# Le problème d'affectation sur concours des étudiants

Jean-Baptiste Rouquier

14 février 2004

Différentes grandes écoles organisent un concours national pour recruter leurs étudiants. Certaines se regroupent et permettent ainsi aux étudiants de se présenter à plusieurs écoles en même temps, en ne passant qu'un concours. À l'issue de moult épreuves, chaque école établit un classement qui lui est propre des étudiants. Réciproquement, chaque étudiant a une liste de souhaits où il classe les écoles par ordre de préférence.

Le concours terminé, il s'agit d'affecter les étudiants aux écoles. Les écoles se sont accordées à procéder de manière centralisée. Ainsi un organisme indépendant collecte les classements des écoles et les listes de souhaits des étudiants. Par exemple, la table 1 représente les informations collectées pour les étudiants $a, b, c$ s'étant présentés aux concours des écoles $A, B, C$.
À partir de ces listes, le but est de proposer une affectation des étudiants dans les écoles la plus juste possible.

| | |
|---|---|
| $A$ | $a > c > b$ |
| $B$ | $c > a > b$ |
| $C$ | $c > a > b$ |

| | |
|---|---|
| $a$ | $B > A > C$ |
| $b$ | $C > A > B$ |
| $c$ | $A > B > C$ |

TAB. 1 – exemple de listes de préférences données par les étudiants et de classements donnés par les écoles

**définitions**

- On numérote les écoles $E_1 \dots E_m$ et les étudiants $e_1 \dots e_n$.
- On note $p_i$ le nombre de place de $E_i$.
- Une *affectation* est une fonction de l'ensemble des étudiants dans l'ensemble des écoles. Cette fonction n'est pas toujours définie car certains étudiants n'ont pas d'école ou préfèrent redoubler. Elle doit respecter le nombre de places de chaque école.
  Autrement dit, c'est un ensemble de couples $(e, E)$ où $e$ est un étudiant affecté l'école $E$, tel que chaque étudiant apparaisse dans au plus un couple, et chaque école $E_i$ apparaisse dans au plus $p_i$ couples.
- La notion d'*affectation stable*, définie plus loin, permet de donner un sens précis aux mots «la plus juste possible».

# 1 cas où chaque école n'a qu'une place

Commençons par regarder le cas où $\forall i, p_i = 1$.

## 1.1 listes complètes et $m = n$.

Descendons même au cas particulier où chaque école a classé tous les étudiants, et réciproquement chaque étudiant mis toutes les écoles dans sa liste de préférences : les listes sont complètes. On suppose aussi qu'il y a autant d'étudiants que d'écoles : $m = n$. Une affectation est donc ici une bijection entre l'ensemble des étudiants et l'ensemble des écoles.

**affectation stable** Une affectation $\mathcal{A}$ est dite *instable* s'il existe un étudiant $e$ et une école $E$ tels que :

- $(e, E) \notin \mathcal{A}$. ($e$ n'est pas affecté à $E$)
- $(e, F) \in \mathcal{A}$ et $(f, E) \in \mathcal{A}$
- $e : E > F$ et $E : e > f$ ($e$ préfère $E$ à $F$ et $E$ préfère $e$ à $f$).

Autrement dit les associations $(e, F)$ et $(f, E)$ ne sont satisfaisantes ni pour $e$ ni pour $E$. Dans ce cas, l'échange qui consiste à faire l'association $(e, E)$ (et donc aussi $(f, F)$) est appelé un *échange améliorant*. Il est noté $(e, E)$.
Si cette situation n'existe pas, l'affectation est dite *stable*. (Pour les amateurs, «Une affectation est dite stable si et seulement si elle n'est pas instable.») Le but est de trouver des affectations stables, nous allons montrer par un algorithme qu'il en existe toujours (dans ce cas particulier). Mais tout d'abord :

