Rapport de Stage de L3
Le Problème des Mariages Stables

Marguerite FLAMMARION

Du 8 au 26 juin 2015

Responsable de stage : Pierre COUCHENEY
Laboratoire PRiSM
Université de Versailles Saint-Quentin

J'exprime ma profonde gratitude aux directions des études de l'Université Paris-Sud et du Magistère de mathématiques qui m'ont permis de completer ma formation par un séjour d'apprentissage hors murs.

Je remercie vivement monsieur Jean-Michel Fourneau pour m'avoir accueillie au sein du laboratoire PRiSM et de l'équipe MAGMAT. Je remercie également de tout coeur Amira Choutri et Lise Rodier pour m'avoir accueillie dans leur bureau avec gentillesse et attention.

Toute ma reconnaissance va à Pierre Coucheney qui a bien voulu encadrer mon stage et m'aider dans mes travaux avec pertinence et gentillesse. Je tiens à lui exprimer ma profonde gratitude pour tout le temps qu'il m'a consacré et pour ses conseils avisés.

J'adresse enfin mes remerciements à Yann Strozecki et Franck Quessette pour leur présence à mon oral de fin de stage.

Préambule

Introduction

J'ai effectué mon stage d'apprentissage hors murs au Laboratoire PRiSM de l'Université de Versailles Saint-Quentin, pendant une période de trois semaines.

Ayant suivi les cours *Algorithmique et Complexité* puis *Informatique Théorique* pendant ma troisième année de Licence, je voulais effectuer mon stage en laboratoire d'informatique afin de mieux connaitre ce domaine. J'ai donc pris contact avec le directeur du Laboratoire PRiSM, Jean-Michel Fourneau, qui m'a redirigé vers la maître de conférence Pierre Coucheney. Celui-ci m'a alors proposé un stage autour du problème des mariages stables.

Le Laboratoire PRiSM

L'organisation

Le Laboratoire PRiSM est divisé en six équipes de recherche.
Les équipes MAGMAT (Models, Algorithms, and Games for Molecules Analysis and Telecommunications) et CRYPTO (Cryptologie et Sécurité de l'Information) travaillent aux étages 3 et 4 du bâtiment Descartes de la faculté de Versailles. Le directeur du laboratoire, Jean-Michel Fourneau est également responsable de l'équipe MAGMAT. Chaque équipe est dirigée par un directeur de recherche, il y a ensuite les chargés de recherche, des maitres de conférence, comme mon responsable de stage, Pierre Coucheney, des enseignants-chercheurs, médecins, post-doc et doctorants.
Pierre fait partie de l'équipe MAGMAT, qui traite essentiellement des algorithmes et de la théorie des jeux.

La vie au Laboratoire

Au laboratoire PRiSM, les chercheurs travaillent au troisième étage, tandisque les doctorants sont réunis au quatrième étage. Les chercheurs sont très interactifs, ils échangent sur beaucoup de sujets ; par exemple, tous les vendredis matin, un séminaire présenté par un membre de l'équipe MAGMAT ou de l'équipe CRYPTO permet aux chercheurs de rester au courant des avancées de leurs collègues. J'ai pu au cours de mon stage assister au séminaire de Jean-Michel Fourneau sur les *Dynamic False Trees*, et à un oral de fin de stage de M2 sur les *Simple Stochastic Games*. En ce qui concerne le rythme de travail, il est très libre, certains chercheurs commencent assez tôt le matin, vers 8h30 ou 9h, mais d'autres préfèrent arriver vers 11h et rester plus tard le soir. Ils peuvent également être souvent en déplacement pour collaborer avec d'autres organismes de recherche.
Finalement, j'ai observé une ambiance très conviviale, et j'ai eu l'impression que les chercheurs du laboratoire étaient très complémentaires dans leurs domaines de compétences.

Travail effectué

Semaine 1

La première semaine, j'ai découvert le laboratoire et le problème des mariages stables sur lequel j'allais travailler pendant tout mon stage. Dans un premier temps, je me suis essentiellement documentée sur ce problème, grâce à des articles fournis par Pierre [1] [2], mais aussi en effectuant des recherches sur internet [3]. J'ai appris à réutiliser le langage CamlLight que j'avais étudié en prépa, ce qui m'a permis de coder l'algorithme de Gale-Shapley. Cet algorithme m'a ensuite été utile pour réaliser de nombreuses simulations. J'ai également rédigé quelques preuves de propositions portant sur cet algorithme.

Semaine 2

La deuxième semaine, j'ai travaillé avec Pierre sur le nombre moyen de propositions dans l'algorithme de Gale-Shapley, et sur la complexité au pire. C'était une partie un peu difficile, et j'ai dû faire appel à mes souvenirs de cours de Combinatoire et de Probabilités. J'ai également effectué beaucoup de tests à l'aide de CamlLight. Nous avons commencé à nous intéresser à la notion de couplage stable équitable ainsi qu'au problème des mutations.

Semaine 3

La troisième et dernière semaine, en m'appuyant sur un livre de KNUTH [4], je suis parvenue à coder l'algorithme de Selkow permetant d'obtenir un couplage stable équitable. J'ai aussi étudié sa complexité. Enfin, j'ai achevé la rédaction de tous les résultats obtenus au cours de ces trois semaines de stage. Le dernier jour, j'ai pu présenter à l'oral le travail effectué pendant mon stage devant quatre personnes et répondre à leur questions. Enfin, j'ai pu discuter du métier de chercheur avec plusieurs personnes.

Conclusion

Ce stage d'apprentissage hors mur m'a permis de mieux connaitre le métier de chercheur ainsi que la vie en laboratoire. J'ai pu partager le bureau de deux doctorantes, à l'étage des "thésards", et découvrir ainsi ce milieu accueillant.

J'ai également pu approfondir mes connaissances en informatique théorique à travers le problème des mariages stables. J'ai constaté à quel point différents problèmes considérés comme classiques sont connectés entre eux (en étudiant par exemple le problème de la collection de coupons). J'ai pu travailler sur le problème des mariages stables en confrontant mes calculs aux résultats de mes simulations numériques, et constater que, souvent, l'ordinateur avait plus raison que moi.
J'ai d'ailleurs eu beaucoup de plaisir à faire de l'informatique "à temps plein", matière que j'avais découvert assez tardivement au cours de ma scolarité puisque je n'avais commencé qu'en milieu de première année de classe préparatoire.

Finalement, j'ai pu voir au cours de mon stage l'utilité des mathématiques pour traiter de problèmes informatiques. Certes, la rigueur des raisonnements m'a été utile, mais au-dela j'ai pu utiliser de nombreuses connaissances acquises en cours de *Combinatoire Algébrique* et de *Théorie de la mesure et Probabilités*, notamment pour calculer des complexités non triviales.

Pour terminer, je suis très heureuse d'avoir eu la chance d'effectuer ce stage au laboratoire PRiSM. Le travail que j'y ai effectué avec l'aide active de Pierre Coucheney m'a vraiment plu et j'ai été très enthousiasmée par le métier de chercheur.

Table des matières

Introduction

Le problème des mariages stables est le suivant : imaginons une communauté de n hommes et n femmes en âge de se marier. Les hommes et les femmes n'ont pas forcément tous les mêmes préférences, et il est difficile de satisfaire tout le monde en formant n couples arbitraires. Néanmoins, on peut chercher à former des couples de telle sorte à ce que personne n'ait envie d'échanger ou de tromper son conjoint. On appelera ce couplage un mariage stable.

