MATH2309P – Projet

À rendre pour le 12 juin 2022

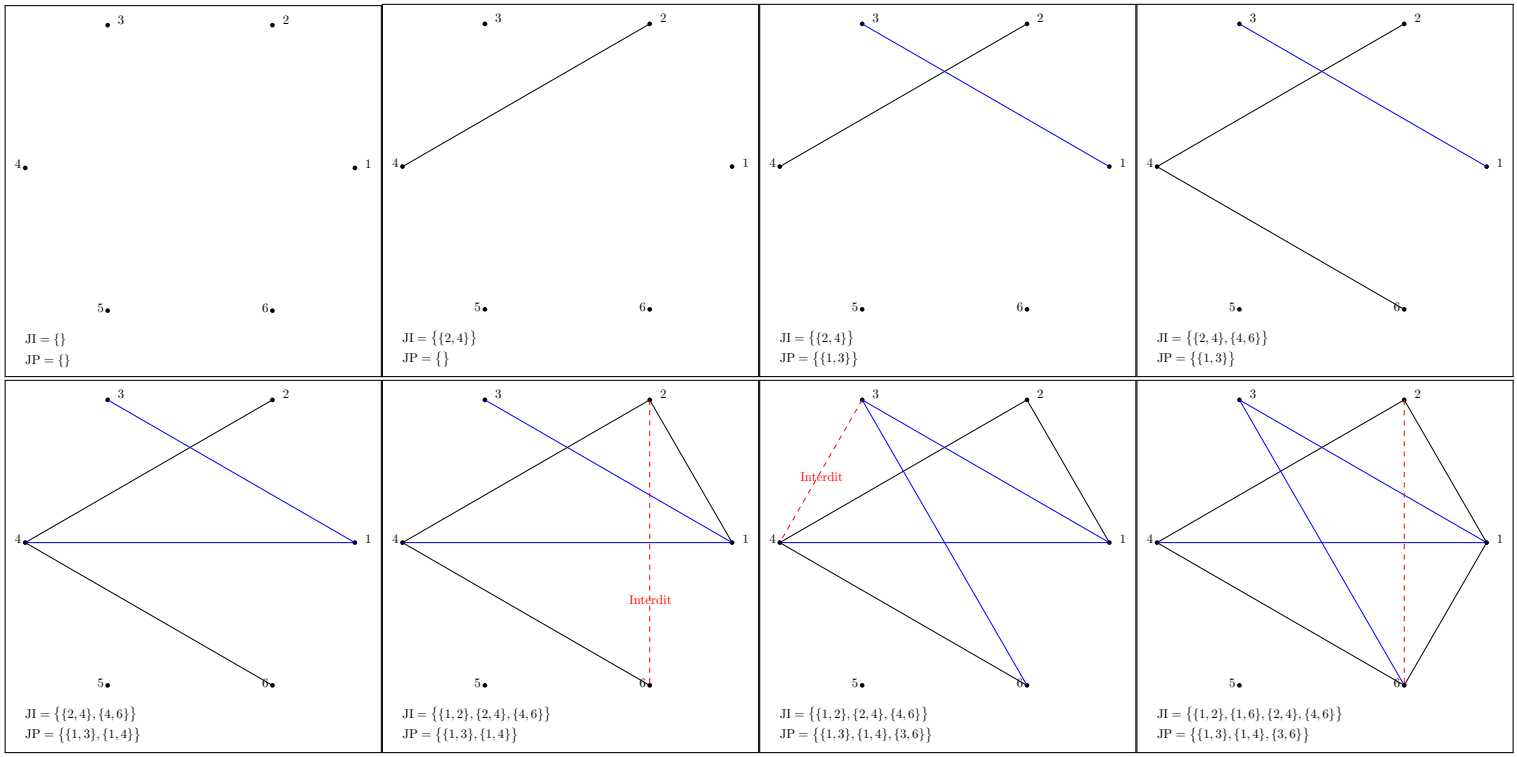
L'objectif de ce petit projet est de fabriquer une petite IA qui va jouer à un jeu et d'essayer de lui apprendre à mieux jouer. Pour des raisons évidentes, nous nous limiterons aux outils étudiés dans le cours.

