IA04 - Systèmes Multi-Agents

Chapitre 9 – Problème des mariages stables et algorithme d'acceptation différée

Khaled Belahcene

khaled.belahcene@hds.utc.fr

Hénoïk Willot

henoik.willot@hds.utc.fr

Problème des mariages stables

Entrées

- n un entier non nul
- A et B deux ensembles de taille n
- À chaque élément a de A on associe un ordre total sur B noté $\succ_a$.
- A chaque élément b de B on associe un ordre total sur A noté $\succ_b$.

Sorties

- Un ensemble de couples $C \subset A \times B$ est un **appariement** si chaque élément de A et chaque élément de B apparaissent dans au plus un élément de C, i.e. $\forall (a,b) \in C \, \forall (a',b') \in C \quad a = a' \iff b = b'$
- L'appariement C est **parfait** si chaque élément de A et chaque élément de B apparaissent dans exactement une paire élément de C, i.e. $|C| = |A| = |B| = n$
- Un appariement parfait C est **instable** lorsque : $\exists (a,b') \in C, \exists (a',b) \in C \quad (b \succ_a b' \ \wedge a \succ_b a')$. Dans ce cas la paire (a,b) est dite **critique** pour C

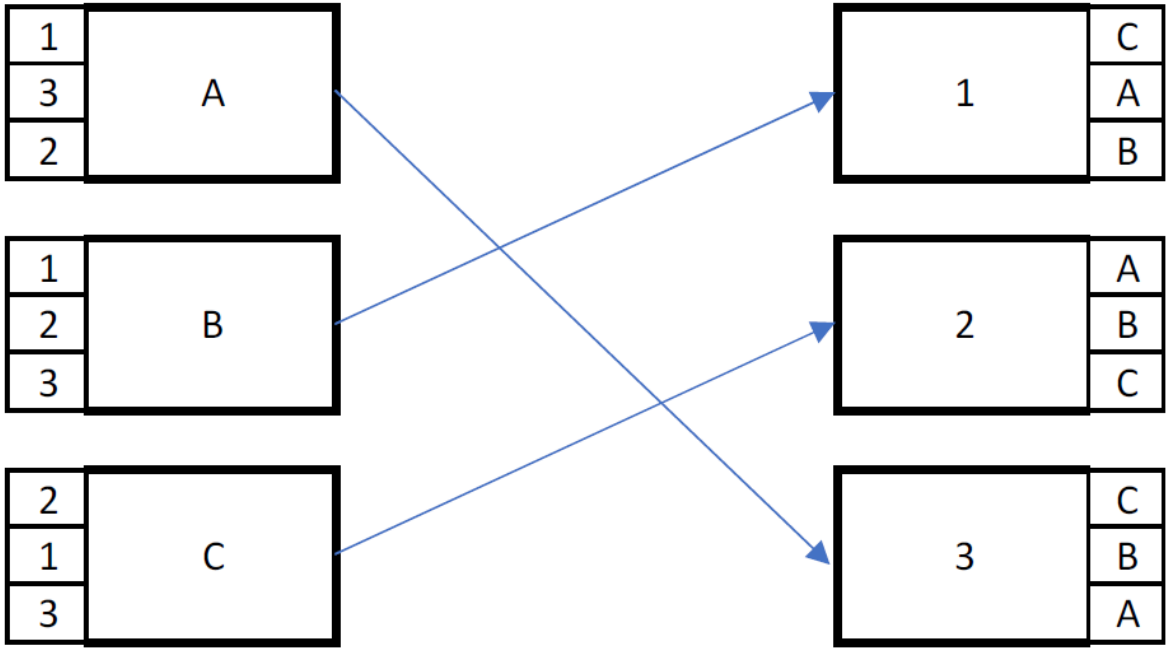
Algorithme non supervisé (*dynamique libre*)

- Initialisation : $C \leftarrow$ un appariement parfait arbitraire
- Tant qu' il existe une paire critique $(a, b) \in C$:
  - $a' \leftarrow$ partenaire de b dans C
  - $b' \leftarrow$ partenaire de a dans C
  - $C \leftarrow C \setminus (a', b), (a, b')$
  - $C \leftarrow C \cup (a, b), (a', b')$

Si cet algorithme termine, l'appariement construit est stable.