### 1.1.1 Une affectation stable n'est pas toujours unique.

Considérons en effet les listes de la table 2. Soit $0 \leq k \leq n-1$, alors l'affectation $(e_i, E_{i+k \bmod n}), 1 \leq i \leq n$ est stable (avec $\bmod n$ défini sur $1..n$). $e_i$ accepterait d'aller dans les écoles $E_i$ à $E_{i+k-1 \bmod n}$ à la place de celle qu'il a eue. Si $E_j$ fait partie de ces écoles, $E_j$ s'écrit $E_{i+l \bmod n}, 0 \leq l \leq k-1$. $E_j$ a reçu $e_{j-k \bmod n}$ et accepterait n'importe quel étudiant à la place de celui qu'elle a reçu, sauf $e_{j-k+1 \bmod n}$ à $e_j$. Ces étudiants refusés sont $e_{i+l-k+1 \bmod n}$ à $e_{i+l \bmod n}$, parmi lesquels figure en particulier $e_i$. Il n'y a donc pas d'échange améliorant.

Ceci montré, un algorithme naïf consistant à faire des échanges améliorants peut ne pas marcher :

### 1.1.2 Une succession d'échanges améliorants ne mène pas toujours à une affectation stable.

Considérons en effet les listes de la table 1 et commençons avec l'affectation $\{(a, A), (b, B), (c, C)\}$. $a$ préfère $B$ qui l'accepte, effectuons l'échange et nous obtenons l'affectation $\{(b, A), (a, B), (c, C)\}$. Mais de même, $(c, B)$ est alors un échange améliorant, qui aboutit à $\{(b, A), (c, B), (a, C)\}$. On continue ainsi et l'on revient à la position de départ, il existe donc une suite infinie d'échanges améliorants. Ceci est résumé dans la table 3.

### 1.1.3 un algorithme pour calculer des affectations stables

**l'algorithme**

| | |
|---|---|
| $E_1$ | $e_2 > e_3 > e_4 > \ldots > e_n > e_1$ |
| $\vdots$ | |
| $E_j$ | $e_{j+1} > \ldots > e_n > e_1 > \ldots > e_j$ |
| $\vdots$ | |
| $E_n$ | $e_1 > e_2 > \ldots > e_n$ |

| | |
|---|---|
| $e_1$ | $E_1 > E_2 > \ldots > E_{n-1} > E_n$ |
| $\vdots$ | |
| $e_i$ | $E_i > \ldots > E_n > E_1 > \ldots > E_{i-1}$ |
| $\vdots$ | |
| $e_n$ | $E_n > E_1 > E_2 > \ldots > E_{n-1}$ |

TAB. 2 – listes pour lesquelles il existe plusieurs affectations stables

| affectation | échange améliorant | justification |
|---|---|---|
| $\{(a, A), (b, B), (c, C)\}$ | $(a, B)$ | $a : B > A$ et $B : a > b$ |
| $\{(b, A), (a, B), (c, C)\}$ | $(c, B)$ | $c : B > C$ et $B : c > a$ |
| $\{(b, A), (c, B), (a, C)\}$ | $(c, A)$ | $c : A > B$ et $A : c > b$ |
| $\{(c, A), (b, B), (a, C)\}$ | $(a, A)$ | $a : A > C$ et $A : a > c$ |
| $\{(a, A), (b, B), (c, C)\}$ | ... | ... |

TAB. 3 – une suite infinie d'échanges améliorants. Chaque échange aboutit à l'affectation de la ligne suivante.

**initialisation** *Tous les étudiants sont libres et toutes les écoles sont vides.*

**boucle** *Tant qu'il existe une école $E$ vide faire :*

*prendre $e$ le premier étudiant de la liste de $E$ à qui $E$ n'a pas encore proposé de poste ;*

*si $e$ est libre, alors affecter provisoirement $e$ à $E$ ;*

*sinon*

*si $e$ préfère $E$ à son école provisoire $F$, alors affecter provisoirement $e$ à $E$ (et $F$ redevient vide) ;*

*sinon $e$ rejette la proposition de $E$ (qui reste vide) ;*

**renvoi** *L'affectation finale est l'ensemble des affectations provisoires.*

**terminaison** Considérons un arrêt sur image de l'algorithme en cours d'éxécution. À chaque école $E$ on associe un entier $\varphi(E)$ : $n+1$ si $E$ n'a encore proposé de poste à personne ($E$ est vide et on n'a pas encore cherché à la remplir), $n+1-j$ si le dernier étudiant auquel $E$ a proposé un poste était le $j^{\text{ème}}$ étudiant sur la liste de $E$. Noter que $e$ n'a pas forcément accepté cette proposition.
À chaque étape de l'algorithme, l'un des $\varphi(E_i)$ décroît d'une unité, d'où la terminaison de l'algorithme.

