



Théorie des jeux

Matt TAYLOR, Yoann SOCHAJ, Benjamin MAUREL

Mai 2022

1 Introduction

Ce projet de Théorie des Jeux s'intéresse à analyser des jeux de 2 à n joueurs, par exemple être capable de déterminer l'équilibre de Nash ou si le jeu est à somme nulle ou non. Nous avons décidé de regrouper les trois exercices demandés au sein du même environnement.

2 Environnement & implémentation

2.1 Environnement

Nous avons décidé d'utiliser le langage **Python** pour ce projet ainsi que le module **Tkinter** pour l'affichage et le module **matplotlib** pour afficher nos résultats sous forme de graphes.

2.2 Notre implémentation

Tout d'abord, nous nous sommes mis d'accord sur un format de données qui permettra la facilité d'accès au sein d'un joueur à tout ses gains potentiels. Ainsi, nous avons opté sur une classe joueur contenant une liste de stratégies. Cette liste est composée de autant de sous-liste que de stratégie de ce joueur. Chaque sous-liste permet de donner la valeur associée à la stratégie des autres joueurs.

Cela nous donne un format comme ceci:

```
00:  
[[-4, -2, -4], [-4, -1, 1], [-3, -1, 0]]  
11:
```

Figure 1: Strategies du joueur 0 dans un jeu à 2 joueurs 3 strategies

Dans ce cas la, on a, pour le joueur 0, 3 stratégies donc 3 sous-listes. Il n'y a qu'un seul autre joueur donc il y a 3 éléments dans chaque sous liste.

La méthode *simulate()* permet de simuler le jeu sur une centaine de réalisations avec l'équilibre de Nash mixte avec les paramètres de la stratégie mixte donnée par l'utilisateur. Il faut tout d'abord vérifier que la **somme des probabilités** des deux stratégies soit **égale à 1**. Ensuite on se sert du résultat de la méthode *equilibreDeNashMixte()* (décrite plus bas dans le rapport), s'il existe un équilibre de Nash mixte alors nous pouvons assigner les probabilités des stratégies du joueur 1 aux valeurs entrées par l'utilisateur et celles du joueur 2 aux résultats de la méthode *equilibreDeNashMixte()*. Puis on **boucle 100 fois**, à chaque itération on génère un float **random** pour savoir **quelle stratégie sera utilisée** et on **met à jour** les listes de **gains des deux joueurs**. Nous avons également la possibilité de **consulter les gains obtenus à chaque étape du jeu** en visualisant le graphe matplotlib qui affiche les gains des deux joueurs sur les 100 itérations de la boucle ceci permet de vérifier expérimentalement si la stratégie mixte obtenu par l'équilibre de Nash est optimale ou pas. Pour que ce graphe s'affiche il faut d'abord fermer le graphe résultat de la méthode *equilibreDeNashMixte()* car matplotlib affiche les graphes un par un.