Algorithme non supervisé (*dynamique libre*)

Exemple :



Algorithme non supervisé (*dynamique libre*)

- Initialisation : $C \leftarrow$ un appariement parfait arbitraire
- Tant qu' il existe une paire critique $(a, b) \in C$:
  - $a' \leftarrow$ partenaire de b dans C
  - $b' \leftarrow$ partenaire de a dans C
  - $C \leftarrow C \setminus (a', b), (a, b')$
  - $C \leftarrow C \cup (a, b), (a', b')$

Si cet algorithme termine, l'appariement construit est stable.

Malheureusement, il ne termine pas toujours...

Algorithme d'acceptation différée (DA, Gale & Shapley 1962)

L'un des deux groupes joue le rôle des *proposants*, l'autre des *disposants*

- tant qu'il reste un proposant non apparié:
  - il propose au disposant qu'il préfère parmi ceux à qui il n'a pas déjà fait de proposition.
  - le disposants abordé répond "non" si la proposition ne lui convient pas, ou bien "peut-être" si la proposition lui convient, en rejetant éventuellement l'offre de leur partenaire actuel.
- les propositions sont finalisées (les "peut-être" deviennent des "oui")

Première remarque

L'algorithme AD n'est pas présenté de manière déterministe : l'ordre dans lequel on considère les proposants n'est pas spécifié.

> *Est-ce important ?*

Implémentation

Il faut des structures de données permettant :

- de tenir à jour l'ensemble des proposants non appariés

  > *un ensemble $X \subset E$ peut être représenté par un `map` à clés dans E, ou par un tableau de booléens de taille $|E|$…*

- de tenir à jour, pour chaque proposant, la liste des proposants à qui il n'a pas encore fait d'offre, ordonnée par préférence

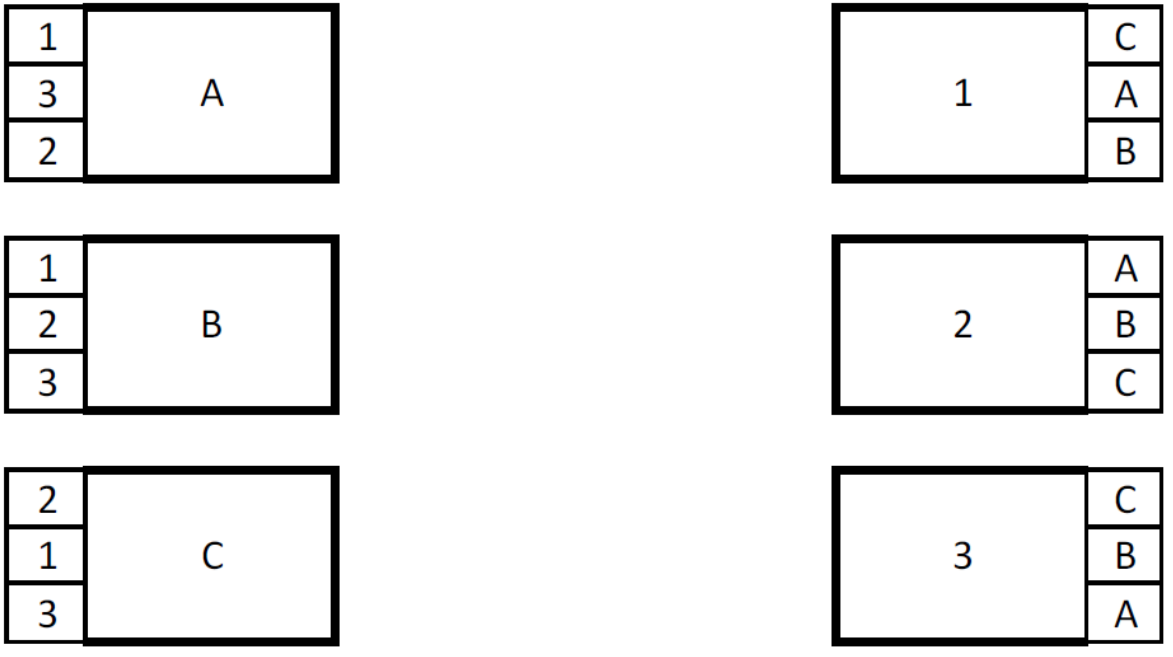
  > *on peut utiliser des listes chaînées, des slices, ou bien des entiers dénotant le rang du dernier admis, associés à un agent proposant via un tableau ou un map*