**correction** Soit $E$ une école, $e$ l'étudiant qui lui a été attribué par l'algorithme. Soit $f$ un étudiant mieux classé que $e$ dans l'école $E$. Soit $F$ l'école où $f$ est finalement affecté. Alors $f$ préfère $F$ à $E$, donc il n'existe pas d'échange améliorant dans l'affectation renvoyée par l'algorithme.
En effet, $f$ ne peut quitter une école que pour une école mieux classée dans ses préférences. En particulier lorsque $E$ l'a appelé (car $E$ a appelé les étudiants jusqu'à $e$), $f$ a soit accepté (pour accepter plus tard la proposition de $F$), soit était déjà dans une école qu'il préférait à $E$. Donc $f$ préfère bien $F$ à $E$.

Nous avons ainsi prouvé (constructivement) qu'il existe toujours une affectation stable dans ce cas particulier.

**complexité** Précisons les structures de données utilisées (dans le style d'une interface pour OCaml) :

```
type ecole = {
  classement : int array;
  mutable index : int
    (*le premier étudiant à qui l'école n'a pas encore proposé
      de poste.
      "mutable" car cela changera au cours de l'algorithme.*)}

type etudiant = {
  mutable affectation : int option;
    (*l'école où il est affecté. "option" car il n'est pas
      affecté au début.*)
  preferences : int array
```

```
    (*preferences.(i) est le rang de l'école i sur la liste de
      cet étudiant. Ainsi l'on peut savoir en temps constant
      s'il préfère l'école i à l'école j.*)}

val ecoles : ecole array (*le tableau contenant les écoles*)
val etudiants : etudiant array (*besoin d'un commentaire ?*)

val ecoles_vides : int Queue.t
  (*la pile LIFO des écoles vides. FIFO conviendrait aussi.*)
```

Ceci permet de faire un passage dans la boucle en temps constant, c'est donc ce que l'on prendra comme opération élémentaire : une proposition d'une école à un étudiant.

Nous avons déjà vu (en montrant la terminaison) que l'algorithme effectuait au plus $O(n^2)$ opérations élémentaires. La table 4 (où les listes sont identiques) nous montre un cas où il faut $O(n^2)$ opérations élémentaires. En effet la seule affectation stable est $\{(e_i, E_i), \quad 1 \leq i \leq n\}$ (ceci sera montré à la partie 1.5). Comme chaque école appelle les étudiants dans l'ordre de la liste, il faut $\frac{n(n+1)}{2}$ étapes.

| $e_i$ | $E_1 < \ldots < E_n$ |
|---|---|

| $E_i$ | $e_1 < \ldots < e_n$ |
|---|---|

TAB. 4 – une borne inférieure pour la complexité de l'algorithme

## 1.2 Qui est le plus satisfait ?

Montrons que cet algorithme, bien que fournissant une affectation stable, privilégie les écoles.

### 1.2.1 Chaque école obtient par cet algorithme le meilleur étudiant qu'elle puisse avoir dans toute affectation stable.

À un instant dans le déroulement de l'algorithme, notons t le nombre d'étapes déjà effectuées (c'est à dire le nombre de passages dans la boucle). Montrons par induction sur t : *si e refuse ou quitte E alors il n'existe pas d'affectation stable contenant* $(e, E)$. Ceci prouvera le résultat annoncé : une école n'appelle un étudiant que si elle ne peut recevoir aucun des précédents sur sa liste.

Soit $\mathcal{A}$ une affectation stable contenant $(e, E)$. Si $e$ refuse $E$ à l'étape $t$, c'est qu'il est dans une école $F$ qu'il préfère. Soit $f$ l'étudiant affecté à $F$ dans $\mathcal{A}$. On a ainsi $\{(e, E), (f, F)\} \subset \mathcal{A}$.