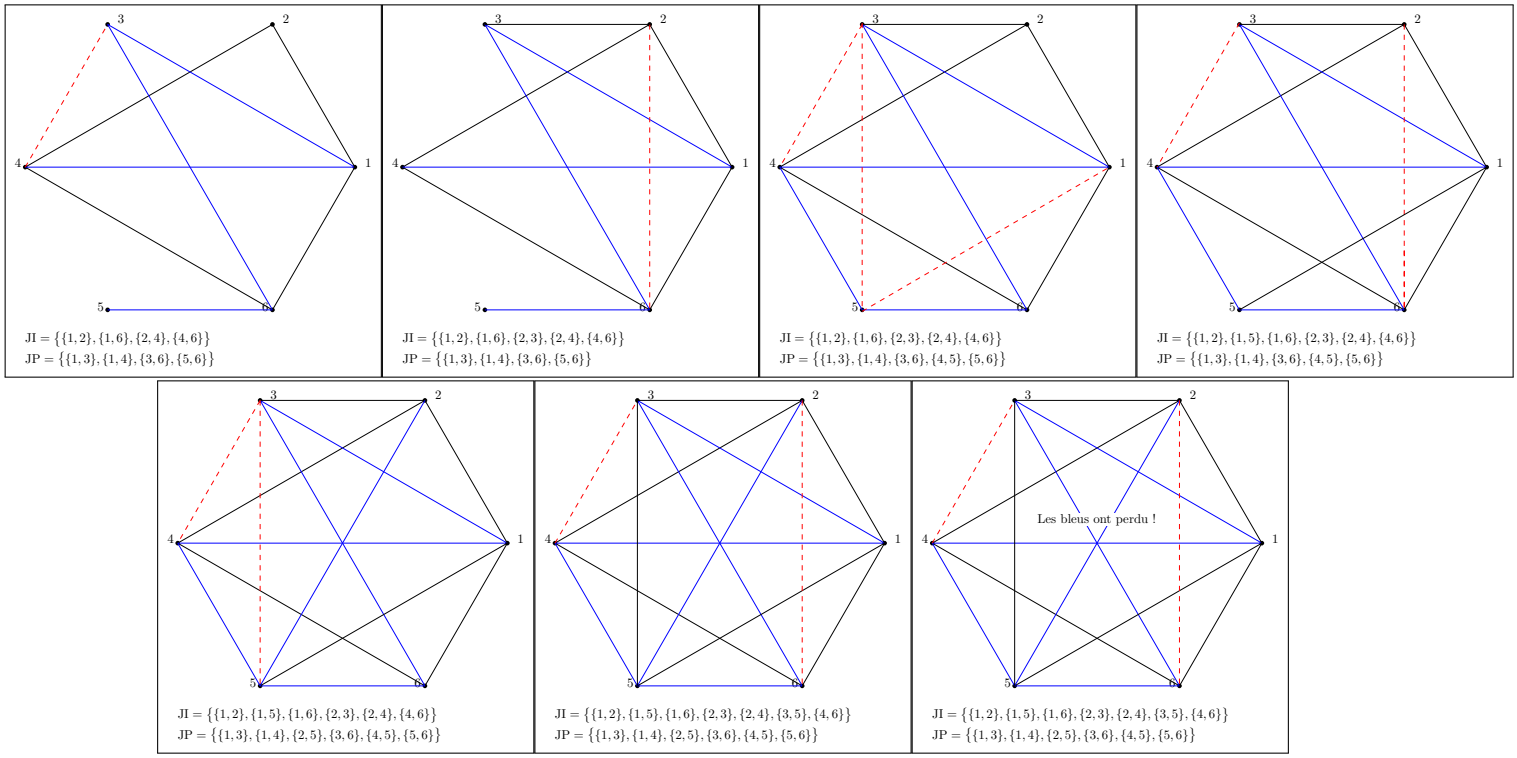
1 Description du jeu

Au départ, nous avons un hexagone régulier, les sommets étant numérotés de 1 à 6. Le jeu se joue à deux joueurs que nous noterons I (comme *impair*) et P (comme *pair*). Le joueur I commence la partie et chaque joueur joue chacun son tour.

