `PULP` en utilisant la méthode `PULP_CBC_CMD`. Le résultat renvoie l'affectation optimale des étudiants aux parcours ainsi que l'utilité minimale atteinte.

Remarque : Dans la solution obtenue, chaque affectation de l'étudiant à un parcours est représentée par $x_{i,j} = 1$, indiquant que l'étudiant i est affecté au parcours j, et l'utilité minimale obtenue correspond à la valeur de $U_{\min}$.

Les résultats renvoyés sont les suivants :

Votre choix : 5
Aucune solution trouvée pour $k = 1$, tentative avec $k = 2$
Aucune solution trouvée pour $k = 2$, tentative avec $k = 3$

Aucune solution trouvée pour $k = 3$, tentative avec $k = 4$
Aucune solution trouvée pour $k = 4$, tentative avec $k = 5$
Solution trouvée pour $k = 5$
Le plus petit k pour lequel la solution est faisable est : 5

Afin de maximiser l'utilité minimale des étudiants, nous avons utilisé la fonction `maximize_min_utility`, qui, à l'aide de la fonction `score_borda_combined` (fonction renvoyant une matrice contenant l'ensemble des sommes des scores étudiants et parcours), nous a permis de déterminer l'utilité minimale la plus haute, atteinte.

L'affichage obtenu est le suivant :

Plus petit k trouvé : 5
Affectation optimale : {(0, 8): 1.0, (1, 4): 1.0, (2, 0): 1.0, (3, 0): 1.0, (4, 1): 1.0, (5, 7): 1.0, (6, 5): 1.0, (7, 2): 1.0, (8, 6): 1.0, (9, 8): 1.0, (10, 3): 1.0}
Utilité minimale atteinte : 8.0

Les valeurs `1.0` indiquent que les affectations des étudiants aux parcours ont bien été réalisées. Sinon, elles seraient à `0.0` (mais l'algorithme renvoie directement la solution trouvée, donc il n'y a pas de `0.0`). Les couples sont sous la forme (étudiant, parcours), et l'utilité minimale trouvée est de 8 pour $k = 5$ ($k_{\min}$). Cela signifie que l'étudiant le moins satisfait a une utilité minimale de 8, et tous les autres étudiants ont au moins 8 comme utilité minimale également.

Modèle de Programmation Linéaire pour Maximiser l'Efficacité Totale avec Équité

Question 14

2.1 PLNE pour maximiser la somme des utilités

Nous devons écrire un **Programme Linéaire en Nombres Entiers (PLNE)** permettant de maximiser la somme des utilités des étudiants et des parcours.

2.1.1 Modèle PLNE

Variables de décision :

$$x_{i,j} \in \{0, 1\}$$

où $x_{i,j} = 1$ si l'étudiant i est affecté au cours j, sinon 0.

Objectif : Maximiser la somme des utilités (score Borda combiné) :

$$\max \sum_{i=1}^{n} \sum_{j \in P_i} s_{i,j} \cdot x_{i,j}$$

où $s_{i,j}$ est le score Borda combiné du couple (i, j) et P_i est l'ensemble des k premiers choix de l'étudiant i.

Contraintes :

- **Chaque étudiant est affecté à un seul cours**

$$\sum_{j \in P_i} x_{i,j} = 1, \quad \forall i \in \{1, ..., n\}$$

- **Capacité des parcours respectée**

$$\sum_{i : j \in P_i} x_{i,j} \leq c_j, \quad \forall j \in \{1, ..., m\}$$

- **Les variables sont binaires**

$$x_{i,j} \in \{0, 1\}, \quad \forall i, j$$

2.1.2 Résolution avec Gurobi

Nous avons résolu le PLNE pour différentes valeurs de k :

k	Utilité totale atteinte	Utilité moyenne atteinte	Utilité minimale	Valeur objective
5	131	11.91	2	129
6	144	13.09	7	144
7	153	13.91	10	153
8	153	13.91	10	153
9	153	13.91	10	153

Table 1: Résultats de l'optimisation en fonction de k

Interprétation :

- L'utilité totale **augmente avec** k, mais **se stabilise à** $k = 7$.
- À partir de $k = 7$, **l'utilité minimale atteint 10**, garantissant une meilleure équité.
- $k = 7$ **est donc un bon choix**, car il maximise l'utilité et assure une équité entre étudiants.