- Si $f$ est mieux classé que $e$ dans $F$, c'est que $F$ a appelé $f$ qui l'a refusée ou quittée (puisque $F$ a ensuite appelé $e$ qui a accepté). Et ce à une étape antérieure à $t$, l'hypothèse de récurrence nous permet de conclure que $\mathcal{A}$ (qui contient $(f, F)$) n'est pas stable : contradiction.
- Si au contraire $f$ est moins bien classé que $e$ dans $F$, alors $(e, F)$ est un échange améliorant : contradiction.
  Noter que ce deuxième cas réalise l'initialisation de la récurrence.

#### 1.2.2 Inversement, chaque étudiant a la pire école qu'il puisse avoir dans toute affectation stable.

Soient en effet $e$ un étudiant, $E$ l'école qui lui est donnée par l'algorithme, $F$ une école moins bien classée que $E$ sur sa liste, $\mathcal{A}$ une affectation stable contenant $(e, F)$. On veut montrer que $\mathcal{A}$ ne peut pas exister. Soit $f$ l'étudiant affecté à $E$ dans $\mathcal{A}$. Résumons : $\{(e, F), (f, E)\} \subset \mathcal{A}$.

- Si $f$ est moins bien classé que $e$ dans $E$, alors $(e, E)$ est un échange améliorant : contradiction.
- Si $f$ est mieux classé que $e$ dans $E$, c'est que $E$ a appelé $f$ qui l'a refusée ou quittée (puisque $E$ a ensuite appelé $e$ qui a accepté). Mais nous venons de prouver qu'alors il n'existe par d'affectation stable contenant $(f, E)$ : contradiction.

#### 1.2.3 Remarquons que les deux propositions précédentes montrent que, quelle que soit l'implémentation de l'algorithme, on obtient la même affectation stable à la fin.

Des implémentations différentes seraient des variations sur le choix de l'école vide à chaque passage dans la boucle. Ici la variable `ecoles_vides` est LIFO ou FIFO, mais on pourrait prendre les écoles dans n'importe quel ordre.

### 1.3 Listes complètes et $m < n$.

Passons maintenant au cas où le nombre total de places est inférieur au nombre d'étudiants : $m < n$. On suppose toujours les listes complètes.

**définitions**

- Ici une *affectation* est une application injective de l'ensemble des écoles dans l'ensemble des étudiants.
- Une affectation est *instable* s'il existe un étudiant $e$ et une école $E$ tels que
  - $E$ préfère $e$ à l'étudiant qui lui est affecté
  - $e$ préfère $E$ à l'école où il est affecté, ou bien $e$ n'est affecté nulle part. C'est à dire «$e$ voudrait aller dans $E$».

#### 1.3.1 Il existe toujours une affectation stable dans ce cas.

Ajoutons en effet $n - m$ écoles fictives, avec des listes arbitraires, et complétons les listes des étudiants avec ces écoles à la fin. Faisons tourner l'algorithme précédent sur les écoles réelles d'abord puis, lorsque celles-ci sont pleines, sur les écoles fictives. Aucun étudiant ne préfère une école fictive à une école réelle, l'ajout des écoles fictives ne pénalisera donc pas les étudiants. D'autre part, l'ordre dans lequel on considère les écoles ne change pas l'affectation finale.

On obtient donc une affectation stable, avec «$e$ affecté nulle part» équivalent à «$e$ affecté à une école fictive».

On en déduit :

### 1.3.2 Un algorithme pour calculer une affectation stable dans ce cas

C'est le même que celui exposé plus haut, simplement il y a moins d'écoles que d'étudiants, donc certains étudiants resteront sans école.

## 1.4 Listes incomplètes et $m = n$

En réalité, les listes collectées ne sont pas complètes : chaque étudiant ne se présente pas à toutes les écoles donc les écoles ne classent pas tous les étudiants ; certains étudiants préfèrent redoubler plutôt que d'intégrer une école qui ne les intéresse pas (et ne mettent pas cette école dans leur liste).

On cherche alors une affectation stable $\mathcal{A}$, définie comme dans le cas «listes complètes et $m = n$» (partie 1.1), avec la condition supplémentaire :
$(e, E) \in \mathcal{A}$ *seulement si* $E$ est dans la liste de $e$ et $E$ a classé $e$.

