POO pour l'optimisation
IG2I, LE4-SI – 2021-2022

Projet – Échange de reins

Diego Cattaruzza, Maxime Ogier

Ce sujet a été rédigé à partir des travaux de thèse de Lucie Pansart [1], et d'une étude de cas proposée par Hadrien Cambazard et Nicolas Catusse [2].

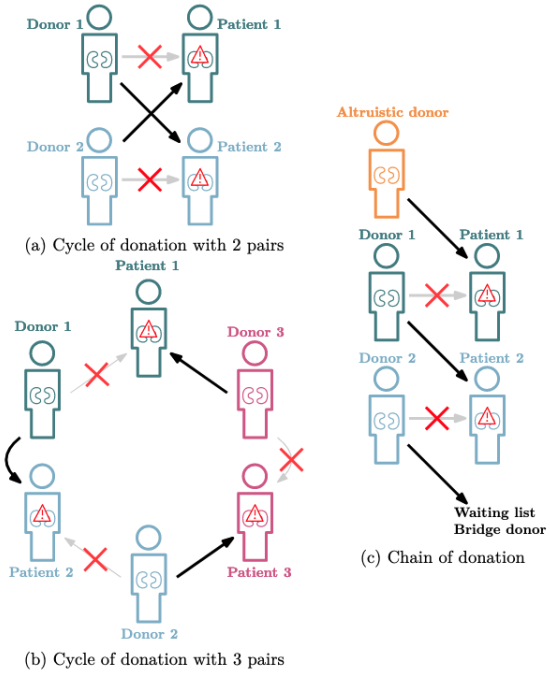
1 Description générale du contexte et du problème

Dans un marché de troc, les participants s'échangent des biens ou des services sans utiliser de moyen de paiement. En général, le troc se passe entre deux personnes, localement et immédiatement, et sans l'intervention d'une organisation étatique. Il y a cependant une exception : les échanges de reins sont organisés par les institutions gouvernementales et peuvent impliquer beaucoup de monde. Les organes ne peuvent pas être vendus, ni échangés d'ailleurs : dans un programme d'échange de reins, ce qui est échangé ce sont les donneurs, pas les reins. Les acteurs du marché sont les patients attendant une greffe de rein à cause d'une maladie rénale. Chaque patient est associé à un proche prêt à donner un rein, mais avec qui il est incompatible. De plus, de nouveaux reins peuvent être injectés dans le marché grâce aux donneurs altruistes. Le rôle des programmes d'échange de reins est de déterminer quels échanges doivent être réalisés, dans le but d'optimiser le "bien commun", tout en respectant les contraintes médicales, légales, éthiques et logistiques. En d'autres termes, ils doivent résoudre un problème d'optimisation combinatoire. En 2017, 90 306 transplantations rénales ont été réalisées dans le monde tandis que 1,2 million de personnes sont décédées d'une maladie rénale [3, 4].

Un patient reçoit en général le rein d'un donneur au moment de son décès et attend des années qu'un donneur compatible soit disponible. La compatibilité est liée aux groupes sanguins (compatibilité ABO) et à certains anti-corps (compatibilité HLA)[1]. Mais un patient peut aussi obtenir un rein d'un donneur vivant car un seul rein est suffisant chez l'humain. Ainsi, la transplantation à partir de donneurs vivants a été la première réalisée en 1950 et s'avère finalement la plus efficace. Par ailleurs, les progrès en immunothérapie ont permis d'augmenter les possibilités d'échange.

[1] Des informations supplémentaires sur la compatibilité peuvent être obtenues ici: https://web.stanford.edu/dept/HPS/transplant/html/hla.html

Ainsi quand un patient trouve un donneur volontaire (en général un membre de sa famille) mais incompatible, ils peuvent entrer dans un programme d'échange et espérer trouver une autre paire avec qui l'échange est possible. En fait, un cycle d'échange avec plusieurs autres paires est tout à fait possible. Mais un donneur ne donne son rein que si il a la garantie que son patient en reçoit également un. Enfin, il y a des donneurs dits *altruistes* qui peuvent également participer, ce qui mène à des chaînes d'échange. Trois situations typiques sont représentées dans la Figure 1.