Réponse finale :

- **Utilité moyenne obtenue :** 13.91
- **Utilité minimale obtenue :** 10
- **Valeur optimale de** k **choisie :** 7

Resultat obtenu:

Parcours	Etudiant(s)
0	[4, 5]
1	[10]
2	[7]
3	[3]
4	[9]
5	[0]
6	[8]
7	[6]
8	[1, 2]

Question 15

Nous souhaitons maximiser la somme des utilités combinées des étudiants et des parcours tout en garantissant une utilité minimale pour chaque étudiant. Le modèle de programmation linéaire est défini comme suit :

Variables de décision

- $x_{ij} = \begin{cases} 1 & \text{si l'étudiant } i \text{ est affecté au parcours } j \\ 0 & \text{sinon} \end{cases}$
- $U_{\min} \geq 0$: Utilité minimale garantie pour chaque étudiant.

Paramètres

- n : Nombre d'étudiants.
- m : Nombre de parcours.
- k : Nombre de premiers choix considérés pour chaque étudiant.
- S_{ij} : Score de Borda combiné de l'étudiant i pour le parcours j.
- C_j : Capacité maximale du parcours j.

Fonction Objectif

Nous maximisons la somme des utilités combinées des étudiants et des parcours, en ajoutant un poids pour garantir l'équité :

$$\max \left(\sum_{i=1}^{n} \sum_{j \in P_i^k} S_{ij} \cdot x_{ij} + U_{\min} \right)$$

où P_i^k est l'ensemble des k premiers choix de l'étudiant i.

Contraintes

1. **Affectation unique des étudiants :**

$$\sum_{j \in P_i^k} x_{ij} = 1, \quad \forall i \in \{1, \ldots, n\}$$

2. **Respect des capacités des parcours :**

$$\sum_{\substack{i=1 \\ j \in P_i^k}}^{n} x_{ij} \leq C_j, \quad \forall j \in \{1, \ldots, m\}$$

3. **Utilité minimale garantie pour chaque étudiant :**

$$\sum_{j \in P_i^k} S_{ij} \cdot x_{ij} \geq U_{\min}, \quad \forall i \in \{1, \ldots, n\}$$

4. **Variables binaires :**

$$x_{ij} \in \{0, 1\}, \quad \forall i \in \{1, \ldots, n\}, \quad \forall j \in P_i^k$$

En appliquant la fonction `maximize_utility_and_fairness` avec les paramètres suivants :

- **Valeur de** k : 5

Nous obtenons les résultats suivants :

- **Utilité totale atteinte** : 131.0
- **Utilité moyenne atteinte** : 11.91
- **Utilité minimale des étudiants** : 8.0

Conclusion

L'application de ce modèle d'optimisation a permis de maximiser l'efficacité totale tout en garantissant une certaine équité entre les étudiants. L'utilité moyenne atteinte est de 11.91, ce qui reflète une distribution globale satisfaisante des affectations. Cependant, l'utilité minimale des étudiants reste relativement basse à 2.0, ce qui indique qu'au moins un étudiant bénéficie d'une affectation loin de ses préférences optimales. Cela souligne la difficulté d'équilibrer parfaitement efficacité et équité dans ce type de problème d'affectation.

Comparaison des différentes solutions obtenues

Question 16

Ensemble des affectations trouvées pour chaque question :

Question	Affectations
Q2	0: [3, 5], 1: [4], 2: [9], 3: [8], 4: [10], 5: [0], 6: [1], 7: [7], 8: [2, 6]
Q3	0: [5, 3], 1: [4], 2: [9], 3: [8], 4: [10], 5: [1], 6: [0], 7: [7], 8: [6, 2]
Q13	0: [2, 3], 1: [4], 2: [7], 3: [10], 4: [1], 5: [6], 6: [8], 7: [5], 8: [0, 9]
Q14	0: [4, 5], 1: [10], 2: [7], 3: [3], 4: [9], 5: [0], 6: [8], 7: [6], 8: [1, 2]
Q15	0: [7, 10], 1: [4], 2: [6], 3: [9], 4: [5], 5: [0], 6: [1], 7: [3], 8: [2, 8]