### 1.4.1 Il y a maintenant des cas pour lesquels il existe une affectation vérifiant la condition supplémentaire, mais aucune affectation stable.

Regardons les listes de la table 5. $b$ n'accepte d'aller que dans $A$, la seule affectation vérifiant la condition est donc $\{(a, B), (b, A)\}$. Mais $(a, A)$ est alors un échange améliorant.

| | |
|---|---|
| $A$ | $a > b$ |
| $B$ | $a > b$ |

| | |
|---|---|
| $a$ | $A > B$ |
| $b$ | $A$ |

TAB. 5 – il n'y a pas toujours d'affectation stable pour des listes incomplètes

Cet exemple montre aussi que le concept d'affectation stable n'est pas satisfaisant : il est juste de donner priorité à $a$ dans le choix de son école, car $a$ était mieux classé partout. L'affectation finale contiendra donc $(a, A)$. On voit alors que $b$ préfère redoubler qu'intégrer $B$ : libre à lui, c'est son choix ! L'affectation finale devrait être $\{(a, A)\}$.

Certains auront remarqué qu'ici non plus il n'existe pas d'échange améliorant. Mais ce n'est pas une affectation au sens défini en 1.3 : il existe une école qui n'a pas reçu d'étudiant. Doit-on mépriser les étudiants pour garantir à chaque école qu'elle remplira son («ses», en anticipant un peu sur la suite) poste(s) ? Ce n'est pas le choix fait aujourd'hui.

### 1.4.2 lemme pour adapter l'algorithme

Ajoutons un étudiant fictif $e$ et une école fictive $E$, avec des listes de préférences complètes, avec $E$ (respectivement $e$) à la fin, et arbitraires pour le reste. Complétons la liste de chaque étudiant par $E$ à la fin, suivie des écoles qu'il n'a pas mises sur sa liste. Complétons de même la liste de chaque école par $e$ à la fin, suivi des étudiants non classés par $E$. On a alors la propriété suivante :

*Il existe une affectation stable pour le système de listes incomplètes initial si et seulement s'il existe une affectation stable pour le système de listes complètes contenant* $(e, E)$

Preuve :

– S'il existe une affectation stable $\mathcal{A}$ pour le système de listes incomplètes initial, alors il n'existe pas d'échange améliorant dans cette affectation, c'est à dire qu'aucun étudiant n'est préféré par une école qu'il préfère. Ajouter des écoles à la fin des listes des étudiants ne change rien à l'ensemble des écoles qu'un étudiant préfère, donc il n'existe pas non plus d'échange améliorant dans $\mathcal{A}$ complétée par $(e, E)$ (sauf éventuellement pour $e$, mais aucune école ne préfère $e$ à l'étudiant qu'elle a reçu dans $\mathcal{A}$).
– Réciproquement, si $\mathcal{A}$ est une affectation stable du système de listes complètes telle que $(e, E) \in \mathcal{A}$, alors supprimer $e$, $E$ et la fin des listes ne risque pas de créer d'échange améliorant. Reste à montrer que l'affectation obtenue vérifie toujours la condition supplémentaire (énoncée en 1.4). Soit $F$ une école autre que $E$, l'étudiant $f$ qui lui est affecté est classé avant $e$ (sinon $(e, F)$ est un échange améliorant car $e$ préfère $F$ à $E$). Mais ceci signifie justement que $f$ est dans la liste initiale de $F$.
Soit de même $g$ un étudiant, $G$ l'école où il est affecté. $G$ est classée avant $E$ dans la liste complétée de $g$ donc $G$ est sur la liste initiale de $g$.

#### 1.4.3 Avec des listes complètes telles que $e$ soit le dernier choix de $E$ et $E$ le dernier choix de $e$, si une affectation stable contient $(e, E)$, alors toutes les affectations stables contiennent $(e, E)$.