Dans ce rapport, on va étudier étudier et prouver quelques résultats sur le problème des mariages stables, et s'intéresser à quelques applications.

1 Le problème des mariages stables

1.1 Définitions

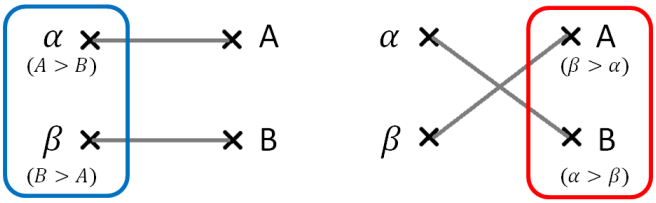
Définition 1.1. Pour le problème des mariages dans une communauté de n hommes (l'ensemble M) et n femmes (l'ensemble W), un *couplage* est une application injective de M dans W.
Pour le problème des écoles à concours multiples pour un ensemble E de p écoles et un ensemble C de q candidats, une *affectation* est une application de C dans E.

Les définitions qui suivent seront valables pour les couplages et les affectations, indifféremment.

Définition 1.2. Un couplage est dit *instable* s'il existe deux couples $(\alpha, A), (\beta, B) \in M \times W$ tels que β préfère A à B et A préfère β à α.
S'il n'existe aucune paire de couples de la sorte, le couplage est dit *stable*.

Définition 1.3. On dit qu'une femme est *possible* pour un homme s'il existe au moins un couplage stable pour lequel cette femme et cet homme sont en couple. Sinon, on dit que cette femme est *impossible* pour cet homme.
De même, on définit un homme possible et impossible pour une femme.

On peut monter avec un exemple simple qu'il n'existe pas toujours une solution satisfaisant à la fois les hommes et les femmes : on a deux hommes α et β et deux femmes A et B tels que α préfère A, β préfère B, A préfère β et B préfère α.



Dans cette situation, une solution favorisera toujours les hommes au dépend des femmes ou les femmes au dépend des hommes.

Définition 1.4. Un couplage stable est dit *optimal pour les hommes* si chaque homme est en couple avec une femme qu'il préfère à toutes les autres femmes possibles.
De même, on définit un couplage stable optimal pour les femmes.

Définition 1.5. Un couplage stable est dit *le moins optimal* pour les hommes si chaque homme est en couple avec une femme à qui il préfère toutes les autres femmes possibles. De même, on définit un couplage stable le moins optimal pour les femmes.

Il n'est pas évident qu'il existe toujours un couplage stable. En revanche, s'il existe un couplage stable, alors il existe un couplage stable optimal et il est unique. De même pour un couplage stable le moins optimal.

1.2 Présentation du problème

Le problème du mariage est le suivant : dans une communauté de n hommes et n femmes, on cherche à marier tous les membres en tenant compte de leurs préférences. On veut donc obtenir un couplage stable à partir des hommes et des femmes de cette communauté.

Par exemple, pour $n = 3$, on peut avoir les "matrice des préférences" suivantes :

$$M1 = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \\ 2 & 3 & 1 \end{pmatrix}$$

$$M2 = \begin{pmatrix} 3 & 1 & 2 \\ 2 & 3 & 1 \\ 1 & 2 & 3 \end{pmatrix}$$

$M1_{i,j}$ donne le rang de la femme j pour l'homme i. Similairement, $M2_{i,j}$ donne le rang de l'homme j pour la femme i.
Pour $n = 3$, il existe $3! = 6$ couplages différents.
Pour cet exemple, on observe que sur ces six possibilités de couplages, seulement trois sont stables : $((1,1),(2,2),(3,3))$, $((1,2),(2,3),(3,1))$ et $((1,3),(2,1),(3,2))$.
Parmi ces couplages stables, le couplage $((1,1),(2,2),(3,3))$ est optimal pour les hommes et le couplage $((1,3),(2,1),(3,2))$ est optimal pour les femmes.

2 Stabilité des mariages

Théorème 2.1. Pour un problème de mariages donné, il existe toujours un couplage stable.

Preuve 2.1. On va donner une preuve constructive de l'existence d'un couplage stable. Il s'agit d'un algorithme itératif. Cet algorithme a été présenté pour la première fois en 1962 par D.Gale et L.S. Shapley [1].

2.1 L'algorithme de Gale-Shapley

On place tous les hommes dans une file d'attente. A chaque tour, le premier homme de la file se propose à la femme qu'il préfère parmi celles à qui il ne s'est pas encore proposé. La femmes lui "non" si elle est déjà affectée à un homme qu'elle préfère, et "peut-être" sinon. Si un homme est rejeté, alors il va se placer à la fin de la file d'attente. L'algorithme termine lorsque la file d'attente est vide, c'est-à-dire lorsque chaque homme est affecté à une femme [3].
On remarque qu'une fois qu'une femme a un partenaire, elle n'est jamais libérée, elle ne peut être que réaffectée à un homme qu'elle préfère ; de plus, un homme qui n'a pas de partenaire effectue des propositions jusqu'à ce qu'il en trouve une, ce qui prouve que l'algorithme termine.

Algorithme

On suppose que $M1$ et $M2$ sont deux matrices carrées de dimension n contenant respectivement les préférences des hommes et les préférences des femmes de la communauté. Ainsi $M1(i, j)$ contient donc le classement de la femme j pour l'homme i. La fonction $Pile(M)$ donne un tableau contenant des listes de préférence. Ainsi $Pile(M)(i)$ contient la liste des femmes selon l'ordre de préférence de l'homme i.

```
MariageStable (M1,M2)
   n = taille(M1)
   Epouse = tableau de taille n initialise a 0
   Mari = tableau de taille n initialise a 0
   Liste = Pile (M1)
   HommesLibres = file qui contient 1,...,n
      tant que HommesLibres <> []
         homme = Premier (HommesLibres)
         Depile HommesLibres
         femme = Premier (Liste(homme))
         Depile(Liste(homme))
            si Mari(femme) = 0
            alors Mari(femme) = homme
                  Epouse(homme) = femme
            sinon si M2(femme,homme)<M2(femme,Mari(femme))
                  alors Enfile (Mari(femme),HommesLibres)
                        Mari(femme) = homme
                        Epouse(homme) = femme
                  fin si
            fin si
      fin tant que
   rendre Mari ;
```

Explication de l'algorithme

$Mari$ est un tableau de taille n qui contient les liste d'attente des hommes. $Mari(f)$ contient donc l'homme préféré par la femme f parmi tous ceux qui lui ont été proposés.
A la fin de l'exécution, $Mari$ contient le choix final de chaque femme (c'est-à-dire l'homme qu'elle a préféré parmi tous les hommes qu'on lui a proposé). Les couples formés sont de la forme $(Mari(f), f)$ pour chaque femme f.

Preuve de l'algorithme

On raisonne par l'absurde. Supposons que le couplage $(Mari(i), i)$ soit instable. Par définition, il existe deux femmes i et j, telles que $Mari(i)$ préfère j à i et j préfère $Mari(i)$ à $Mari(j)$. Cela veut dire que $M1(Mari(i), j) < M1(Mari(i), i)$. Donc $Mari(i)$ a été proposé à j, sans quoi il n'aurait pas été proposé à i. Or $Mari(j)$ est l'homme que j a préféré à tous les autres hommes qui lui ont été proposés, y compris $Mari(i)$. Donc j préfère $Mari(j)$ à $Mari(i)$, ce qui est absurde. On en déduit que le couplage $(Mari(i), i)$ construit par l'algorithme est stable.