Nous avons plusieurs solutions :

- **Gale-Shapley côté étudiants (Résultats Q2)** : C'est la solution obtenue par l'algorithme Gale-Shapley basé sur les préférences des étudiants.
- **Gale-Shapley côté parcours (Résultats Q3)** : C'est la solution obtenue par l'algorithme Gale-Shapley basé sur les préférences des parcours.
- **Solutions des questions 13, 14 et 15** : Ces solutions proviennent de l'optimisation visant à maximiser l'utilité totale tout en respectant certaines contraintes supplémentaires.

Nous allons comparer ces solutions selon les critères suivants :

- **Stabilité** (existe-t-il des paires instables ?)
- **Utilité moyenne** (la moyenne des utilités des étudiants)
- **Utilité minimale** (l'utilité minimale parmi les étudiants)
- **Nombre de paires instables** (déterminé par la question 6)

1. Stabilité

Selon la nature de l'algorithme **Gale-Shapley**, il garantit que les paires retournées sont stables. Par conséquent, les solutions obtenues par **Q2** et **Q3** doivent être stables.

- **Résultats Q2 (GS côté étudiants)** : La solution est stable par définition.
- **Résultats Q3 (GS côté parcours)** : La solution est stable par définition.
- **Résultats Q13** : Instable d'après le résultat obtenu.

- **Résultats Q14** : Instable d'après le résultat obtenu.
- **Résultats Q15** : Instable d'après le résultat obtenu.

Si nous souhaitons trouver des affectations stables (contrainte), il faut donc utiliser l'algorithme de Gale-Shapley, mais néanmoins, cet algorithme (côté étudiant comme côté parcours) ne garantit pas l'équité et l'efficacité.

2. Utilité moyenne

L'utilité moyenne représente la satisfaction globale des étudiants. Une utilité moyenne plus élevée indique une plus grande satisfaction des étudiants.

- **Résultats Q2 (GS côté étudiants)** : A déterminer
- **Résultats Q3 (GS côté parcours)** : A déterminer
- **Résultats Q13** : Utilité moyenne 11.91
- **Résultats Q14** : Utilité moyenne 13.91
- **Résultats Q15** : Utilité moyenne 11.72

La solution **Q14** a la plus haute utilité moyenne, indiquant que les étudiants sont plus satisfaits, comparé aux autres solutions. Donc la meilleure solution est k=7 pour maximiser l'utilité moyenne.

3. Utilité minimale

L'utilité minimale est la valeur de l'utilité la plus faible parmi tous les étudiants. Idéalement, on souhaite que cette valeur soit aussi élevée que possible afin de garantir l'équité.

- **Résultats Q2 (GS côté étudiants)** : A déterminer
- **Résultats Q3 (GS côté parcours)** : A déterminer
- **Résultats Q13** : Utilité minimale 2.0
- **Résultats Q14** : Utilité minimale 10.0
- **Résultats Q15** : Utilité minimale 8.0

La solution, **Q14** a une utilité minimale de 10, ce qui est bien plus équitable que les solutions des autres questions, qui ont des utilités minimales inférieures ou égales à 8.

Conclusion

- **Stabilité** : Les solutions **Q2** et **Q3** sont stables (garanties par Gale-Shapley), tandis que les solutions **Q13**, **Q14**, et **Q15** sont plus optimisées, mais peuvent contenir des paires instables.
- **Utilité moyenne** : La solution **Q14** a la plus grande utilité moyenne, ce qui signifie que les étudiants sont globalement plus satisfaits.
- **Utilité minimale** : La solution **Q14** a la plus grande utilité minimale (10), ce qui est plus équitable que **Q13** et **Q15**, qui ont des utilités minimales inférieures ou égales à 8 (à partir de k=5).

En résumé, les solutions **Q13**, **Q14**, et **Q15** offrent le meilleur compromis, bien qu'elles puissent avoir quelques paires instables, elles améliorent l'utilité des étudiants et sont plus équitables que les affectations données par Gale-Shapley.