ALGO1 – Cours d'introduction
Problème des mariages stables

François Schwarzentruber

September 13, 2020

Modalités

Biblio : [Papadimitriou et al.], [Cormen et al.], [Kleinberg-Tardös]
Cours : mardi après-midi à 14h à l'ENS Rennes (amphi)
TD : jeudi matin de 8h à 10h à Beaulieu avec Léo Henry et Raphaël Truffet
Terminal : devoir sur table 2h
Contrôle continu : à chaque séance de TD, un binôme rend une correction du TD précédent

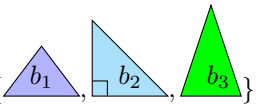
1 Motivation

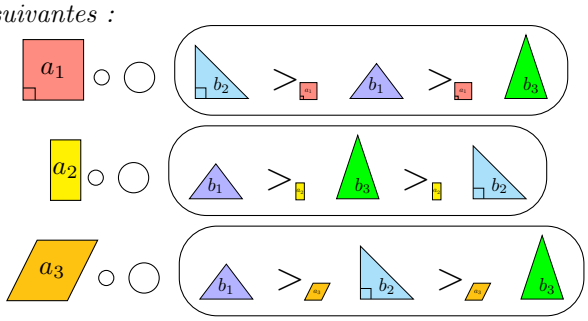
But : marier les $a \in A$ et les $b \in B$.

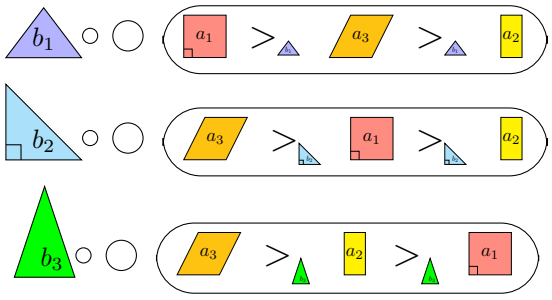
A	B
hommes	femmes
étudiant.e.s	encadrant.e.s de stage
candidats.es	postes
donneurs.ses de reins	receveurs.ses de reins

Chaque $a \in A$ exprime ses préférences sur les éléments de B.
Chaque $b \in B$ exprime ses préférences sur les éléments de A.

2 Modélisation

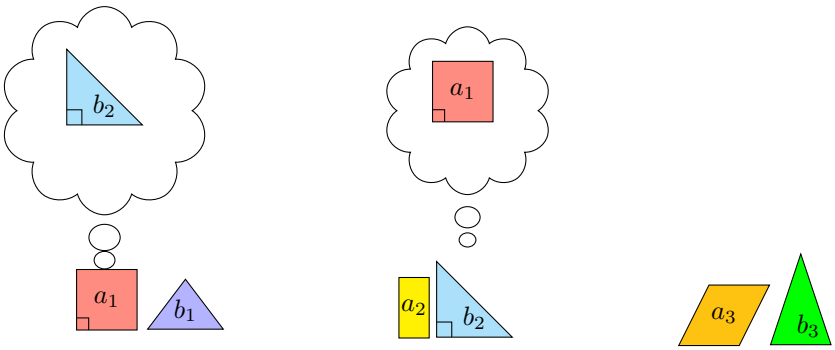
Exemple 1 *On considère l'ensemble* $A = \{$  $\}$, *l'ensemble* $B = \{$  $\}$ *et les préférences suivantes :*

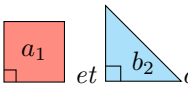


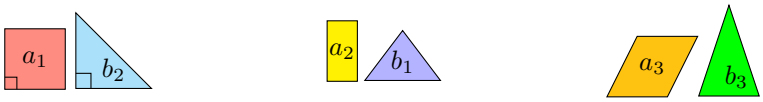


On ne veut pas de "regret commun" / instabilité.