(a) Cycle of donation with 2 pairs

(b) Cycle of donation with 3 pairs

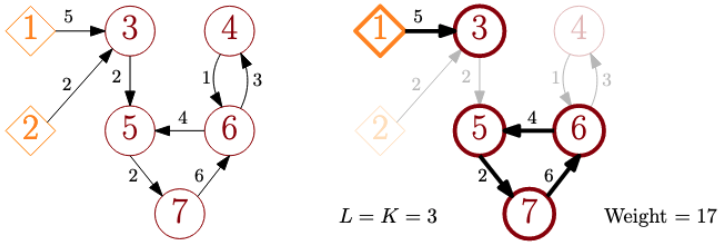
(c) Chain of donation

Figure 1: Représentation de 3 types d'échanges: (a) un cycle d'échange de taille 2. (b) un cycle de taille 3. (c) une chaîne initiée par un donneur altruiste.

Un donneur pourrait se retirer du cycle d'échange de façon impromptu dès lors que son patient a reçu un rein et briser le cycle. Pour éviter cette situation, les opérations doivent se dérouler simultanément... Cela limite la taille K des cycles en pratique (nombre de chirurgiens, d'équipements, etc.) à $K = 3$ ou $K = 4$ paires maximum. Le record est détenu par la république Tchèque qui a réalisé un cycle de taille 7 [5]. La longueur maximum L des chaînes de donations

peut être nettement plus importante car un patient reçoit toujours un rein avant que son donneur ne fournisse le sien. Ainsi, la plus longue chaîne a été réalisée aux États-Unis et comportait 68 patients [6]. Néanmoins la taille des chaînes est plus modeste en pratique et nous considérons des valeurs de L entre 4 et 13 dans cette étude de cas. Ainsi une chaîne d'échange ne peut pas contenir plus de L échanges.

Le problème peut être représenté sous la forme d'un graphe orienté et pondéré $\mathcal{D} = (\mathcal{V} = \mathcal{P} \cup \mathcal{N}, \mathcal{A})$. Chaque sommet de $\mathcal{P}$ représente une paire de patient-donneur et chaque sommet de $\mathcal{N}$ est un donneur altruiste. Un arc (i, j) dans $\mathcal{A}$ signifie que le rein du donneur de la paire i (ou du donneur altruiste i) peut être transplanté au patient de la paire j. Un poids $w_{ij} \in \mathbb{R}^+$ indique le bénéfice médical de cette transplantation. La définition de la fonction w donnant le bénéfice médical est évidemment une question très épineuse qui mobilise médecins et mathématiciens. La figure 2 présente un exemple de ce modèle de graphe des données et une solution. De façon générale, le problème consiste à trouver un ensemble d'échanges disjoints (un donneur ne peut apparaître que dans un échange) maximisant la somme des bénéfices médicaux. Les échanges, en termes de graphes, sont des *circuits* élémentaires de $\mathcal{D}$ et des *chemins* élémentaires de $\mathcal{D}$ ayant pour origine un sommet de $\mathcal{N}$.



(a) Model (b) Solution of the KEP

Figure 2: Représentation du modèle de graphe (a) et d'une solution (b) (pour les besoin d'un exemple suffisamment petit, on a pris $L = K = 3$). Les donneurs altruistes sont représentés par des losanges et les paires patient/donneurs par des cercles.

2 Description détaillée du problème à résoudre

2.1 Les données

Le problème de l'échange de reins comporte les données suivantes :

- P : le nombre de paires patient-donneur ;
- N : le nombre de donneurs altruistes ;
- K : la taille maximale des cycles d'échange entre paires patient-donneur ;
- L : la taille maximale des chaînes d'échange initiées par un donneur altruiste ;
- $\mathcal{P}$: l'ensemble des paires patient-donneur ;
- $\mathcal{N}$: l'ensemble des donneurs altruistes.