□

Proposition 2.1. L'algorithme de Gale-Shapley a une complexité en $O(n^2)$.

Preuve 2.2. A chaque passage de la boucle, un homme fait une demande à une nouvelle femme. Il n'y a donc que n^2 demandes possibles.

□

2.2 Propriétés

Proposition 2.2. L'algorithme de Gale-Shapley contruit l'unique couplage stable optimal pour les hommes.

Preuve 2.3. On raisonne par l'absurde. Supposons que le couplage obtenu par l'algorithme ne soit pas obtimal pour les hommes. Alors il existe au moins un homme qui s'est fait rejeter par une femme possible au cours de l'algorithme. Soit M le premier homme rejeté par une femme possible W. On note M' l'homme pour lequel W a rejeté M. On sait donc que W préfère M' à M.
De plus, comme W est possible pour M, il existe un couplage stable pour lequel M et W sont en couple, et M' et W' sont en couple. Donc W' est possible pour M'. Or M est le premier homme rejeté par une femme possible, donc M' n'a pas été rejeté par W', donc il ne l'a pas encore rencontrée. Comme il rencontre les femmes par ordre de préférence, on en déduit que M' préfère W à W'.
On en déduit que le couplage pour lequel M et W sont en couple est instable, donc W est impossible pour M, d'où la contradiction.
On en déduit que le couplage obtenu par l'algorithme est obtimal pour les hommes.

□

Proposition 2.3. L'algorithme de Gale-Shapley contruit l'unique couplage stable le moins optimal pour les femmes.

Preuve 2.4. On raisonne par l'absurde. Supposons que le couplage obtenu par l'algorithme ne soit pas le moins optimal pour les femmes. Soit W une femme en couple avec M et $M*$ un homme en couple avec $W*$, tels qu'il existe un couplage stable pour lequel W est en couple avec $M*$, et tels que W préfère M à $M*$. Or le couplage construit par l'algorithme est optimal, donc, par la proposition 1.2, M préfère W à toutes les autres femmes possibles. Donc le couplage pour lequel W est en couple avec $M*$ est instable, d'où la contradiction.
On en déduit que le couplage obtenu par l'algorithme est le moins optimal pour les femmes.

□

3 Complexité de l'algorithme

3.1 Nombre moyen de proposition

On va à présent étudier le nombre moyen N de propositions faites par les hommes au cours de l'algorithme de Gale-Shapley. Pour cela, on va commencer par approcher N par le calcul du nombre de propositions pour un grand nombre de matrices générées aléatoirement. Puis on va simplifier le problème afin de se ramener à un problème connu : le *problème de la collection de coupons*.

Estimation

Dans l'algorithme de Gale-Shapley (donné en annexe), on rajoute un compteur qu'on augment de 1 à chaque nouvelle proposition faite par un homme. On somme ensuite ces compteurs pour un grand nombre (ici, 1000) de matrices de préférences générées aléatoirement, et on divise le résultat par le nombre de matrices pour obtenir une moyenne :

```
let moyenne n =
    let cpt = ref 0 in
    let M = rand_mat n in
        for i = 1 to 1000 do
            cpt := !cpt + (MS (rand_mat n) M)
        done ;
    (float_of_int !cpt)/. 1000. ;;
```

On notera que la matrice de préférence des femmes est donnée, seule celle des hommes change afin d'établir la moyenne.
Les résultats numériques permettent de trouver l'ordre de grandeur de la complexité de l'algorithme. La figure 1 montre que le nombre de propositions est très proche de $n\ln(n)$.

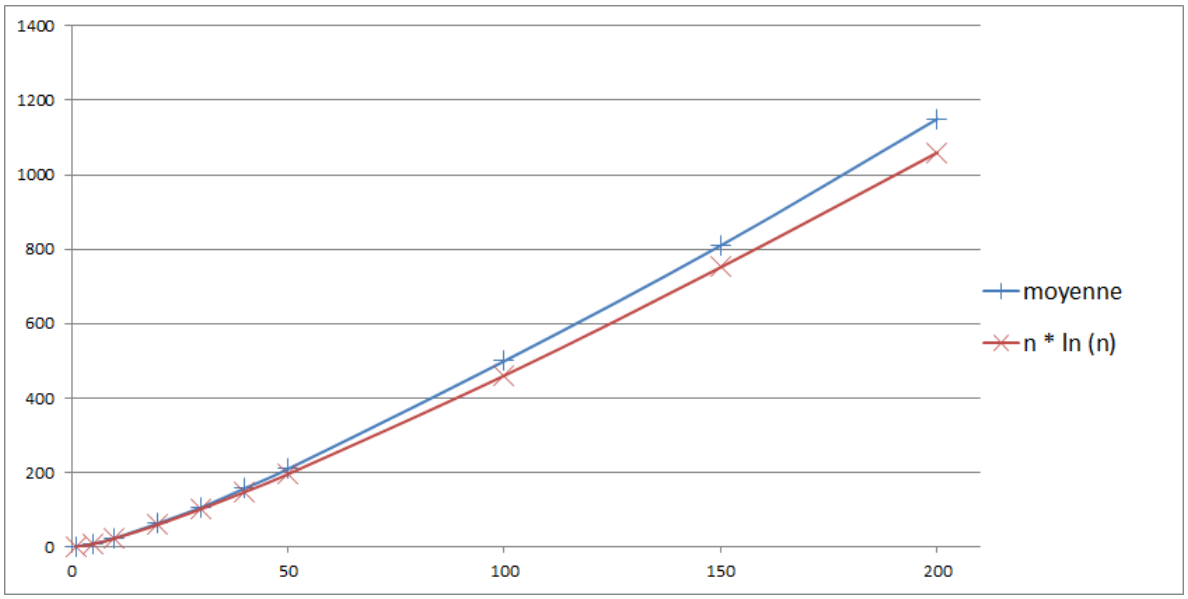


FIGURE 1 – Nombre moyen de propositions en fonction du nombre d'hommes n comparé à $n\ln(n)$.

Simplification du problème

On suppose que les hommes sont *amnésiques*. Ainsi, ils ne se rappellent plus des femmes à qui ils se sont déjà proposés. A chaque itération, ils ont donc autant de chance d'aller voir chaque femme. Le couplage stable est atteint quand toutes les femmes ont eu au moins une proposition.
Le problème des mariages stables simplifié (sous l'hypothèse que les hommes sont amnésiques) est donc équivalent au problème de la collection de coupons.

Collection de coupons

Le problème de collection de coupons est le suivant : on suppose que dans une collection, il existe n coupons différents, et qu'on obtient un coupon à chaque fois qu'on achète une boite de céréales, avec équiprobabilité des coupons.
Soit $N1$ le nombre de boites qu'on doit acheter en moyenne pour obtenir les n coupons. On note p_k la probabilité qu'il faille acheter exactement k boites pour obtenir les n coupons. On pose :

$$q_k = p_k + p_{k+1} + ...$$

q_k est la probabilité qu'au moins k boites soient nécessaires. On remarque que :

$$q_1 = 1 \text{ et } p_k = q_k - q_{k+1}$$

Le nombre moyen de boites utilisées est donc :

$$N1 = p_1 + 2p_2 + 3p_3 + ... = q_1 + q_2 + q_3 + ...$$

On décompose le problème en m étapes successives, m correspondant au nombre de coupons différents déjà en notre possession. Pour l'étape m, on a donc :

$$q_1 = 1$$
$$q_2 = \frac{m}{n}$$
$$q_3 = \left(\frac{m}{n}\right)^2$$
$$...$$
$$q_1 + q_2 + q_3 + ... = \frac{n}{n-m}$$

Le nombre moyen de boites à acheter pour obtenir les n coupons est donc :

$$\frac{n}{n-0} + \frac{n}{n-1} + ... + \frac{n}{n-(n-1)} = nH_n$$

où H_n est la somme des n premiers termes de la *série harmonique* ($H_n = \ln(n) + \gamma + o(1)$).

