Théorie des jeux algorithmique – TP

Équilibres de Nash

Dernière modification: 15 novembre 2023

L'objectif de ce TP est de calculer l'ensemble des équilibres de Nash d'un jeu à deux joueurs sous forme normale, avec un nombre quelconque d'actions.

On procède par *énumération des supports*, combinée avec l'élimination itérée des actions (stratégies pures) strictement dominées étant donné le support σ_{-i} de l'adversaire.

Définition 1 (Domination stricte conditionelle)**.** Un action $a_i \in A_i$ du joueur i est strictement dominée pour le support $\sigma_{-i} \subseteq A_{-i}$ des autres joueurs, s'il existe une autre action $a' \in A_i$, telle que :

$$\forall b \in \sigma_{-i}, u_i(a, b) < u_i(a', b)$$

En effet les actions strictement dominées quand l'adversaire joue une stratégie de support σ_{-i}, ne peuvent pas apparaître dans un équilibre de Nash dans lequel l'adversaire joue une stratégie de support σ_{-i} : on peut donc les éliminer.

On remarque en outre que si a' domine strictement a pour le support σ_{-i} alors c'est toujours le cas pour tout support $\sigma'_{-i} \subseteq \sigma_{-i}$.

Cela nous donne donc un algorithme récursif :

1. on part de supports qui contiennent toutes les actions des deux joueurs ;
2. on élimine de manière itérative les stratégies des deux joueurs qui sont strictement dominées pour le support de l'adversaire. Une action éliminée d'un joueur peut créer une nouvelle domination stricte pour son adversaire.
3. on calcule par programmation linéaire les équilibre de Nash correspondant aux deux supports résultants, s'ils sont non-vides tous les deux. Si l'un des deux est vide, c'est le cas de base de la récursion : on ne peut plus avoir d'équilibres de Nash avec ces supports, ou des supports plus petits, et on s'arrête sur cette branche.
4. pour chacune des actions des deux supports, on crée une nouvelle paire de supports ne contenant pas cette action, et on appelle l'algorithme récursivement.

On va implémenter cet algorithme. Le fichier fourni `nash.py` contient la définition d'une classe `Game` représentant un jeu à deux joueurs. Notez que dans cette classe la matrice d'utilités du deuxième joueur est transposée, de façon à ce que chaque joueur joue dans sa matrice selon les lignes.

Les méthodes `strictly_dominated` et `irsds` implémentent l'élimination itérée des stratégies strictement dominées (inconditionellement) et pourront vous servir d'inspiration.

L'algorithme esquissé ci-dessus est déjà implémenté dans la méthode `nash_equilibria`, il reste à écrire les méthodes appelées dedans.

On propose de représenter les supports des deux joueurs par un couple de listes contenant les indices des actions contenues dans le support.

Q1. Ajouter une méthode `strictly_dominated_as` qui pour un joueur p $\in \{0, 1\}$, une action `a`, et un couple de supports `sigma`, renvoit `True` si l'action `a` du joueur `p` est strictement dominée pour le support $\sup[1 - p]$;

Q2. Ajouter une méthode `irsda_support` qui étant donnée un couple de supports élimine [1] de façon itérée les actions strictement dominées des actions des supports de chaque joueurs, étant donné le support de son adversaire ;

Q3. Ajouter une méthode `nash_support` qui étant donné un couple de supports renvoit un équilibre de Nash basé sur ce support, s'il y en a un.

Q4. Tester sur quelques jeux : un jeu sans équilibre pur, un jeu sans équilibre mixte, un avec les deux, etc.

1. Attention ! Contrairement à `irsds` les actions ne sont pas supprimées dans le jeu lui-même, seulement dans le support.