Par ailleurs, afin de modéliser la compatibilité entre donneur et receveur, ainsi que l'intérêt médical des transplantations, nous utilisons les données suivantes :

- $\mathcal{A}$: l'ensemble des transplantations réalisables : $(i, j) \in \mathcal{A}$ si et seulement si le donneur de la paire $i \in \mathcal{P}$ ou le donneur altruiste $i \in \mathcal{N}$ est compatible avec le receveur de la paire $j \in \mathcal{P}$;
- $w_{ij} \in \mathbb{R}^+$: indique pour chaque arc $(i, j) \in \mathcal{A}$ le bénéfice médical de la transplantation du rein du donneur de la paire $i \in \mathcal{P}$ ou du donneur altruiste $i \in \mathcal{N}$ au receveur de la paire $j \in \mathcal{P}$.

2.2 Les décisions

Pour résoudre le problème d'échange de reins, il faut sélectionner un ensemble de transplantations qui seront effectuées parmi l'ensemble des transplantations $\mathcal{A}$.

2.3 Les contraintes du problème

Pour qu'une solution soit réalisable, il faut qu'elle respecte les conditions suivantes :

- chaque donneur altruiste $i \in \mathcal{N}$ ne peut être donneur que dans une transplantation au maximum ;
- chaque paire patient-donneur $i \in \mathcal{P}$ ne peut être donneur que dans une transplantation au maximum ;
- chaque paire patient-donneur $i \in \mathcal{P}$ ne peut être receveur que dans une transplantation au maximum ;
- chaque paire patient-donneur $i \in \mathcal{P}$ ne peut être donneur dans une transplantation que s'il est receveur d'une autre transplantation ;
- les cycles de transplantation sont limités à K transplantations ;
- les chaînes de transplantations sont limitées à L transplantations.

Notez que les quatre premières contraintes sont équivalentes au fait de dire qu'on ne peut faire que des cycles et des chaînes de transplantations, avec chaque donneur altruiste n'apparaissant qu'au début d'une seule chaîne au maximum, et chaque paire patient-donneur n'apparaissant que dans une seule chaîne ou un seul cycle au maximum.

2.4 Fonction objectif

L'objectif est ici de maximiser la somme des bénéfices médicaux des transplantations réalisées.

Soit $\mathcal{A}^{sol}$ l'ensemble des transplantations de la solution. Alors l'objectif est de maximiser :

$$\sum_{(i,j)\in\mathcal{A}^{sol}} w_{ij}.$$

2.5 Hypothèses sur les données

On suppose qu'on peut toujours faire au moins une transplantation. Mais il n'y a aucune garantie qu'il existe une solution dans laquelle toutes les paires patient-donneur reçoivent une transplantation.

Notez par ailleurs que certaines instances peuvent ne contenir aucun donneur altruiste. Il n'est alors pas possible de faire de chaînes de transplantations, on ne peut faire que des cycles.

3 Solution triviale et solutions pour appréhender le problème

Évidemment, une solution réalisable triviale est de ne faire aucun échange. Cette solution a un coût de 0, c'est donc la pire solution qui soit.

Afin d'appréhender le problème, on peut commencer par chercher une solution dans laquelle on ne considère que des cycles de taille 2. Par la suite, on peut se limiter à chercher des cycles de taille K au maximum. Notez que ce type de solution sera tout à fait valable pour des instances qui ne contiennent aucun donneur altruiste.

On peut ensuite chercher des solutions avec seulement des chaînes dans le cas où il y a des donneurs altruistes. Et enfin, regarder des solutions avec à la fois des chaînes et des cycles d'échange.

4 Fichiers d'instance et fichiers de solution

4.1 Fichiers d'instance

Un ensemble de 22 instances initiales est fourni sur Moodle dans le dossier `instancesInitiales`. Vous devrez faire vos tests sur tous ces fichiers, plus éventuellement vos propres instances. Au cours du projet, nous rajouterons des instances supplémentaires. Vos résultats finaux seront évalués sur l'ensemble des instances.