- de tenir à jour l'appariement, avec un accès rapide par disposant

  > *tableau ou map…*

Algorithme d'acceptation différée

Exemple :



Algorithme d'acceptation différée

- *tant qu'il reste un proposant non apparié:*
  - *il propose au disposant qu'il préfère parmi ceux à qui il n'a pas déjà fait de proposition.*
  - *le disposants abordé répond "non" si la proposition ne lui convient pas, ou bien "peut-être" si la proposition lui convient, en rejetant éventuellement l'offre de leur partenaire actuel.*
- *les propositions sont finalisées (les "peut-être" deviennent des "oui")*

Terminaison

Théorème : L'algorithme AD termine en $O(n^2)$ tours de la boucle *tant que*

dém : Ici, la boucle *tant que* est amené à considérer un couple $(a, b) \in A \times B$ à chaque itération. Un même couple ne peut être considéré deux fois. On en déduit donc qu'il y au plus $|A \times B| = n^2$ itérations.

Algorithme d'acceptation différée

- *tant qu'il reste un proposant non apparié:*
  - *il propose au disposant qu'il préfère parmi ceux à qui il n'a pas déjà fait de proposition.*
  - *le disposants abordé répond "non" si la proposition ne lui convient pas, ou bien "peut-être" si la proposition lui convient, en rejetant éventuellement l'offre de leur partenaire actuel.*
- *les propositions sont finalisées (les "peut-être" deviennent des "oui")*

Terminaison

Corollaire : si tant est que chaque opération (identifier le meilleur élément, vérifier une préférence, former ou défaire un couple de l'appariement, itérer sur une liste de préférences) puisse être réalisé en temps constant (indépendant de n), la complexité temporelle de l'algorithme est en $O(n^2)$.

Algorithme d'acceptation différée

- *tant qu'il reste un proposant non apparié:*
  - *il propose au disposant qu'il préfère parmi ceux à qui il n'a pas déjà fait de proposition.*
  - *le disposants abordé répond "non" si la proposition ne lui convient pas, ou bien "peut-être" si la proposition lui convient, en rejetant éventuellement l'offre de leur partenaire actuel.*
- *les propositions sont finalisées (les "peut-être" deviennent des "oui")*

Correction

Il faut vérifier que la réponse de l'algorithme constitue: i) un appariement, ii) parfait, iii) stable

Algorithme d'acceptation différée

- *tant qu'il reste un proposant non apparié:*
  - *il propose au disposant qu'il préfère parmi ceux à qui il n'a pas déjà fait de proposition.*
  - *le disposants abordé répond "non" si la proposition ne lui convient pas, ou bien "peut-être" si la proposition lui convient, en rejetant éventuellement l'offre de leur partenaire actuel.*
- *les propositions sont finalisées (les "peut-être" deviennent des "oui")*

Correction - i) non redondance

Th : A tout moment, l'ensemble C des couples constitue un appariement.

dém : par récurrence

- c'est vrai à l'état initial $C = \emptyset$
- tout proposant ajouté ne figurait pas dans C
- lorsqu'un disposant non libre est ajouté à C, le couple où il figurait précédemment est enlevé.

Algorithme d'acceptation différée

- *tant qu'il reste un proposant non apparié:*
  - *il propose au disposant qu'il préfère parmi ceux à qui il n'a pas déjà fait de proposition.*
  - *le disposants abordé répond "non" si la proposition ne lui convient pas, ou bien "peut-être" si la proposition lui convient, en rejetant éventuellement l'offre de leur partenaire actuel.*
- *les propositions sont finalisées (les "peut-être" deviennent des "oui")*

Correction - ii) perfection de l'appariement

Th : l'appariement final est parfait.

dém : c'est la condition de sortie de boucle.

Algorithme d'acceptation différée

> - *tant qu'il reste un proposant non apparié:*
>   - *il propose au disposant qu'il préfère parmi ceux à qui il n'a pas déjà fait de proposition.*
>   - *le disposants abordé répond "non" si la proposition ne lui convient pas, ou bien "peut-être" si la proposition lui convient, en rejetant éventuellement l'offre de leur partenaire actuel.*
> - *les propositions sont finalisées (les "peut-être" deviennent des "oui")*

Correction - iii) stabilité de l'appariement

Th : l'appariement final est stable.