Soit $\mathcal{A}$ une affectation stable contenant le couple $(e, E)$, supposons par l'absurde qu'il existe $\mathcal{B}$ une affectation stable ne le contenant pas.
Soit $F_1$ l'école où $e$ est affecté dans $\mathcal{B}$, $f_1$ l'étudiant affecté à $F_1$ dans $\mathcal{A}$. De manière plus générale, soit $F_i$ l'école où $f_{i-1}$ est affecté dans $\mathcal{B}$, $f_i$ l'étudiant affecté à $F_i$ dans $\mathcal{A}$. Résumons :

$$\begin{array}{rcl} \{(e, E), (f_1, F_1), (f_2, F_2), \ldots, (f_k, F_k)\} & \subset & \mathcal{A} \\ \{(e, F_1), (f_1, F_2), \ldots, (f_{k-1}, F_k), (f_k, E)\} & \subset & \mathcal{B} \end{array}$$

$F_1$ préfère $f_1$ à $e$ car $\mathcal{A}$ est stable. Donc $f_1$ préfère $F_2$ à $F_1$ car $\mathcal{B}$ est stable. Et de même $F_2 : f_2 > f_1$ car $\mathcal{A}$ est stable. Donc $f_2 : F_3 > F_2$ car $\mathcal{B}$ est stable.
Une récurrence immédiate donne $F_k : f_k > f_{k-1}$. Or $f_k : F_k > E$ car $\mathcal{A}$ est stable. Donc $\mathcal{B}$ n'est pas stable ($(f_k, F_k)$ est un échange améliorant). Contradiction.

#### 1.4.4 On peut alors résoudre le cas des listes incomplètes :

On ajoute une école et un étudiant fictifs comme ci-dessus, on fait tourner l'algorithme, les deux lemmes précédents (1.4.2 et 1.4.3) garantissent qu'il existe une affectation stable pour le problème initial si et seulement si l'algorithme retourne une affectation contenant $(e, E)$.

La complexité de cette méthode est celle de l'algorithme : $O(n^2)$, les autres opérations (compléter les listes par exemple) se faisant aussi en $O(n^2)$.

Rappelons que cette affectation stable peut être criante d'injustice car elle cherche à tout prix à fournir un étudiant à chaque école, donc interdit aux étudiants de redoubler, donc peut en désavantager un pour permettre à un autre d'obtenir une école de sa liste. Une stratégie pour les étudiants serait alors de ne mettre que très peu d'écoles sur leur liste, pour forcer l'algorithme à leur donner une école parmi les meilleures. Le seul risque pour les étudiants est qu'il n'existe pas d'affectation stable.

## 1.5 Réputation des écoles

La réputation des écoles tend à rendre les listes des étudiants presque identiques. Pour simplifier, supposons que tous les étudiants ont la même liste complète $E_1 > \ldots > E_m$. Supposons de plus $m = n$.

### 1.5.1 Il existe alors une unique affectation stable.

Récurrence sur n :
C'est trivial s'il n'y a qu'une école et un étudiant.
Supposons la propriété vérifiée jusqu'à n. Soit $e$ le premier étudiant de $E_1$. Toute affectation ne donnant pas $E_1$ à $e$ admet $(e, E_1)$ comme échange améliorant. Mais une fois $e$ affecté à $E_1$, l'hypothèse de récurrence permet de conclure.

### 1.5.2 Comment calculer cette affectation

La taille des données est en $O(n^2)$, il est donc peu vraisemblable d'obtenir une meilleure complexité sans imposer de conditions suplémentaires sur l'entrée. On garde donc le même algorithme qui est en $O(n^2)$ (ou mieux : proportionnel à la longueur totale des listes des écoles si elles sont incomplètes).

# 2 Cas général

Revenons au cas général (décrit en introduction) où les nombres $p_i$ de places sont quelconques. On suppose qu'il y a moins de places que d'étudiants, c'est à dire que $\sum_{i=1}^{m} p_i \leq n$.

On reprend la définition d'une affectation exposée dans l'introduction (les listes ne sont donc pas supposées complètes).
Une affectation est dite *instable* si elle contient un couple $(e, E)$ tel que

- $e$ voudrait aller dans $E$ (il est affecté nulle part ou dans une moins bonne école)
- $E$ préfère $e$ au dernier (selon son classement) étudiant qui lui est affecté.

Voir éventuellement la partie 1.3.

## 2.1 Il n'existe pas toujours d'affectation stable.

Le contre exemple d'un cas particulier vu plus haut reste vrai pour le cas général.