Conclusion

Le nombre moyen N de propositions faites par les hommes au cours de l'algorithme est donc *majoré* par le nombre moyen N' de propositions faites par les hommes amnésiques au cours de l'algorithme. Or N' est également le nombre moyen de boites qu'il faut acheter pour obtenir les n coupons dans le problème du coupon collector. Donc $N' \sim n \times (\ln(n) + \gamma)$. On en déduit que :

$$N \leq n \times (\ln(n) + \gamma)$$

3.2 Minoration de la complexité

Dans cette partie, on va étudier une minoration du nombre moyen de propositions.
Pour ce faire, on ne suppose plus à présent que les hommes sont amnésiques, mais au contraire qu'ils ont une mémoire leur permettant de se souvenir des k dernières femmes qu'ils ont vues. On note $T(m, n, k)$ le temps qu'il faut à un homme en moyenne pour trouver une nouvelle femme parmi n quand il en a déjà vues m. Alors on a :

$$T(m,n,k) = \left(1 - \frac{m}{n}\right) \times 1 + \frac{m}{n} \times (T(m-1, n-1, k-1) + 1) = 1 + \frac{m}{n} \times T(m-1, n-1, k-1)$$

$$T(m,n,0) = \frac{n}{n-m}$$

On en déduit que :

$$T(m,n,k) = 1 + \frac{m}{n}\left(1 + \frac{m-1}{n-1}\left(1 + ...\left(1 + \frac{n-k}{n-m}\right)\right)\right)$$

$$T(m,n,k) = \sum_{i=0}^{k-1} \frac{m!(n-i)!}{n!(m-i)!} + \frac{m!(n-k)!}{n!(m-k)!}\frac{n-k}{n-m}$$

Pour obtenir un minorant de N, on se place dans le cas où $k = m$, c'est-à-dire dans le cas où la mémoire correspond au nombre de femmes déjà vues, ce qui est le cas dans l'algorithme de Gale-Shapley. On a alors :

$$\begin{aligned}
T(m,n,k) &= \sum_{i=0}^{m} \frac{m!(n-i)!}{n!(m-i)!} \\
&= \sum_{i=0}^{m} \frac{m!(n-i)!(n-m)!}{n!(m-i)!(n-m)!} \\
&= \sum_{i=0}^{m} \frac{\binom{n-i}{n-m}}{\binom{n}{m}} \\
&= \frac{1}{\binom{n}{m}} \sum_{i=0}^{m} \binom{n-i}{n-m} \\
&= \frac{\binom{n+1}{n-m+1}}{\binom{n}{m}} \\
&= \frac{n+1}{n-m+1}
\end{aligned}$$

On peut alors sommer sur toutes les étapes m pour obtenir la moyenne :

$$\begin{aligned}
N &\geq \sum_{m=0}^{n-1} \frac{n+1}{n-m+1} \\
&\geq (n+1) \sum_{m=2}^{n+1} \frac{1}{m} \\
&\geq (n+1)H_{n+1} - (n+1)
\end{aligned}$$

Finalement :

$$N \geq (n+1)H_n - n$$

Conclusion

On a donc obtenu un encadrement de N :

$$(n+1)H_n - n \;\leq\; N \;\leq\; n \times H_n$$

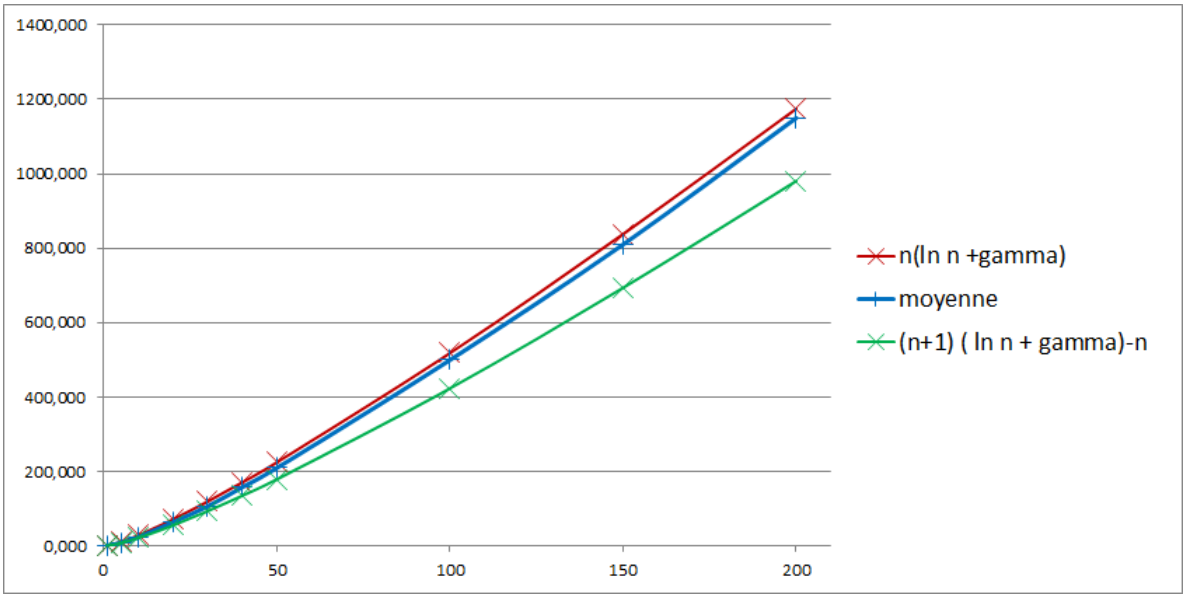


FIGURE 2 – Encadrement du nombre moyen de propositions.

Une meilleure approximation ?

On note M la variable aléatoire correspondant au nombre moyen de femmes après n étapes. On a :

$$E[T_{m\to m+1}|M \leq m] = \frac{n-1}{n-m}$$

$$E[T_{m\to n}|M = m] = \sum_{i=m}^{n-1} \frac{n-1}{n-i} = (n-1=) \sum_{i=1}^{n-m} \frac{1}{i} = (n-1)H_{n-m}$$

Si U et V sont complémentaires, alors on a :

$$\#\{V = m\} = \#\{U = n-m\} = \binom{n}{m} \sum_{k=0}^{m} (-1)^k \binom{m}{k} (m-k)^n$$

D'où :

$$\begin{aligned} E[T_{0\to n}] &= \sum_{m=1}^{n} P[M = m] E[T_{m\to n}|M = m] \\ &= \sum_{m=1}^{n} \frac{\#\{U = n-m\}}{n^n} (n-1)H_{n-m} \\ &= \sum_{m=1}^{n} \frac{(n-1)}{n^n} H_{n-m} \binom{n}{m} \sum_{k=0}^{m} (-1)^k \binom{m}{k} (m-k)^n \\ E[T_{0\to n}] &= \frac{(n-1)}{n^n} \sum_{m=1}^{n} H_{n-m} \binom{n}{m} \sum_{k=0}^{m} (-1)^k \binom{m}{k} (m-k)^n \end{aligned} \tag{1}$$