Exemple 2 *Voici un couplage avec une instabilité :*



Le risque est que  *et communiquent, divorcent et se marient ensemble :*



Ce dernier couplage est également instable.

Exemple 3 *Voici des exemples de couplages stables :*

3 Formalisation

Soit A et B deux ensembles finis à n éléments.

Définition 4 (couplage) *Un sous-ensemble $S \subseteq A \times B$ est un* couplage *si tout élément de $A \cup B$ n'apparaît au plus qu'une seule fois dans un seul couple de S.* *(pas de polygamie)*

Définition 5 (couplage parfait) *Un couplage $S \subseteq A \times B$ est* parfait *si tout élément de $A \cup B$ apparaît exactement une fois dans un seul couple de S.* *(pas de polygamie, pas de célibataire)*

On modélise les préférences des individus par des relations d'ordre totales strictes.

Définition 6 (collection de préférences) *Une (A, B)-collection de préférences, notée $>$, est un couple de familles $((>_a)_{a \in A}, (>_b)_{b \in B})$ tel que :*

- *pour tout $a \in A$, $>_a$ est une relation d'ordre totale strict sur B ;*
- *pour tout $b \in B$, $>_b$ est une relation d'ordre totale strict sur A.*

Définition 7 (instabilité) *Étant donné $>$ et un couplage S, un couple $(a, b') \in A \times B$ est une $>$-instabilité pour S s'il existe $(a, b), (a', b') \in S$ avec $b' >_a b$ et $a >_{b'} a'$.*

Définition 8 (couplage stable) *Étant donné $>$, un couplage S est $>$-stable s'il est parfait et s'il n'y a pas de $>$-instabilité pour S.*

Définition 9 (problème des mariages stables) *Le problème des mariages stables est :*

- *entrée : deux ensembles finis A, B de même cardinal n, une (A, B)-collection de préférences $>$;*
- *sortie : un couplage $>$-stable $S \subseteq A \times B$ s'il en existe un.*

4 Algorithme de Gale-Shapley

```
fonction GS(A, B, >)
    initialiser tous les éléments de A et de B comme libres
    tant que il existe a ∈ A libre, qui n'a pas fait de proposition à tous les b ∈ B faire
        choisir un tel a ∈ A
        soit b l'élément préféré de a, à qui a n'a pas encore de proposition
        //a fait une proposition à b
        si b est libre alors
            (a, b) s'engage
        sinon b est engagé avec un certain a'
            si a >_b a' alors
                (a, b) s'engage
                a' devient libre
    retourner l'ensemble S des couples engagés
```

Durant l'exécution de l'algorithme :

- les partenaires d'un a sont de pire en pire ;
- les partenaires d'un b sont de mieux en mieux.

Définition 10 *On dit qu'un élément $a \in A$ se fait rejeter quand : soit il se propose à un b mais b a déjà mieux. Soit, il se fait larguer par b.*

5 Exemple d'exécution

Avec magnets.

6 Terminaison

Théorème 11 *$\mathsf{GS}(A, B, >)$ termine après au plus n^2 itérations de la boucle **tant que** où $n = \#A = \#B$.*

Démonstration. Exhiber un variant, c'est-à-dire une quantité qui décroit strictement à chaque itération de boucle.

(le choix du variant est subtil : par exemple le nombre d'individus libres n'est pas un variant, car il peut rester inchangé d'une itération à l'autre)

Soit $C(t)$ l'ensemble des couples (a, b) où a a fait une proposition à b entre le début de l'algorithme et la fin de la $t^{\text{ème}}$ itération de la boucle **tant que** .

$$\#C(t) < \#C(t+1) \text{ et } \#C(t) \leq n^2.$$

Donc il ne peut y avoir plus de n^2 itérations. ■

7 Correction

Remarque 12 *L'algorithme est sous-spécifié. Sur une même entrée $(A, B, >)$, il admet plusieurs exécutions, que l'on peut résumer sous la forme d'un arbre, appelé arbre de calcul.*

noeud : configuration

branche : une exécution

En fait, les algorithmes sont quasiment toujours sous-spécifiés (asynchronisme dans une architecture multi-coeurs, délai de réponse d'un serveur, etc.)