## 2.2 Un algorithme pour calculer une affectation stable

**algorithme** On remplace chaque école $E_i$ par $p_i$ écoles à une place ayant la même liste que $E_i$. On peut alors appliquer l'algorithme précédent.

**complexité** Comme précédemment, proportionnel à la longueur totale des listes des écoles.

**remarques**

- Ceci marche pour des listes incomplètes.
- Une dernière fois, une affectation stable n'est pas toujours satisfaisante. En particulier elle peut privilégier des étudiants astucieux qui tirent parti des particularités de l'algorithme pour obtenir une bonne école, et ce même s'ils y sont moins bien classés que d'autres qui voudraient également entrer dans cette bonne école. Voir l'exemple de la table 5. D'où...

## 2.3 un autre algorithme

### 2.3.1 description

On répartit les candidats en $m$ listes suivant le premier vœu des candidats ($m$ est toujours le nombre d'écoles). Pour chaque école $E_i$ (chacune des $m$ listes), on garde les $p_i$ premiers candidats ($p_i$ est toujours le nombre de places dans $E_i$), les autres sont stockés dans une unique liste $L$. On procède ainsi pour chaque école en ajoutant des candidats à $L$. Pour tous les candidats de $L$, on note leur premier vœu comme impossible à satisfaire.

On répartit alors les candidats de $L$ dans les $m$ listes suivant leur premier vœu potentiellement satisfiable (à cette étape c'est le deuxième, mais ce ne sera pas toujours le même pour tous les candidats de $L$). $L$ est alors vide.

Puis on recommence : pour chaque école $E_i$, on met les candidats au-delà du rang $p_i$ dans $L$. Pour ces candidats, on note leur vœu courant comme impossible à satisfaire (à cette étape c'est le premier ou le deuxième mais la démarche est générale). Il est ainsi possible qu'un candidat ayant été provisoirement affecté dans l'école de son premier vœu, soit «chassé» dans $L$ par un candidat mieux classé que lui ayant mis cette école en deuxième vœu.

On continue ainsi en vidant et remplissant alternativement $L$. Les candidats dont la liste de vœux est épuisée (ceux chassés dans $L$ alors qu'ils étaient dans la liste de l'école de leur dernier vœu) ne sont affectés nulle part. Si l'école ne classe pas un candidat (il n'est pas dans la liste de préférences de cette école) qui arrive dans sa liste provisoire, il est automatiquement chassé dans $L$.

Lorsque tous les candidats sont provisoirement affectés à une école ($L$ reste vide), cette affectation devient l'affectation définitive (ou la proposition de la semaine dans le système actuel). (C'est à dire qu'on propose à chaque candidat l'école de la liste dans laquelle il se trouve à la fin de l'exécution.)

### 2.3.2 justification

L'algorithme s'arrête évidemment car on progresse dans les listes de vœux.

On est assuré d'obtenir une affectation vérifiant la condition supplémentaire définie en 1.4 : un candidat n'est affecté dans une école que s'il y est classé et s'il a mis cette école dans sa liste de vœux.

Enfin il n'y a pas d'échange améliorant dans l'affectation finale (affectation au sens défini en introduction car il est possible qu'une école $E_i$ reçoive moins de $p_i$ candidats).

En effet, on montre par induction sur $t$ le nombre d'étapes déjà effectuées par l'algorithme : *lorsqu'on chasse un candidat $e$ de $E$ dans $L$, c'est que le vœu courant de $e$ ne pouvait être satisfait.* Si l'affectation finale contenait $(e, E)$, alors il existerait un candidat $f$ mieux classé que $e$ dans $E$ (puisque $e$ a été

chassé dans $L$). $f$ préfère $E$ à l'école qui lui est donnée dans l'affectation finale (ou bien $f$ n'est affecté nulle part) car $f$ avait été chassé dans $L$ pour chacun de ses vœux précédant $E$, et ce à des étapes antérieures. Donc $(f, E)$ est un échange améliorant.