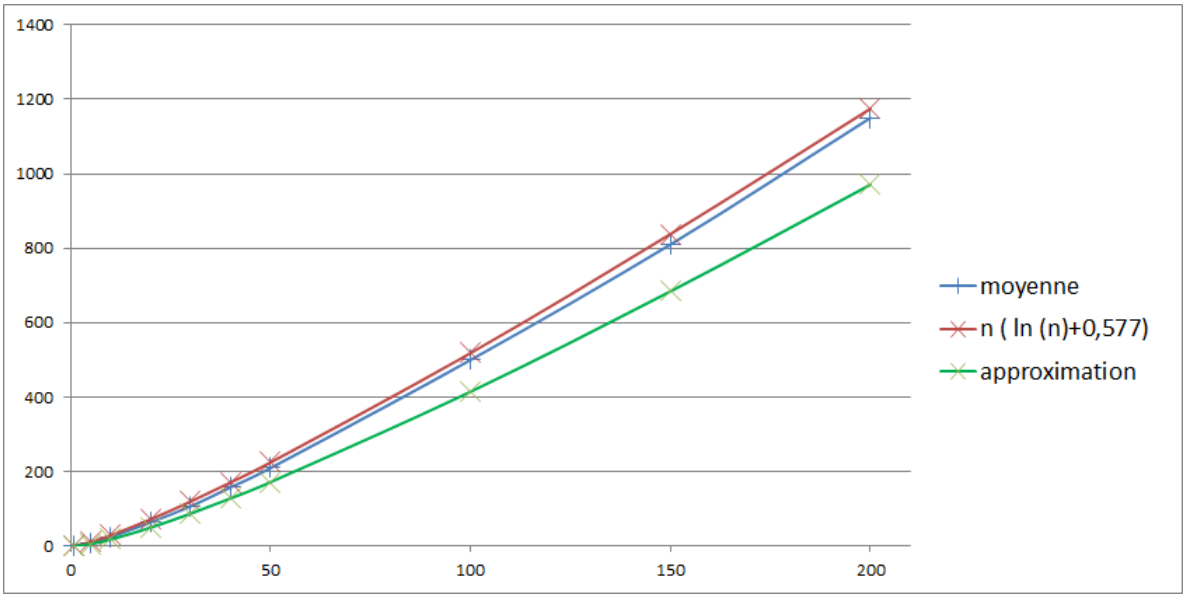


FIGURE 3 – Approximation du nombre moyen de propositions à l'aide de la formule (1).

La figure 3 permet de comparer cette approximation à celle qui a été faite avec le problème de la collection de coupons. On remarque alors que la collection de coupons était une très bonne approximation.

3.3 Complexité dans le pire des cas

On peut facilement majorer la complexité de l'algorithme par $n^2 - n + 1$.

Proposition 3.1. La complexité N de l'aglorithme de Gale-Shapley est majorée par $n^2 - n + 1$.

Preuve 3.1. L'algorithme termine lorsque toutes les femmes ont été demandées au moins une fois. Chaque homme peut se proposer à au plus $(n-1)$ femmes avant que l'algorithme termine par la proposition à une n-ième femme. Ce qui nous donne donc :

$$N \leq n(n-1) + 1$$

$$N \leq n^2 - n + 1$$

□

Il reste à montrer qu'il existe toujours un *pire des cas* où cette majoration de la complexité est atteinte.
Pour cela, on va exhiber deux matrices de préférences de taille n et on va montrer que l'algorithme se fait en $n^2 - n + 1$ propositions.

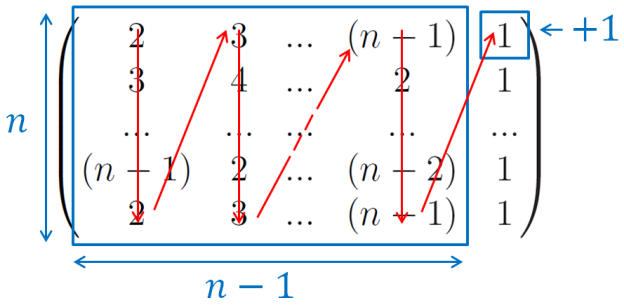
Proposition 3.2. On se place dans la convention où $M[i][j]$ est la j-ième préférence de l'homme i. On définit les matrices de préférences suivantes :

$$M1 = \begin{pmatrix} 2 & 3 & \dots & (n-1) & 1 \\ 3 & 4 & \dots & 2 & 1 \\ \dots & \dots & \dots & \dots & \dots \\ (n-1) & 2 & \dots & (n-2) & 1 \\ 2 & 3 & \dots & (n-1) & 1 \end{pmatrix}$$

$$M2 = \begin{pmatrix} & A & R & B & I & T & R & A & I & R & E & \\ 2 & 3 & \dots & & n & & & & & & 1 \\ \dots & \dots & \dots & & \dots & & & & & & 2 \\ \dots & \dots & \dots & & \dots & & & & & & \dots \\ n & 1 & \dots & & (n-2) & & & & & & (n-1) \end{pmatrix}$$

Alors l'algorithme de Gale et Shapley sur $M1$ et $M2$ se termine en exactement n^2-n+1 propositions.

Preuve 3.2. L'algorithme se déroule de la manière suivante : les hommes se proposent aux femmes selon leur ordre de préférence, à chaque tour chaque femme a un nouveau prétendant qu'elle préfère à l'homme à qui elle a dit peut-être (sauf 1). On parcourt ainsi la matrice $M1$ colonne par colonne.



L'algorithme se termine donc lorsque qu'un homme se proposera à 1 pour la première fois, c'est-à-dire au bout de $n(n-1)+1 = n^2-n+1$ propositions.

□

Ainsi la complextité dans le pire des cas est bien :

$$\mathscr{C}_{\text{pire}} = \mathbf{n^2 - n + 1}$$

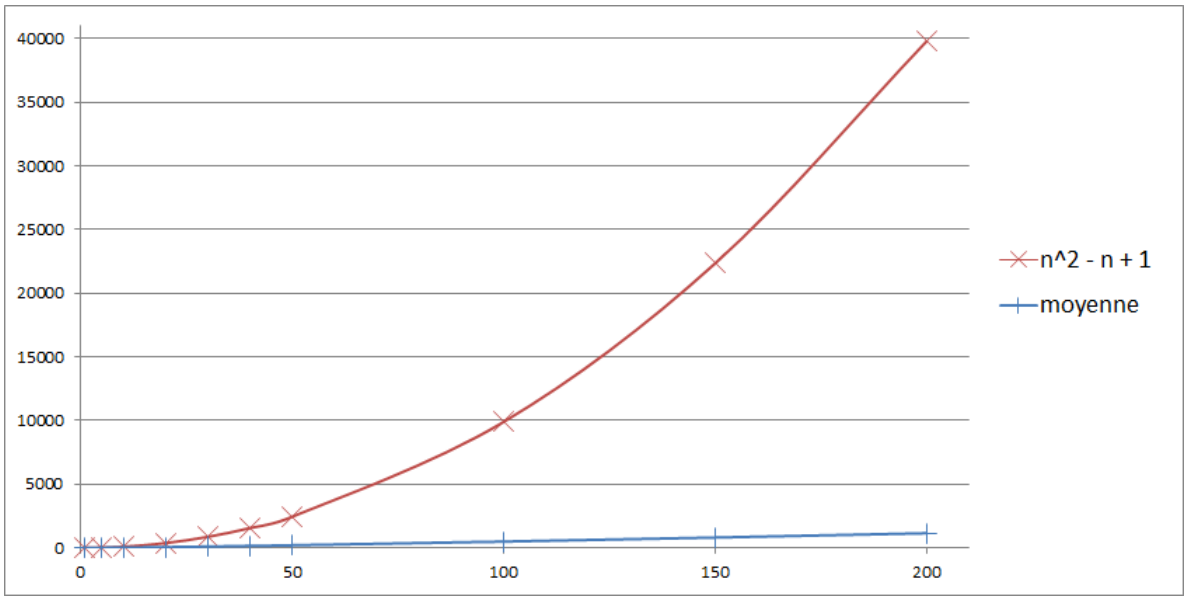


FIGURE 4 – Nombre de propositions dans le pire des cas comparé au nombre moyen de propositions.