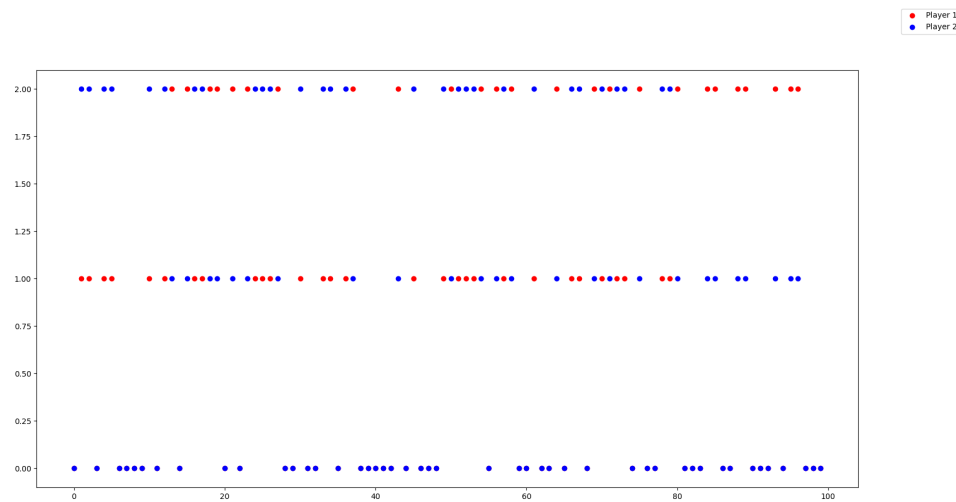


Figure 2: Gains obtenus à chaque étape du jeu pour chaque joueur

Pour l'équilibre de Nash, on applique la même méthode quelque soit le nombre de joueurs. Dans un premier temps, on liste toutes les possibilités de manière récursive car étant donnée que l'on ne connaît pas le nombre de joueur à l'avance, on ne pouvait pas juste faire un nombre de boucles prédéfini. Pour ce faire, on descend via une récursion dans la matrice de jeu. Cela donne une liste qui ressemble a ça :

```
[0, 0], [0, 1], [0, 2],
[1, 0], [1, 1], [1, 2],
[2, 0], [2, 1], [2, 2]]
```

Figure 3: Liste des possibilités d'un jeu 2 joueurs avec 3 stratégies

Une fois avec ce résultat, on va parcourir cette liste des combinaisons possible et regarder via la liste des stratégies de chaque joueur si quelqu'un a intérêt unilatéralement de changer sa stratégies. Tout les résultats qui n'admettent aucune amélioration sont donc des équilibres de Nash qu'on ajoute a une liste et qu'on retourne.

Le problème de cette méthode est que leur complexité est élevé. En effet, nous sommes obliger de parcourir toute les cases de la matrice. Avec un grand nombre de joueur et de stratégies, le nombre de cellule est cependant nous avons pas réussi a trouvé de modèle plus adapté. La complexité peut également être diminuer en enlevant de l'équation l'ensemble des stratégies dominées mais notre modèle de données ne permet pas ce genre d'optimisation.

Nous avons aussi le calcul de l'équilibre de Nash en stratégie mixte qui fonctionne. Pour l'implémenter nous avons simplement adapté les exemples du cours et vu en TD afin de le faire fonctionner sur un ordinateur. Dans un premier temps on calcule l'utilité des deux joueurs avec leurs deux stratégies. On va donc avoir l'utilité des stratégies, comme dans le cours on pourra donc tracer deux courbes pour les deux joueurs. Et l'intersection entre ces deux courbes est l'équilibre de Nash en stratégie mixte non pure.

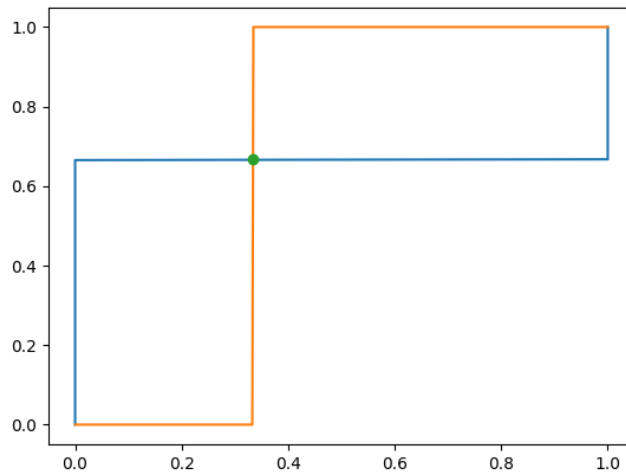


Figure 4: Résultat sur l'exemple de la bataille des sexes

Pour finir, nous avons la méthode *strategieDomine()*. Notre modèle est particulièrement efficace pour traiter ce problème. En effet étant donnée que les différents gains sont stocké dans le joueur. Nous avons juste a comparer les sous-liste correspondant au stratégies entre elles. De ce fait si tout les éléments d'une sous-liste sont supérieur a une autre, on peut dire qu'une stratégies est dominé. Si elle domine toutes les autres, on peut dire qu'elle est dominante.

3 Usage & Résultats

Pour lancer le programme il suffit de lancer le fichier *main.py*. Il faut tout d'abord créer un jeu, pour cela on doit d'abord créer des joueurs, on spécifie le nom du joueur ainsi que son nombre de stratégies puis on clique sur le bouton *Create Player*. Un message de confirmation apparaît dans la console.