Chaque instance est sous forme d'un fichier texte, avec comme séparateur des tabulations. Les lignes de commentaire commencent par '//'. Voici les données que vous allez trouver, dans l'ordre, dans un fichier d'instance.

- `P : Nombre de paires patient-donneur` : le nombre de paires patient-donneur.
- `N : Nombre de donneurs altruistes` : le nombre de donneurs altruistes.
- `K : Taille max cycles` : la taille maximale des cycles d'échange de reins.
- `L : Taille max chaines` : la taille maximale des chaînes d'échange de reins.
- `Matrice des bénéfices médicaux des échanges` : la matrices avec les bénéfices médicaux des échanges w_{ij}. Les N premières lignes correspondent aux donneurs altruistes. Les P dernières lignes correspondent aux paires patient-donneur. Sur chaque ligne i, on indique le bénéfice médical de la transplantation du rein du donneur $i \in \mathcal{P} \cup \mathcal{N}$ au patient de la paire patient-donneur $j \in \mathcal{P}$. La matrice contient donc $N + P$ lignes et P colonnes. Une valeur -1 dans la matrice signifie que la transplantation n'est pas réalisable. Ainsi cette matrice permet à la fois de décrire l'ensemble des transplantations réalisables $\mathcal{A}$, et aussi de donner la valeur du bénéfice médical w_{ij} de chaque arc $(i, j) \in \mathcal{A}$.

Vous trouvez sur Moodle le fichier `testInstance.txt` qui est un exemple d'instance, tiré de la Figure 2.

4.2 Fichier de solution

Si le nom du fichier d'instance est `nomInstance.txt`, alors le nom du fichier de solution associé devra être `nomInstance_sol.txt`. Ceci est notamment nécessaire pour que la solution puisse être contrôlée par le *checker* proposé (voir Section 5).

Chaque fichier solution est sous forme d'un fichier texte, avec comme séparateurs des tabulations. Vous pouvez mettre des commentaires sur une ligne dans le fichier solution : pour être considérés comme des commentaires, la ligne doit commencer par '//'.

Le fichier de solution doit contenir les informations suivantes :

- la première ligne de contenu contient un nombre entier qui représente le coût total de la solution (voir Section 2.4) ;
- chacune des lignes suivantes doit contenir les informations sur un cycle ou une chaîne d'échange : il s'agit de donner les identifiants des donneurs dans la chaîne ou le cycle en respectant les modalités suivantes :

– les donneurs altruistes sont indicés de 1 à N, et les paires patient-donneur sont indicées de $N+1$ à $N+P$ (s'il n'y a aucun donneur altruiste, la première patient-donneur a donc pour indice 1) ;
– les cycles peuvent commencer par n'importe quel identifiant, et il ne doit pas y avoir d'identifiant en double ;
– les chaînes doivent commencer par un identifiant d'un donneur altruiste (qui est celui qui initie la chaîne).

Vous trouvez sur Moodle le fichier `testInstance_sol.txt` qui est un exemple de fichier solution, lié à l'instance `testInstance.txt`. Il s'agit de la solution représentée sur la Figure 2.

Un programme dont les fichiers solutions ne respectent pas le format demandé n'est pas considéré valide.

5 Validation de la solution

Un *checker* est à votre disposition dans le dossier `checker` qui se trouve sur Moodle. Il est nommé `CheckerKEP.jar`. Pour utiliser le *checker*, vous devez mettre dans le même dossier :

- le *checker* ;
- les fichiers d'instance ;
- les fichiers de solution associés.

Vous devez impérativement respecter le nommage des fichiers et leur format. Le *checker* s'utilise en ouvrant une invite de commande. Pour tester la validité du fichier de solution nommé `nomInstance_sol.txt`, il suffit de taper la commande suivante : `java -jar CheckerKEP.jar nomInstance`. Pour tester la validité de tous les fichiers de solution (terminant par `_sol.txt`) présents dans le répertoire, il suffit de taper la commande suivante : `java -jar CheckerKEP.jar -all`.