4 Mariages équitables

4.1 Couplage stable équitable

On a vu que l'algorithme de Gale-Shapley permet de trouver ou bien un couplage stable optimal pour un des deux sexes, mais le moins optimal pour l'autre. Une telle injustice à l'heure de notre société actuelle ne manquera pas de faire frémir notre lecteur. On va donc essayer d'obtenir un couplage stable qui soit *équitable*, c'est-à-dire qui ne favorise ni les hommes, ni les femmes.

Définition 4.1. On appelle *regret* d'un homme le rang, parmi ses préférences, de la femme avec qui il est en couple.
On définit de même le regret d'une femme.
On appelle *regret maximal* le maximum des regrets de tous les membres de la communauté.

Définition 4.2. Parmi tous les couplages stables d'une situation donnée, on appelle couplage stable *équitable* un couplage qui minimise le regret maximal.

4.2 Algorithme de Stan Selkow

L'algorithme de Stan Selkow présenté par Knuth [4] permet d'obtenir un couplage stable équitable.

Principe de l'algorithme

Les solutions optimales de l'algorithme de Gale-Shapley donnent des bornes inférieures et supérieures pour le regret de tous les individus. On constuit donc à chaque itération un couplage stable qui diminue de 1 le regret d'une des personnes ayant un regret maximal. L'algorithme termine lorque toutes les bornes inférieures et supérieures sont égales.

```
MariageStableEquitable (M1,M2)
  (Hommes,Femmes) = Initialiser (M1,M2)
    tant que il existe homme tq BorneInf<>BorneSup faire
      BorneInf(hommes),BorneSup(femmes) <- MariageOpt (M1,M2,Hommes)
      BorneInf(femmes),BorneSup(hommes) <- MariageOpt (M1,M2,Femmes)
        Chercher homme ou femme tq BorneSup maximal
        (parmi ceux tq BorneInf<>BorneSup )
        Si c'est un homme faire : BorneSup(h)-1
                                  BorneInf(f)+1
        Sinon faire :             BorneSup(f)-1
                                  BorneInf(h)+1
    Construire Epouse a partir de BorneInf(hommes)
  Rendre Epouse ;;
```

Propriétés

Proposition 4.1. L'algorithme de Selkow construit un couplage stable.

Proposition 4.2. L'algorithme de Selkow minimise le regret maximal.

On calcule le regret maximal moyen pour différentes valeurs de n pour l'algorithme de Gale-Shapley et pour celui de Selkow.

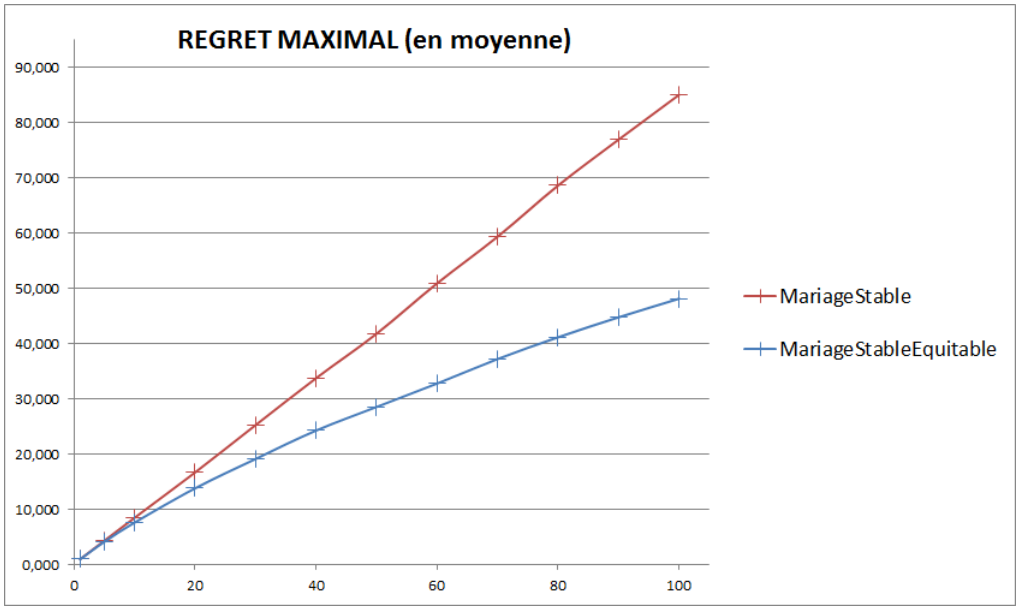


FIGURE 5 – Regret maximal pour l'algorithme de Gale-Shapley, et pour l'algorithme de Selkow.

On observe sur la figure 5 que l'algorithme de Selkow est bien plus équitable que celui de Gale-Shaley.

Complexité

La complexité de l'algorithme de Selkow est polynomiale en nombre de couplages.

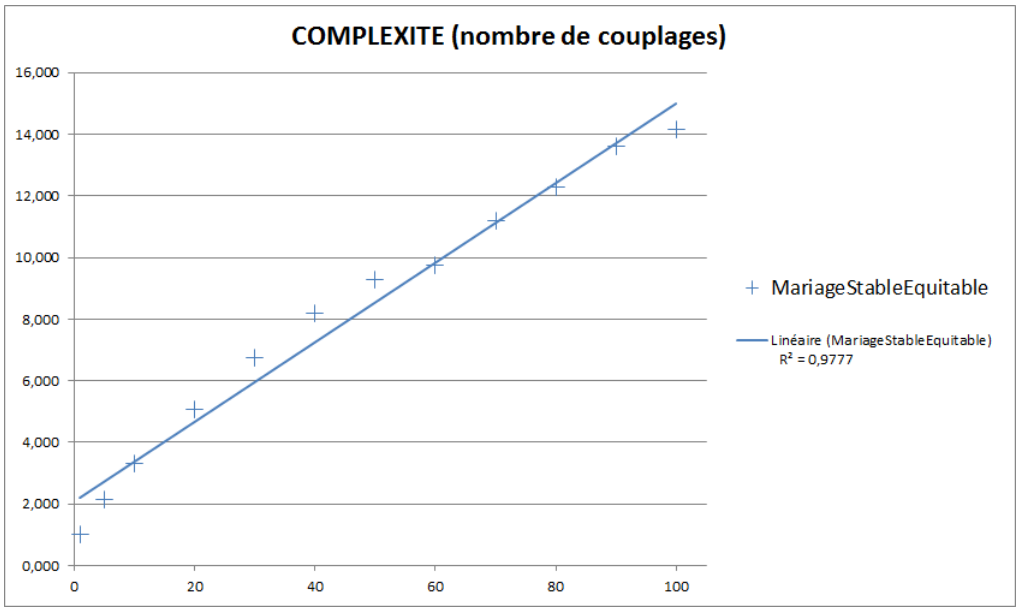


FIGURE 6 – Nombre moyen de couplages lors de l'algorithme de Selkow en fonction de n.