> *Pour démontrer cette propriété importante, on va commencer par en démontrer deux autres, qui permettent de bien comprendre comment fonctionne AD.*

Correction - iii) stabilité de l'appariement

Th : Au cours de l'exécution, le sort de chaque proposant apparié ne fait qu'empirer.

dém : les proposants parcourent leur liste de préférence par ordre décroissant.

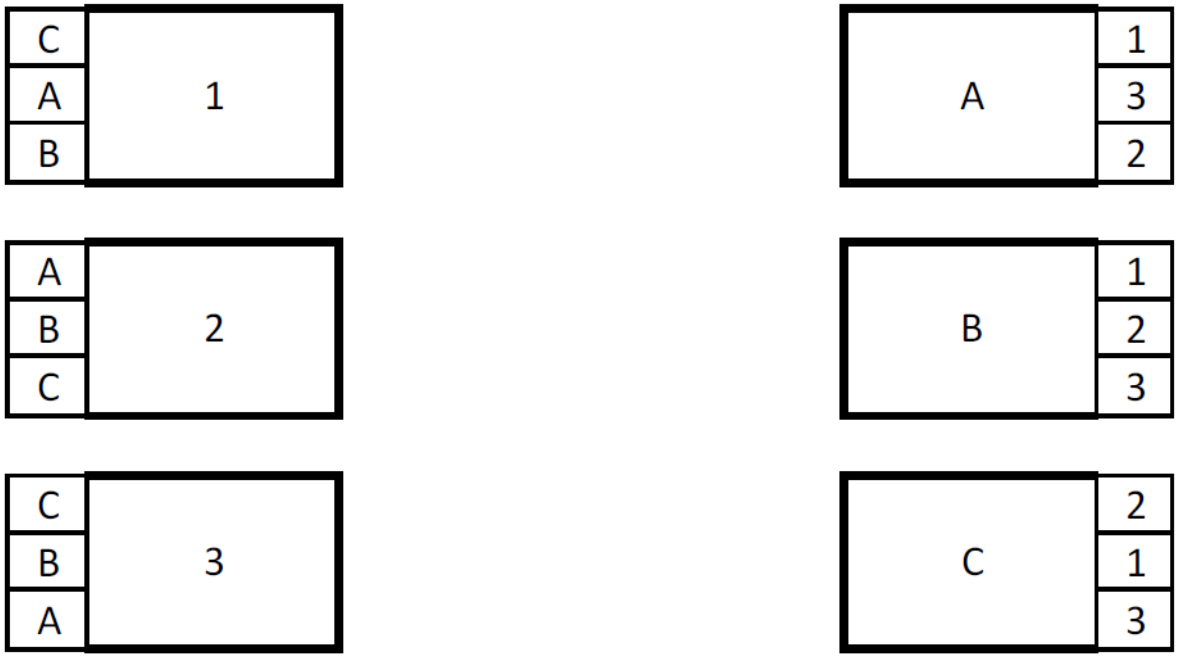
Th : Au cours de l'exécution, le sort de chaque disposant apparié ne fait que s'améliorer.

dém : par construction de la disposition, les disposants n'acceptent que des échanges qui leur sont favorables.

Th : l'appariement final est stable.

dém : Supposons le contraire, et soit $(p\star, d\star)$ une paire bloquante. Comme $p\star$ préfère $d\star$ à son partenaire final, il lui a déja fait une proposition. Soit $d\star$ l'a immédiatement rejetée, soit il a fini par en accepter une autre, mais dans tous les cas, il est apparié à un proposant qu'il préfère à $d\star$, ce qui contredit l'hypothèse d'instabilité.

Que se passe-t-il si l'on inverse proposants et disposants ?



Quelles différences avec le précédent appariement ?

*(In)*équité de la procédure AD

> *parmi tous les appariements stables, l'algorithme renvoie le plus favorable aux proposants, et le plus défavorable aux disposants*

Définition :

- pour tout proposant p, on note $best(p)$ le partenaire que p préfère parmi tous ceux qui lui sont accessibles dans un appariement stable.