Théorème 13 *Toute exécution de $\mathsf{GS}(A, B, >)$ retourne un couplage S qui est $<$-stable.*

Démonstration. Exhiber un ou plusieurs invariants, i.e. des propriétés qui restent vraies tout au long de l'algorithme (ou d'une partie)

1. Montrons que S est un couplage.

   (I1): "l'ensemble des couples engagés est un couplage".

   (I1) est un invariant car vrai au début car $\emptyset$ est un couplage; conservé par une itération.

2. Montrons que S est parfait.

   (I2): Pour tout $a \in A$ libre, il existe un $b \in B$ à qui l'agent a n'a pas fait de proposition.

   Mq (I2) est un invariant. Par l'absurde, supposons qu'à un moment dans une exécution, il existe $a \in A$ libre, et qui s'est proposé à tous les $b \in B$. Donc tous les b sont engagés (dès qu'un b a reçu une proposition, il reste engagé avec quelqu'un pour toujours !). Donc le couplage est parfait ! Alors que a est libre... Contradiction.

   Du coup, par (I2), la condition de la boucle **tant que** devient "**tant que** il existe un a libre". A la fin d'une exécution, il n'y a pas de a libre. Donc S est parfait.

3. Montrons que S est stable. Par l'absurde, supposons qu'il existe une instabilité, i.e. qu'il y a $(a, b), (a', b') \in S$, $b' >_a b$ avec $a >_{b'} a'$.

   La dernière proposition réussie de a était faite à b. Comme $b' >_a b$, l'agent a a fait une proposition à b' (si a a fait une proposition à un certain b, il a fait une proposition à toutes les personnes avant ce b dans sa liste de préférence).

   Ensuite, l'agent a a été ensuite rejeté par b' pour un a'' que b' préfère, i.e. $a'' >_{b'} a$.

   Comme les partenaires de b' sont de mieux en mieux, et que a' est le partenaire final de b', on a $a' \geq_{b'} a''$, i.e. soit $a'' = a'$, ou alors $a' >_{b'} a''$. Par transitivité de $>_{b'}$, on a $a' >_{b'} a$. Contradiction . S est bien stable.

■

Corollaire 14 *Étant donné A, B et $>$, il existe toujours un couplage $>$-stable.*

8 Implémentation

Théorème 15 *On peut implémenter GS en temps $O(n^2)$, le corps de la boucle s'exécutant en $O(1)$.*

DÉMONSTRATION. Voici comment procéder.

- On suppose $A = B = \{1, \ldots, n\}$;
- Un $n \times n$-tableau $aPref$ donne les préférences des $a \in A$. Plus précisément, $aPref[a, i]$ est le $i^{\text{ème}}$ élément de B préférée par a.
- Pareil pour $bPref$.
- Liste chaînée pour stocker les $a \in A$ libres;
- Un tableau $next$ pour savoir à qui demande ensuite. $next[a]$ est le rang du $b \in B$ suivant à qui a va faire une proposition.
  - Au début, $next[a] = 1$ pour tout $a \in A$.
  - L'agent a propose à $aPref[a, next[a]]$, puis on incrémente $next[a]$ à chaque fois qu'il propose.
- Si b est libre, $current[b] = free$, sinon $current[b]$ est le partenaire courant de b.
- Malheureusement, le test $a >_b a'$ est en $O(n)$ avec $bPref$. Pour avoir ce test en $O(1)$, on précalcule une table $ranking$ où $ranking[b, a]$ est le rang de a dans les préférences de b, i.e. le i tel que $bPref[b, i] = a$.