Exemple

Pour l'implémentation, se référer à l'annexe 2.

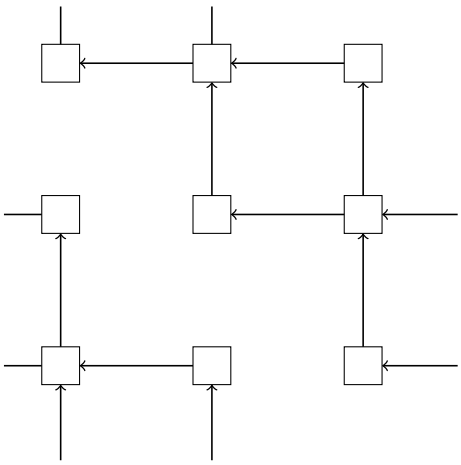
5 Graphes-mariages

On va s'intéresser à une nouvelle approche du problème des mariages stables impliquant des *graphes*, présentée par Balinski et Ratier [2].
On note $M = \{m_1, m_2, ..., m_k\}$ l'ensemble des hommes et $W = \{w_1, w_2, ..., w_k\}$ l'ensemble des femmes.

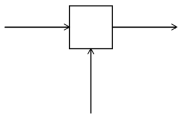
5.1 Définitions et notations

Définition 5.1. On construit le *graphe-mariage* Γ de la manière suivante :
- les sommets de Γ sont les paires $(m, w) \in M \times W$ telles que m est acceptable pour w et w pour m,
- les arrêtes orientées de Γ sont les arc horizontaux $\{(m, w_i), (m, w_j)\}$ tels que m préfère w_j à w_i, et les arcs verticaux $\{(m_i, w), (m_j, w)\}$ tels que w préfère m_j à m_i.

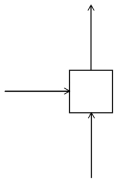
Par exemple, avec la situation décrite dans l'article précédent, on obtient le graphe suivant :



Définition 5.2. On appelle *meilleur noeud* pour une femme un noeud de la forme :



On appelle *meilleur noeud* pour un homme un noeud de la forme :



Définition 5.3. Deux graphes-mariages sont dits *équivalents* s'ils admettent exactement les mêmes mariages stables.

Proposition 5.1. On simplifie Γ en supprimant certains noeuds (ceux qui correspondent aux mariages impossibles).
Alors le graphe Γ' ainsi obtenu est équivalent à Γ. On dit que c'est un graphe-mariage *libre*.

Preuve 5.1. On remarque que simplifier un graphe revient à enlever tous les mariages impossibles dans la liste des propositions de chaque individu. Ainsi, le graphe Γ' est bien équivalent à Γ.

□

Proposition 5.2. L'ensemble des meilleurs noeuds pour les femmes d'un graphe-mariage libre est stable.

Théorème 5.1. Pour tout problème de mariages, il existe au moins une affectation stable. De plus, on peut en trouver une en $O(n^2)$.

On ne donnera pas la preuve de ce théorème, on remarque juste qu'elle est constructive puisqu'elle s'appuie sur un algorithme de réduction s'effectuant en $O(n^2)$. Pour une preuve complète, se référer à l'article de Balinski et Ratier [2].

5.2 Stratégies

Nous avons tous déjà entendu parler de *stratégie* à adopter pour classer ses voeux d'orientation afin d'obtenir de meilleurs résultats. Dans le cas du problème des mariage, on peut imaginer que les hommes ou les femmes adoptent la stratégie suivante : ils *mentent* sur leurs préférences afin d'obtenir un meilleur couplage (pour eux-même). On va voir dans quels cas cette stratégie est gagnante et dans quels cas il ne peut pas y avoir de stratégie.

Qui a intérêt à mentir ?

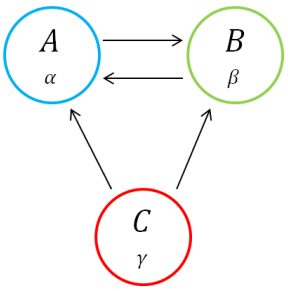
On peut montrer que les hommes n'ont jamais intérêt à mentir dans l'ordre de leurs préférences. En effet, avec l'algorithme de Gale-Shapley, le couplage est optimal pour eux. Ils sont donc affectés à la meilleure femme possible, et mentir ne permettra pas de rendre posssible des femmes impossibles.
Par contre, certaines femmes peuvent avoir intérêt à mentir. En controllant l'avancée des hommes dans leur liste de propositions, elles peuvent réussir à être affectées à des hommes qui ne soit pas les moins optimaus pour elles (par exemple en rendant impossible les hommes les moins optimaux). Mais quelques exemples simples montrent que les femmes n'ont pas toujours intérêt à mentir : certaines situations ne sont pas *améliorables*, même si les femmes n'ont pas leur premier choix.

On voit alors l'intérêt de l'algorithme de Selkow qui permet de minimiser le regret maximal.
Quant à l'algorithme de Gale-Shapley, il est bien adapté pour l'affectation des candidats à des écoles, car la meilleure stratégie pour les candidats est tout simplement d'ordonner les écoles selon leur réelle préférence.

6 Le problème des mutations

6.1 Présentation

Le problème des mutations est le suivant : on a trois enseignants α, β et γ, qui sont initialement dans trois établissements, respectivement A, B et C. Cependant, ils demandent tous les trois une mutation avec cette condition : ils ne veulent pas se retrouver dans un établissement qu'ils aiment moins que ce qu'ils ont déjà (autrement ils n'ont aucun intérêt à demander une mutation).
On se place dans le cas où on a trois enseignants α, β et γ, qui sont initialement dans trois établissements, respectivement A, B et C. α veut aller en B mais pas en C, β veut aller en A mais pas en C, γ veut aller en A ou en B.



Les établissements préfèrent tous γ, mais A et B sont tenus d'indiquer respectivement α et β en première préférence, à cause de la condition définie ci-dessus. Ce qui donne comme matrices de préférences :

$$M1 = \begin{pmatrix} 2 & 1 & 3 \\ 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix}$$

$$M2 = \begin{pmatrix} 1 & 3 & 3 \\ 3 & 1 & 2 \\ 2 & 2 & 1 \end{pmatrix}$$

On applique alors l'algorithme de Gale-Shapley avec *Caml* :

```
#MariageStable M1 M2 ;;
- : int vect = [|0; 1; 2|]
```

La solution stable optimale pour les enseignants est donc... de ne pas bouger !
On comprend bien que personne ne veuille aller dans l'établissement C, et par conséquent que γ soit obligé d'y rester. Par contre, on voit mal pourquoi α et β ne peuvent pas échanger leurs établissement. La raison réside dans le fait que l'affectation $(\alpha, B), (\beta, A), (\gamma, C)$ est *instable*. Pourtant cette situation serait préférable à la situation actuelle, pour les enseignants et pour les établissements. Il faut donc développer un autre algorithme que celui de Gale-Shapley qui permette de résoudre ce problème.