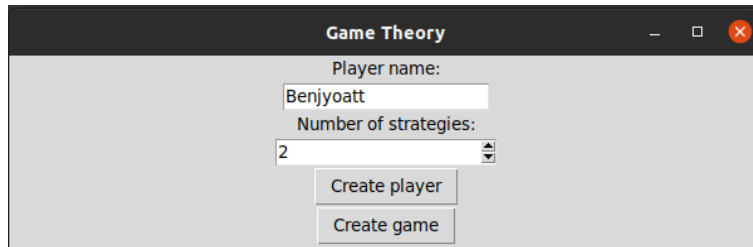


Figure 5: Create players

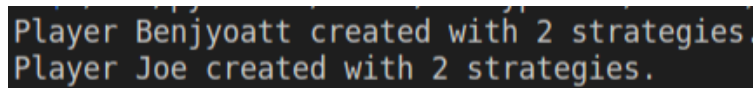


Figure 6: Player successfully created

Nous pouvons maintenant créer un jeu en cliquant sur *Create game*, cela fera apparaître une **liste d'options** ainsi que la matrice du jeu. Cette matrice, généré aléatoirement, s'**affichera uniquement** lorsqu'il s'agit d'un jeu comportant **deux joueurs** - l'affichage n'est pas disponible lorsqu'il s'agit de trois joueurs ou plus. Les boutons *Mixed Nash equilibrium* et *Create a mixed strategy* ne marchent que pour des jeux à 2 joueurs avec 2 stratégies chacun.

Il est également possible de **rentrer des valeurs manuellement dans la matrice** en positionnant le curseur sur l'une des cases de la matrice. Attention à bien respecter le format pour que les valeurs soient lues correctement [nombre, nombre] comme ceci : [1, -4].

- **Zero-sum game** : permet de déterminer si le jeu est à somme nulle.
- **Nash equilibrium** : permet de déterminer l'équilibre de Nash s'il existe.
- **Dominant & Dominated Strategies** : permet de déterminer les stratégies dominées et dominantes.
- **Mixed Nash equilibrium** : permet de déterminer l'équilibre de Nash mixte s'il existe.
- **Create a mixed strategy** : permet de créer une stratégie mixte en rentrant la probabilités des deux stratégies.

Voici les résultats pour la matrice de la figure 7 : Remarque: il n'y a pas d'équilibre de Nash mixte non pur pour la matrice de la figure 7 donc la simulation n'est pas possible.

The screenshot shows a window titled "Game Theory" with a dark header. Below the header, there are several input fields and buttons. The "Player name:" field contains "Joe". The "Number of strategies:" field is a dropdown menu set to "2". Below these are buttons for "Create player", "Create game", "Matrix", "Zero-sum game?", "Nash equilibrium", "Dominant & Dominated Strategies", "Mixed Nash equilibrium", and "Create a mixed strategy". The "Matrix" field displays a 2x2 grid of values: $\begin{bmatrix} -5 & -4 \\ 2 & -3 \end{bmatrix}$ and $\begin{bmatrix} -2 & 2 \\ -1 & -1 \end{bmatrix}$.

Figure 7: Game successfully created

Zero-sum game: False

Figure 8: Zero-sum game?

Nash: $\begin{bmatrix} -1 & -1 \end{bmatrix}$

Figure 9: Nash equilibrium

Pour le joueur Benjyoatt , la strategie 0 est dominee par la strategie 1 ;
 La strategie 1 du joueur Benjyoatt est dominante.
 Pour le joueur Joe , la strategie 0 est dominee par la strategie 1 ;
 La strategie 1 du joueur Joe est dominante.

Figure 10: Dominant & Dominated strategies

The screenshot shows a dialog box titled "Create a mixed strategy". It contains two input fields: the first is set to "0.2" and the second is set to "0.8". Below these fields is a button labeled "Simulate".

Figure 11: Create a nixed strategy

4 Conclusion

Ce projet a été l'occasion pour nous de mieux comprendre, d'essayer d'optimiser le concept d'équilibre de Nash sur ordinateur et de mieux cerner ce qu'était un équilibre mixte. Il a également pour nous été l'occasion de nous casser la tête sur l'adaptation de notre programme 2 joueurs aux contraintes à N joueurs.