6 Directives informatiques

6.1 Développement de l'application

Le programme doit être écrit dans le langage Java.

Pour le développement de votre programme, nous vous conseillons de le faire de manière incrémentale en suivant les étapes décrites ci-dessous.

6.1.1 Première version

Dans cette première version, il faut pouvoir :

- lire les fichiers d'instance (en format `txt`) ;
- réaliser les traitements métiers afin de fournir une solution réalisable (comme proposé dans la Section 3) ;
- écrire la solution selon le format spécifié dans un fichier `txt`.

Ici, l'objectif est de prendre en main le problème, et de valider que vous arrivez à produire des solutions réalisables (mêmes si elles ne sont pas du tout de bonne qualité). Par ailleurs, dans cette première version, il s'agit d'un programme Java sans interface. Mais cela vous permet d'entamer la réflexion sur le modèle objet à utiliser.

Cette première version doit aussi être l'occasion de vérifier que vous arrivez à produire un fichier au format `.jar` de votre application, afin que les enseignants puissent exécuter facilement votre application (voir Section 6.2).

6.1.2 Deuxième version

Dans cette deuxième version, il faut travailler sur la partie algorithmique pour améliorer la qualité des solutions proposées. Vous pouvez réutiliser des éléments algorithmiques vus dans les séances de TD, ou bien proposer d'autres approches, qu'il est préférable de valider au préalable avec les enseignants. Par ailleurs, faites attention à ce que le temps de résolution reste raisonnable (quelques minutes au maximum).

6.1.3 Version finale

Pour aboutir à la version finale, vous vous focaliserez sur les deux points suivants de manière équitable.

- Améliorer le traitement algorithmique afin d'être capable de fournir des solutions de très bonne qualité, tout en restant sur des temps de résolution raisonnables. N'hésitez pas à discuter avec les enseignants autour de la notion de "temps raisonnable". Cela peut varier selon la nature de l'algorithme proposé. Ainsi, un temps plus long de quelques minutes pourrait être accepté si la complexité de votre algorithme le justifie.
- Proposer une interface graphique qui permet notamment de visualiser les solutions. Il peut s'agir d'une interface graphique développée en Java avec Swing, ou bien d'une interface web. Dans le cas d'une interface web, on peut même envisager un autre programme qui vient se connecter directement sur votre programme, ou qui utilise les fichiers d'instance et de solution.

6.2 Fichier jar pour pouvoir exécuter votre application

Afin d'évaluer la qualité des solutions que vous obtenez, nous vous demandons de produire un fichier au format `.jar` de votre application. Ce fichier doit être nommé **Projet_Nom1_Nom2_Nom3_Nom4.jar** avec les noms des membres de l'équipe. Il doit fonctionner avec la ligne de commande suivante : `java -jar Projet_Nom1_Nom2_Nom3_Nom4.jar -inst nomFichierInstance -dSol nomRepertoireSolutions`, avec :

- `nomFichierInstance` : le nom du fichier d'instance sur lequel votre algorithme sera exécuté (ce nom peut contenir un chemin d'accès du fichier) ;
- `nomRepertoireSolutions` : le nom du répertoire dans lequel sera enregistré le fichier de solution produit par votre algorithme (il doit y avoir un seul fichier solution par instance).

Vous devez faire des tests pour vous assurer que l'exécution du *jar* se passe sans soucis.

Pour créer ce fichier, il faut dans votre code une classe principale (sur Netbeans, sur le nom du projet faire un clic-droit, puis *'Properties'*, puis *'Run'*, puis définir quelle est la classe principale de votre projet). Dans la méthode **main** de votre classe principale, vous devez lire le paramètre **args** qui contient dans un tableau tous les paramètres passés lors de l'appel. Ici, il y a aura donc 4 éléments dans le tableau **args** :

- `"-inst"` ;
- le nom du fichier d'instance ;
- `"-dSol"` ;
- le nom du répertoire de solutions.