6.2 Pistes de solutions

Définition

On pourrait utiliser un modèle similaire à celui des mariages stables, mais en choisissant une nouvelle définition de la stabilité plus appropriée à ce problème en particulier.

Graphes

On pourrait également utiliser un système de graphes pour simplifier le problème et appliquer l'algorithme de Gale-Shapley uniquement sur des petits groupes isolés.

Annexes

Implémentation de l'algorithme de Gale-Shapley

On va s'intéresser dans cette annexe à l'implémentation de l'algorithme de Gale-Shapley. Pour cela, on va utiliser la langage *Caml*.

Fonctions préliminaires

On a besoin de deux fonctions sur les files, ainsi que d'une fonction *Ordonner*.

```
#open "queue" ;;

let File n =
   let file = new () in
   for i = 0 to (n-1) do add i file done ;
   file ;;

let est_vide f =
   try peek f ; false with
   | Empty -> true ;;

#open "stack" ;;

let Ordonner M1 =
   let n = vect_length M1.(0) in
   let T = make_vect n (new ()) in
   for i = 0 to (n-1) do
      let pile = new () in
      let T1 = make_vect n 0 in
         for j = 0 to (n-1) do
            T1.(M1.(i).(j)-1) <- j
         done ;
         for k = 0 to (n-1) do
            push T1.(n-1-k) pile
         done ;
      T.(i) <- pile
   done ;
   T ;;
```

La fontion *Ordonner* prend en argument une matrice des préférences, par exemple celle des hommes, et elle renvoit un tableau contenant des piles des femmes selon la préférence de chaque homme. Sa complexité est en $\Theta(n^2)$, mais on peut facilement montrer qu'il n'existe pas d'algorithme plus efficace.

Algorithme de Gale-Shapley

On peut alors écrire l'algorithme de Gale-Shapley :

```
let MariageStable M1 M2 =
   let n = vect_length M1.(0) in
   let Epouse = make_vect n (-1) in
   let Mari = make_vect n (-1) in
   let Liste = Ordonner M1 in
   let HL = File n in
```

```
    while not(est_vide HL) do
      let homme = take HL in
      let femme = pop Liste.(homme) in
        if Mari.(femme) = -1
        then begin Mari.(femme) <- homme ;
             Epouse.(homme) <- femme end
        else if M2.(femme).(homme) < M2.(femme).(Mari.(femme))
             then begin add Mari.(femme) HL ;
                  Mari.(femme) <- homme ;
                  Epouse.(homme) <- femme end
             else add homme HL
    done ;
  Epouse ;;
```

Cette fonction est du type :

```
MariageStable : int vect vect * 'a vect vect -> int vect = <fun>
```

Exemple

On peut désormais s'intéresser à des exemples. On reprend l'exemple qui a été traité en 1.2 :

$$M1 = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \\ 2 & 3 & 1 \end{pmatrix} \; ; \; M2 = \begin{pmatrix} 3 & 1 & 2 \\ 2 & 3 & 1 \\ 1 & 2 & 3 \end{pmatrix}$$

Si les hommes se proposent aux femmes, le résultat obtenu est :

```
#MariageStable M1 M2 ;;
- : int vect = [|0; 1; 2|]
```

En revanche, si les femmes se proposent aux hommes, le résultat est différent :

```
#MariageStable M2 M1 ;;
- : int vect = [|1; 2; 0|]
```

Optmalité

On peut regarder qui est le plus satisfait parmi les hommes et les femmes selon le groupe qui se propose à l'autre. Pour cela, on introduit une fonction *Satisfaction* qui rend les rangs des épouses et des maris à la fin de l'algorithme de Gale-Shapley.

```
let Satisfaction M1 M2 =
  let T = MariageStable M1 M2 in
  let n = vect_length T in
  let Sat = [| make_vect n 0 ; make_vect n 0 |] in
    for i = 0 to (n-1) do
      Sat.(0).(i) <- M1.(i).(T.(i)) ;
      Sat.(1).(i) <- M2.(T.(i)).(i)
    done ;
  Sat ;;
```

On applique cette fonction aux matrices $M1$ et $M2$ définies tout à l'heure :

```
#Satisfaction M1 M2 ;;
- : int vect vect = [|[|1; 1; 1|]; [|3; 3; 3|]|]
```

```
#Satisfaction M2 M1 ;;
- : int vect vect = [|[|1; 1; 1|]; [|3; 3; 3|]|]
```

On observe que lorsque les hommes se proposent aux femmes, ils épousent tous leur première préférence. Le couplage est donc optimal pour eux.
De même, il est optimal pour les femmes lorsqu'elles se proposent aux hommes.

Implémentation de l'algorithme de Selkow

On va avoir besoin de quelques fonctions auxiliaires (dont on ne donnera pas l'implémentation ici afin de ne pas prolonger inutilement ce rapport).
La fonction *Initial* donne un tableau contenant les bornes sup et inf des hommes et des femmes. La fonction *Testequi* teste si pour la borne sup de chaque homme est égale à sa borne inf. La fonction *MariageOpt* effectue un couplage stable, optimal en tenant compte des bornes sup des hommes et des femmes. Il s'agit de l'algorithme de Gale-Shapley légerement modifié. Enfin, la fonction *TrouveIndividu* cherche l'homme ou la femme, parmi ceux dont la borne sup est différente de la borne inf, tel que sa borne sup soit maximale.

```
let MariageStableEquitable M1 M2 =
   let n = vect_length M1.(0) in
   let PrefH = Pref M1 in
   let PrefF = Pref M2 in
   let (Hommes,Femmes) = Initial M1 M2 in
      while not(TestEqui Hommes) do
         let N1 = MariageOpt M1 M2 Hommes in
         let N2 = MariageOpt M2 M1 Femmes in
         for i=0 to (n-1) do
            Hommes.(i).(0) <- M1.(i).(N1.(i)) ;
            Femmes.(i).(0) <- M2.(i).(N2.(i)) ;
            Hommes.(N2.(i)).(1) <- M1.(N2.(i)).(i) ;
            Femmes.(N1.(i)).(1) <- M2.(N1.(i)).(i)
         done ;
         let (ind,val) = TrouveIndividu Hommes Femmes in
            if ind <> (-1) then begin
            if val then let femme = PrefH.(ind).(Hommes.(ind).(1)-1) in
                   begin Hommes.(ind).(1) <- Hommes.(ind).(1)-1 ;
                   Femmes.(femme).(0) <- Femmes.(femme).(0)+1 end
            else let homme = PrefF.(ind).(Femmes.(ind).(1)-1) in
                   begin Femmes.(ind).(1) <- Femmes.(ind).(1)-1 ;
                   Hommes.(homme).(0) <- Hommes.(homme).(0)+1 end
            end
      done ;
   let Epouse = make_vect n (-1) in
      for i=0 to (n-1) do
         Epouse.(i) <- PrefH.(i).(Hommes.(i).(0)-1)
      done ;
   Epouse ;;
```

Références

[1] D. Gale & L.S. Shapley. College admissions and the stability of marriage. 1962.

[2] M. Balinski & G. Ratier. Graphs and marriages. 1998.

[3] B. Werner & F. Pottier. *Algorithmique et programmation, un problème type : les mariages stables*. 26 avril 2013.

[4] Donald E. Knuth. *Mariages stables et leurs relations avec d'autres problèmes combinatoires*. Les Presses de l'Université de Montréal, 1976.