#### 2.3.3 remarques

- Cet algorithme privilégie les candidats : alors que sur l'exemple de la table 2 le premier aurait donné à chaque école le candidat le mieux classé dans cette école, cet algorithme donne à chaque étudiant la première école de sa liste de vœux.
- Il faut bien sûr utiliser une structure de données appropriée, un tas à $p_i$ éléments par exemple, au lieu des listes commodes pour la description de l'algorithme. Vider $L$ consiste toujours à répartir les étudiants en $m$ listes, mais on traite ensuite chacune des ces listes comme suit : pour chaque école $E$, pour chaque étudiant $e$ de la liste (que l'on vient de remplir en vidant $L$), si $e$ est moins bien classé que la racine du tas de $E$, alors $e$ retourne dans $L$. Sinon la racine (i.e. l'étudiant provisoirement affecté à $E$ le moins bien classé) retourne dans $L$ et on insère $e$ dans le tas. La complexité du pire cas est alors

  $$O\left(n\sum_{i=1}^{m}(1+\log p_i)\right)$$

  car chaque étudiant «passe» au pire dans les $m$ écoles, où il se fait éventuellement insérer dans le tas en temps $\log p_i$.
- Sur l'exemple de la table 5, cet algorithme donne l'affectation attendue : $\{(a, A)\}$ et $b$ redouble.

# 3 Casse-tête au service hébergement

En épilogue, l'école $Y$ recrute $2n$ étudiants qu'elle se charge de loger dans sa résidence à raison de deux étudiants par chambre. Après une grande fête d'intégration qui a permis à tous les étudiants de se rencontrer, l'école demande à chaque étudiant de classer par ordre de préférence les autres étudiants avec qui il souhaite partager sa chambre. L'école réclame des *listes complètes*, c'est à dire où tous les étudiants sont classés.

Muni de toutes ces listes et pour maintenir une bonne ambiance dans la résidence, le service hébergement cherche une *partition stable* des étudiants. Une *partition* est ici une partition des $2n$ étudiants en $n$ paires de deux étudiants. Une partition est *instable* s'il existe deux paires $(a, b)$ et $(c, d)$ telles que $a$ préfère $d$ à $b$ et $d$ préfère $a$ à $c$.

## 3.1 Il n'existe pas toujours de partition stable.

Considérons en effet les listes de préférences suivantes :

$$\begin{aligned} a &: c > d > b \\ b &: \text{arbitraire} \end{aligned}$$

$$\begin{array}{rcl} c & : & d > a > b \\ d & : & a > c > b \end{array}$$

Les trois partitions possibles ont toutes un échange améliorant :

$$\begin{array}{rcl} (a,b),\ (c,d) & : & (a,d) \\ (a,c),\ (b,d) & : & (c,d) \\ (a,d),\ (b,c) & : & (a,c) \end{array}$$

En fait, le problème d'affectation stable des parties précédentes est plus facile que le problème de la partition stable.

### 3.2 Réduction

*Pour toute instance du problème d'affectation de $n$ étudiants dans $n$ écoles avec listes complètes, on peut construire une instance du problème de partition stable avec $2n$ personnes tel que les partitions stables correspondent exactement aux affectations stables de l'instance initiale.*

**instance** Il suffit de prendre $n$ personnes qui représentent les écoles, $n$ autres qui représentent les étudiants. Donnons aux étudiants les liste de préférences des étudiants du problème initial, de même pour les écoles. Les listes doivent être complètes, ajoutons donc à la fin de des listes des étudiants les autres étudiants dans un ordre arbitraire, de même pour les écoles.

**justification** Une affectation stable est trivialement transposée en une partition stable.

Réciproquement, il reste à montrer que dans une partition stable un étudiant ne peut être avec un étudiant (le cas des écoles est symétrique). Par abus de langage, on note «étudiant» pour «personne représentant un étudiant». Si un étudiant était avec un étudiant, alors il y aurait une école avec une école (principe des tiroirs). Mais un étudiant préfère être avec une école et une école préfère être avec un étudiant. Donc il existerait un échange améliorant.

**note** Il existe un algorithme en $O(n^2)$ qui permet à partir des listes complètes des $2n$ étudiants de déterminer s'il existe une partition stable et d'en calculer une si oui. Mais c'est un peu plus compliqué...

## Liste des tableaux