■

9 Pseudo-code détaillé

```
GS(A, B, <)
        init()
        declarerLibreA(A)
        declarerLibreB(B)

        Tant que existeLibreA()
                choisir a libre
                Soit b := getElementPrefereeNonPropose(a)

                Si estLibreB?(b)
                        engage(a,b)
                Sinon
                        a' := partenaireA(b)
                        Si a' <b a
                                engage(a,b)



declarerLibreA(A)
        for(a in A)
                L := a::L

declarerLibreB(B)
        for(b in B)
                current[b] = .


existeLibreA()
        retourner L != null
```

```
init()
        for(a in A)
                Next[a] := 1
        for(b in B, i in {1, ..n})
                ranking[b, BPref[b, i] = i


getElementPrefereeNonPropose(a)
        APref[a, Next[a]]
        incrémenter Next[a]
```

10 Algorithme injuste

Exemple 16

$$\begin{array}{ll} a : b >_a b' & b : a' >_b a \\ a' : b' >_{a'} b & b' : a >_{b'} a' \end{array}$$

L'algorithme GS donne :

$$\begin{array}{cc} a & b \\ a' & b' \end{array} \Rightarrow \begin{array}{ccc} a & - & b \\ a' & & b' \end{array} \Rightarrow \begin{array}{ccc} a & - & b \\ a' & - & b' \end{array}$$

(et pareil si on commence par a')

Ainsi, le couplage stable $\begin{array}{ccc} a & - & b' \\ a' & - & b \end{array}$ *n'est jamais calculé, mais le serait si c'était les $b \in B$ qui feraient les propositions.*

En fait, toutes les exécutions calculent le même couplage : celui où chaque $a \in A$ est avec "son meilleur partenaire"... à définir formellement.

10.1 Algorithme favorisant les A

Définition 17 (partenaire valide) *L'agent b est un partenaire valide de a s'il existe un couplage stable contenant (a, b).*

Définition 18 (meilleur partenaire) *Le meilleur partenaire de a est $best(a) = sup_{\leq_a}\{partenaires\ valides\ de\ a\}$*
i.e. l'unique partenaire valide de a tel que pour tout $b' \in B$, b' partenaire valide de a implique $b' \leq_a best(a)$,

Théorème 19 *Toute exécution de $\mathsf{GS}(A, B, >)$ retourne $S^* = \{(a, best(a)) \mid a \in A\}$.*

Démonstration. Par l'absurde, supposons qu'il existe une exécution de $\mathsf{GS}(A, B, >)$ qui retourne $S \neq S^*$. Autrement dit, il existe $(a, b) \in S$ avec $b \neq best(a)$.

Fait 20 *Il existe un $a \in A$ rejeté par un partenaire valide.*

Démonstration. Prenons un a avec $(a, b) \in S$ avec $b \neq best(a)$. L'agent a est marié à b à la fin. Il s'est donc proposé en mariage à tous les éléments de B que a préfère à b, en particulier à $best(a)$, qui est un partenaire valide. Il s'est donc fait rejeté par $best(a)$. ■

Considérons le premier moment (*) de cette exécution où un certain $a \in A$ est rejeté par un partenaire **valide**. Ce partenaire valide est forcément $best(a)$: en effet, a a proposé dans l'ordre décroissant de ses préférences, et comme $best(a)$ est son partenaire valide préféré, ses rejets passés éventuels ne sont pas avec des partenaires valides.

- Soit a se propose à $best(a)$, mais $best(a)$ a déjà mieux : un beau a'.
- ou alors a se fait remplacer, car $best(a)$ a eu une meilleure offre par un beau a'.

Juste après le rejet de a, dans les deux cas, il y a un engagement $(a', best(a))$ avec $a' >_{best(a)} a$.

Par ailleurs, comme $best(a)$ est valide pour a, il existe un couplage stable S' contenant $(a, best(a))$. Soit b' le partenaire de a' dans S', i.e. tel que $(a', b') \in S'$. Le fait suivant conclut la démonstration avec une contradiction car S' est censé être stable.