Th : Quel que soit l'orde dans lequel les proposants sont considérés, l'algorithme AD retourne l'appariement

$$C^\star = (p, best(p)), p \in proposants$$

*(In)*équité de la procédure AD

Lemme : Au cours de l'exécution de AD :

1. chaque proposant p qui est apparié a un partenaire qu'il juge au moins aussi bon que $best(p)$

2. chaque proposant p qui n'est pas apparié n'a fait des propositions qu'à des disposants qu'il préfère strictement à $best(p)$

dém : Par récurrence : Initialisation, rien à démontrer. Hérédité : à la $n+1$-ème itération, p propose à d avec $d \succsim_p best(p)$.

- Si d accepte, on satisfait 1.
- Si d était appariée à p' on sait par $HR1$ que p' préfère d à $best(p')$.
- Si d quitte p' pour p, il reste à montrer que $d \neq best(p')$. Supposons un appariement stable dans lequel on aurait (p', d) et (p, d'). Comme p préfère d à $best(p)$ et $best(p)$ à d', (p, d) est critique.
- Si d est apparié avec p' et ne le quitte pas pour p, il reste à montrer que $d \neq best(h)$. Supposons un appariement stable avec (p, d) et (p', d'). Comme p' préfère d à $best(p')$, et $best(p')$ à d', (p', d) est critique.

*(In)*équité de la procédure AD

Définition :

- pour tout disposant d, on note $worst(d)$ le partenaire que d préfère le moins parmi tous ceux qui lui sont accessibles dans un appariement stable.

Th : On a aussi $C^\star = (worst(d), d), d \in disposants$.

dém : ??

Etude de cas : *Parcoursup*

Afin d'affecter les étudiants en première année de l'enseignement supérieur, le MESR recours depuis 2018 à une procédure semi-décentralisée :

- Algorithme *AD*
- Les *proposants* sont les établissements d'enseignement supérieur.
- Les *disposants* sont les candidats.
- L'algorithme est *public*. Le programme l'implémentant a été vérifié à l'aide d'outils formels.
- L'algorithme *évolue* d'année en année. En particulier, il intègre maintenant des critères d'"équité" géographique (voeux intra-académiques privilégiés) et sociale (prise en compte d'un taux d'étudiants boursiers) qui modifient profondément le comportement de l'algorithme

Etude de cas : *Parcoursup*

- Les *proposants* sont les établissements d'enseignement supérieur.
- L'algorithme centralisé prévoit qu'ils fournissent leurs préférences quant aux candidats sous la forme d'un ordre total.
- De manière décentralisée, il suffit qu'ils puisse à tout moment répondre à la requête *Candidat Suivant*.
- En pratique, ils remontent le classement complet (et strict) des candidatures au début de la procédure
- Les *disposants* sont les candidats.
  - L'algorithme centralisé prévoit qu'ils fournissent leurs préférences quant aux candidats sous la forme d'un ordre total.
- De manière décentralisée, il suffit qu'ils puissent à tout moment répondre à la requête *Préférence* les interrogeant quant à leur préférence entre deux établissements.
- en pratique, ils doivent répondre à une question *Accepter l'offre*, en répondant soit *Oui, immédiatement*, soit *Oui, pour le moment*, soit *Non, définitivement*

Parcoursup : Pourquoi tant de haine ?

- procédure (semi-) décentralisée ?
- proposants = établissements ?
- classement complet des candidats par les établissements ?
- comparaisons par paires des établissements par les candidats ?
- "Oui, immédiatement" / "Oui, pour le moment" / "Non, définitivement" ?
- critères d'équité ?

Top Trading Cycles

Vu la semaine dernière dans le cadre des marchés simples.

A été adapté par Abdulkadiroglu & Sonmez en 2003 aux marchés doubles :

> *Tant qu'il reste des agents libres (non appariés) :*
>
> - *chaque agent du groupe A pointe vers son agent libre préféré du groupe B*
> - *chaque agent du groupe B qui a reçu une offre pointe vers son agent libre préféré du groupe A*
> - *identifier un cycle d'échange, i.e. une liste d'agents , $[a_1, b_1, \ldots a_k, b_k]$, tel que l'agent a_i pointe l'agent b_i qui pointe l'agent a_{i+1} et que l'agent b_k pointe l'agent a_1*
> - *apparier chaque agent b_i du cycle avec l'agent qu'il pointe*

Cet algorithme fonctionne-t-il ? termine-t-il ? fournit-il un appariement ? parfait ? stable ? favorise-t-il un des deux groupes ? peut-il être décentralisé ?