Sur Netbeans, pour créer le *jar*, il faut cliquer sur *'Run'* dans la barre de menus, puis sélectionner *'Clean and Build Project'*. Le fichier `.jar` est alors disponible dans le répertoire `dist` du projet. Vous pouvez alors le renommer correctement.

Pour l'exécuter il faut ouvrir une invite de commande dans le répertoire où se trouve le fichier *jar*, puis exécuter la ligne de commande mentionnée précédemment.

7 Consignes générales

7.1 Plagia

Il n'est pas interdit d'utiliser des algorithmes *sur papier* provenant d'autres groupes, voire des codes trouvés sur internet, **à condition d'en citer les sources** et de ne constituer qu'une **partie minoritaire** du code global (inférieur à 10%). Toute infraction à cette règle sera considérée comme du plagia et sanctionnée comme tel, ce qui implique le risque de passer devant le conseil de discipline de Centrale Lille et d'en assumer les conséquences.

7.2 Travail en groupe

Le travail est à réaliser par groupes de 3 ou 4 élèves, que vous pouvez constituer comme vous le souhaitez. Il devra y avoir 5 groupes de 4 élèves et 2 groupes de 3 élèves. Au début de la séance du 28 mars 2022, les personnes non encore affectées à un groupe seront affectées d'office par les enseignants.

8 Rendu

8.1 Rendu intermédiaire

Vous déposez sur Moodle, dans la zone de dépôt **Dépôt intermédiaire du Projet 2021/2022** le fichier `.jar` de votre projet, nommé **Projet_Nom1_Nom2_Nom3_Nom4.jar** avec les noms des membres de l'équipe (- 2 points si ce nommage n'est pas respecté). Voir la Section 6.2 pour plus de renseignements.

La date limite de rendu est le vendredi **6 mai 2022 à 23h59** (-2 points par heure de retard). Le retard sera calculé de la façon suivante : $\lceil \frac{nbMinutesRetard}{60} \rceil$ où $nbMinutesRetard$ est le nombre de minutes de retard. Donc une minute de retard comptera pour une heure et donnera lieu à 2 points de pénalisation.

8.2 Rendu final

Vous déposez sur Moodle, dans la zone de dépôt **Dépôt final du Projet 2021/2022** une archive compressée de votre répertoire de projet. Cette archive doit être nommée **Projet_Nom1_Nom2_Nom3_Nom4.zip** avec les noms des membres de l'équipe (- 2 points si ce nommage n'est pas respecté). Cette archive doit comprendre tous vos fichiers sources, les fichiers de solutions que vous avez obtenus, ainsi que le fichier *jar* de votre application (voir Section 6.2).

La date limite de rendu est le vendredi **17 juin 2022 à 23h59** (-2 points par heure de retard). Le retard sera calculé de la façon suivante : $\lceil \frac{nbMinutesRetard}{60} \rceil$ où $nbMinutesRetard$ est le nombre de minutes de retard. Donc une minute de retard comptera pour une heure et donnera lieu à 2 points de pénalisation.

8.3 Soutenance finale

Par ailleurs, une soutenance aura lieu le mardi **21 juin 2022 après-midi, salle IG-302**. Cette soutenance comportera une présentation de votre projet durant laquelle vous présenterez l'architecture de votre application, l'algorithme de résolution, l'organisation de votre travail en groupe, les résultats obtenus et une démonstration de votre interface graphique. Il s'en suivra des questions posées par les enseignants.

9 Notation

Une note globale sera donnée sur la base du travail fourni par le groupe, mais une note individuelle sera ensuite déterminée pour chaque étudiant. Chaque membre du groupe doit **participer activement** au développement de l'application et connaître son fonctionnement général (même s'il n'a pas tout implémenté). Durant la soutenance, chaque membre du groupe est sensé être capable de répondre aux questions sur toutes les différentes composantes de l'application, même s'il n'a pas développé la partie du code associée.