Fait 21 *On a $best(a) >_{a'} b'$. Et donc on a l'instabilité suivante dans S' :*

DÉMONSTRATION. Par l'absurde, supposons que $b' >_{a'} best(a)$.

1. L'agent a' n'a pas été rejeté par un partenaire valide avant le moment (*).
2. En (*), on a $a' - best(a)$. Cela implique que a' a été rejeté par b' avant le moment (*), car a' propose par ordre de préférence décroissant, et donc a proposé à b'.
3. Mais b' est valide pour a' car $(a', b') \in S'$.

Contradiction. ■
■

10.2 Algorithme défavorisant les B

On vient de voir que l'algorithme favorise les A, mais de façon symétrique, on peut montrer qu'il défavorise les B.

Définition 22 (partenaire valide) *L'agent a est valide pour b s'il existe un couplage stable contenant (a, b).*

Définition 23 (pire partenaire) *Le pire partenaire valide pour b est $worst(b) = inf_{\leq_b}\{\text{partenaires valides de } b\}$.*

Théorème 24 *On a $S^* = \{(worst(b), b) \mid b \in B\}$.*

DÉMONSTRATION. Par l'absurde, supposons qu'il existe $a \in A$ tel que, en notant $b = best(a)$, on ait $a \neq worst(b)$. Alors il existe un couplage stable S' avec $(worst(b), b) \in S'$. Il existe $b' \neq b$ avec $(a, b') \in S'$. Dans S', on a l'instabilité :

■

11 Ensemble de tous les couplages stables

Fait 25 *Il y a des exemples avec un nombre exponentiel de couplages stables en n.*

Exemple 26 *On prend $A = \{1, \ldots, n\}$ et $B = \{1, \ldots, n\}$ avec n pair.*
Préférences des éléments de A :
1 : 1 2 ...
2 : 2 1 ...
3 : 3 4 ...
4 : 4 3 ...
$\vdots$
$n-1$ *:* n $n-1$ *...*
n *:* $n-1$ n *...*

Préférences des éléments de B :

1 : ... 1
2 : ... 2
3 : ... 3
4 : ... 4
$\vdots$
$n-1$ *: ...* $n-1$
n *: ...* n

Groupons les éléments de A par paire : 12, 34, ... Supposons que chaque élément d'une paire se marie soit avec leur soit avec leur premier choix, ou alors avec leur second choix.

Quand deux éléments de A d'une paire sont mariés à leur second choix, ils ne peuvent pas avoir leur premier choix car c'est le plus détesté des éléments de B.

Donc, tous ces mariages sont stables ! Il y en $2^{n/2}$.

12 Notes bibliographiques

Nous utilisons l'idée de Kleinberg et Tardos [KT06] : introduire l'algorithmique avec le problème des mariages stables et l'algorithme de Gale-Shapley. Gale et Shapley ont présenté l'algorithme dans [GS62].

Pour aller plus loin, vous pouvez consulter des ouvrages de référence dans le domaine du choix social comme [KMR16] ou [Man13], ainsi qu'un document en français écrit par Donald Knuth [Knu76].

References

[GS62] David Gale and Lloyd S Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

[KMR16] Bettina Klaus, David F. Manlove, and Francesca Rossi. Matching under preferences. In Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D. Procaccia, editors, *Handbook of Computational Social Choice*, pages 333–355. Cambridge University Press, 2016.

[Knu76] Donald Ervin Knuth. *Mariages stables et leurs relations avec d'autres problemes combinatoires: introduction a l'analysis mathematique des algorithmes-.* Les Presses de l'Université de Montréal, 1976.

[KT06] Jon Kleinberg and Eva Tardos. *Algorithm design.* Pearson Education India, 2006.

[Man13] David F. Manlove. *Algorithmics of Matching Under Preferences*, volume 2 of *Series on Theoretical Computer Science.* WorldScientific, 2013.