1. À son tour, le joueur trace un trait reliant deux des six sommets de l'hexagone qu'il trace avec sa couleur (dans les graphiques suivants, le joueur I joue avec la couleur noire et le joueur P joue avec la couleur bleue), avec les règles suivantes
   (a) il n'a pas le droit de tracer un trait déjà tracé ;
   (b) il n'a pas le droit de faire un triangle *dont les sommets sont trois des six points de l'hexagone* tracé avec sa couleur (mais il a le droit de faire un triangle avec des traits de la couleur de l'adversaire).
2. Un joueur a perdu quand il ne peut plus tracer de trait !

Voici un exemple de partie (on note JI les traits joués par le joueur I et JP les traits joués par le joueur P), les traits en pointillés rouges sont les traits interdits *avant* le trait joué.





2 Contraintes

Les parties seront conservées comme une liste de coups numérotés.

In[1]

```
coups = []
for i in range(1, 7):
    for j in range(i+1, 7):
        coups.append({i, j})
print(coups)
```

```
[{1, 2}, {1, 3}, {1, 4}, {1, 5}, {1, 6}, {2, 3}, {2, 4}, {2, 5}, {2, 6},
    {3, 4}, {3, 5}, {3, 6}, {4, 5}, {4, 6}, {5, 6}]
```

Ainsi, la partie ci-dessus sera décrite par la suite, où le numéro correspond à l'indice du trait dans la liste `coups` et 30 correspond à une impossibilité de jouer de manière à ce que chaque partie possède 14 traits

$$[6, 1, 13, 2, 0, 11, 4, 14, 5, 12, 3, 7, 10, 30]$$

3 Ce qui est demandé

1. Montrer qu'il n'y a pas de partie nulle (soit I gagne, soit P gagne).
2. Générer 1000 parties en choisissant à chaque fois aléatoirement le trait joué et les stocker dans une matrice 1000×14.
3. Faire une ACP produisant une projection sur le plan principal et interpréter ce graphique.
4. Faire une analyse discriminante séparant les parties où I gagne des parties où P gagne.
5. En déduire une fonction d'évaluation affine permettant d'effectuer un choix de coup en fonction de la situation.

6. Programmer une IA permettant de jouer à l'aide d'un $\alpha - \beta$.
7. En utilisant les techniques du *fitting*, proposer une autre fonction d'évaluation meilleure.

On produira finalement une fonction `choix` dont les paramètres seront JI et JP et qui renvoie un numéro de coup à jouer.

```
def choix(JI, JP):
    ...
    return no  # Numéro du coup à jouer
```

Pour ceux qui veulent y participer et uniquement pour le plaisir, j'organiserai un tournoi entre les différentes fonctions choix proposées.

4 Avertissement

1. Les projets se feront par groupe de 3 ou 4 (à déclarer la semaine prochaine).
2. Toute communication entre les groupes est *interdite*.
3. Toute tricherie (communication entre groupe, échange de code, recopiage de code sur Internet) sera sévèrement punie.

Ce qui est important dans le projet est

1. un travail sérieux ;
2. une grande honnêteté (pas de tricherie) ;
3. la qualité du rapport et des codes.

Ce qui est moins important est

1. la qualité de la fonction `choix` trouvée à la fin.

5 Bilan

Date limite : 12 juin 2022 à 23h59.
Fournir deux fichiers (un rapport en `.pdf` et des codes commentés en `.ipynb`).