Seuls les projets qui proposent les éléments des trois versions (décrites dans les Sections 6.1.1–6.1.3) peuvent aspirer à obtenir une note qui permet de valider le projet. Bien sûr, la qualité de la solution proposée sera aussi importante pour le calcul de la note. Il n'est pas suffisant de *faire* les choses, mais il faut veiller à ce qu'elles soient *bien* faites.

9.1 Critères de notation

Ce projet est évalué sur 20 points. Les points sont attribués en tenant en compte des aspects suivants :

- La qualité de votre code. Les aspects pris en compte sont par exemple :
  - le respect des principes de la programmation orientée objet ;
  - la mise en place des *design pattern* vus en cours ;
  - la qualité d'implémentation des algorithmes (utilisation des bonnes structures de données, attention portée à la complexité algorithmique des traitements) ;
  - la présentation du code (indentation correcte, méthodes courtes, respects des conventions Java dans les noms de classe, attributs, méthodes et variables, commentaires pertinents au format Javadoc) ;
  - la factorisation du code pour éviter les redondances ;
  - la présence et la qualité des tests.
- La qualité des solutions obtenues au rendu intermédiaire et au rendu final. Pour avoir des points sur ce critère vous devez vous assurez que votre fichier `.jar` (voir Section 6.2) soit capable de fournir une solution réalisable sur toutes les instances qui vous sont données.
- L'algorithmique. Les aspects pris en compte sont par exemple :
  - l'originalité des algorithmes proposés ;
  - la pertinence des algorithmes proposés par rapport au problème à résoudre.
- L'interface graphique. Les aspects pris en compte sont par exemple :
  - la clarté et l'exhaustivité des informations affichées ;
  - le nombre et la pertinence des fonctionnalités ;
  - l'expérience utilisateur.
- La soutenance finale. Les aspects pris en compte sont par exemple :
  - la clarté de la présentation ;
  - la qualité des supports ;
  - la présentation critique de résultats.
- L'avancement progressif de votre travail (vous devez avoir des choses à montrer à chaque séance et pas uniquement à la soutenance finale).

9.2 Coefficients individuels

Par ailleurs, chaque groupe doit proposer un coefficient par membre de son groupe, reflétant l'investissement de chacun des membres sur le projet. Ces coefficients seront ensuite multipliés par la note globale que le groupe a obtenue, ce qui donnera une note individuelle sur ce projet. La moyenne de ces coefficients doit être de 1, et la différence entre le plus grand et le plus petit coefficient doit être d'au moins 0,05 et d'au plus 0,4.

Ces coefficients doivent être **envoyés par mail aux enseignants**, avec tous les membres du groupe en copie, avant le **17 juin 2022 à 23h59**.

9.3 Participation active

Les enseignants se réservent la possibilité d'exclure de l'évaluation du travail d'un groupe les étudiants qui n'ont pas participé activement au développement du programme. Leur note sera automatiquement zéro.

References

[1] L. Pansart, *Algorithmes de chemin élémentaire : application aux échanges de reins*, thèse de doctorat, Université Grenoble Alpes, 2020.

[2] H. Cambazard, N. Catusse, L. Pansart, *Etude de cas – Echange de reins – Théorie des graphes, programmation linéaire, programmation dynamique*, Grenoble INP – Génie Industriel, 2ème année, 2022.

[3] Institute for Health Metrics and Evaluation. *Gbd compare: Global burden of disease study*, 2018.

[4] Transplant Observatory. *Global observation on donation and transplantation*, 2018.

[5] Péter Biró, Lisa Burnapp, Bernadette Haase, Aline Hemke, Rachel Johnson, Joris van de Klundert, and David Manlove. .*First handbook: Kidney exchange practices in europe. Technical report, European Network for Collaboration on Kidney Exchange Programmes*, 2017.

[6] UWHealth. *Longest kidney chain ever completed wraps up at uw hospital and clinics*S