Bordeaux INP / Enseirb-Matmeca Ingé 2 2021/2022

# Intelligence Artificielle – Semestre 2

**Le grand tournoi des Ingé 2 sur un plateau**

---

Vous devez rendre un joueur qui puisse s'intégrer au tournoi entre joueurs. Dans le tournoi, chacun va rencontrer tous les autres dans deux matchs (en tant que Noirs puis en tant que Blancs)).

**Notations** : un certain nombre de points (à titre indicatif, 4 points) seront directement donnés par rapport à votre classement dans le tournoi. 3 autres points seront donnés par rapport à votre capacité à battre des IA de différent niveaux (joueur aléatoire, minimax niveau 1, minimax niveau 3, ...). Le reste des points sera donné suivant le code que vous aurez rendu (expliquez vos méthodes dans le code !) et les techniques que vous aurez employées. Il faudra nous rendre, dans l'archive un fichier README (txt ou md) qui contiendra une description des points forts de votre joueur (faites court, listez les techniques, décrivez l'heuristique codée, précisez les structures de données, évoquez comment vous avez décidé d'utiliser telle ou telle technique, mettez en avant ce dont vous êtes le plus fier...). **Nous serons très attentif au fait que vous ayez écrits vous-même votre code.** Le projet sera à rendre via le rendu MOODLE associé à ce projet. Soyez également vigilants aux instructions données via Moodle.

## Règles du jeu

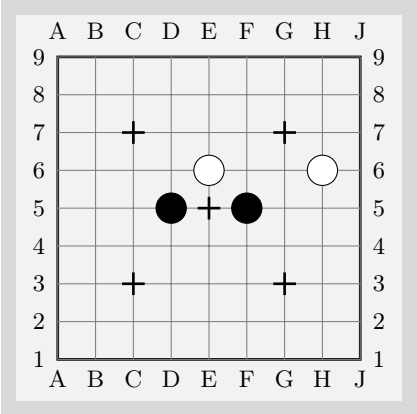


FIGURE 1 – Système de coordonnées et exemple de plateau de Go en 9x9. Notez qu'il n'y a pas de 'I'.

Les règles du GO peuvent être assez complexes à comprendre au départ, mais ce sera l'occasion d'apprendre ! Pour autant, le but de ce projet n'est pas de vous rendre des experts du GO mais de vous faire programmer une IA qui, elle, sait battre un débutant (donc probablement vous). Il existe de nombreuses règles et de nombreuses stratégies enseignées, mais vous devrez vous concentrer sur l'algorithmique. Suivant vos envies, vous pourrez coder des heuristiques demandant plus ou moins de connaissances, mais comme le calcul de ces heuristiques a un coût, on ne peut prédire qui va gagner entre le spécialiste du GO et le spécialiste en algorithmes.

— Pour appréhender les règles, vous pouvez vous référer à l'excellent site `https://senseis.xmp.net`. Pour la simplicité de ses règles, nous utiliserons la formalisation du GO décrite dans les règles dites de `Tromp-Taylor`, que vous trouverez ici `https://senseis.xmp.net/?TrompTaylorRules`,

ou encore ici `http://webdocs.cs.ualberta.ca/~hayward/396/hoven/tromptaylor.pdf` (principalement pour faire le décompte des points à la fin).

— Pour jouer, vous pourrez essayer l'excellent programme Gnugo, et on ne peut que vous recommander de lancer quelques parties, par exemple pour vous donner une idée des ouvertures possibles. Pour lancer Gnugo avec ces règles il faudra faire :

```
gnugo --chinese-rules --boardsize 9 --never-resign
```

Les règles du jeu sont implantés dans un code Python donné, comme pour les échecs. Ce code vous fournira les coups autorisés, ce qui simplifiera grandement (énormément) votre travail. Vous pourrez vous inspirer de certaines parties de ces règles, ou même modifier le code, pour que le fichier `Goban.py` calcule aussi vos heuristiques.

### Détails du tournoi

Le classement final se fera suivant un nombre total de point collectés pendant le tournoi. Chaque projet rencontrera chaque autre projet. Chaque rencontre donnera 3 points en cas de victoire, 1 point en cas d'égalité et 0 point en cas de défaite. Celui qui aura le plus de point remportera le tournoi.

Chaque joueur **n'aura le droit qu'à 15 minutes de temps réel de réflexion sur toute la partie**. Il est interdit à un joueur de consommer du temps CPU en dehors des appels explicites aux fonctions de l'interface (vous ne devez par exemple pas répondre à la méthode `getPlayerMove()` en laissant trainer un thread après votre réponse). Si votre joueur consomme, sur une partie, plus de 15 minutes de temps réel, alors vous perdez la partie et l'autre est immédiatement déclaré vainqueur. Et cela même si celui qui a fait un Timeout était bien mieux placé pour gagné à ce moment là. Soyez donc extrêmement vigilant sur votre consommation de temps.

## 1 Interface de votre joueur

Pour pouvoir être interfacé avec les autres joueurs, votre IA doit implanter l'interface donnée dans le script `playerInterface.py` dont le code est donné figure 2. Les explications de chaque méthode sont données dans le fichier. Un exemple de joueur aléatoire implémentant l'interface est donné dans le fichier `myPlayer.py`.

Cette interface est importante : votre joueur sera mis en réseau grâce à un script python qui appellera vos fonctions. Pour vous donner une idée de comment le tournoi sera organisé, deux autres scripts python (`localGame.py` et `namedGame.py`) permettent de connecter deux joueurs via leurs interfaces. Lisez bien le fichier `README` donné dans l'archive car elle donne des informations supplémentaires sur tous les fichiers donnés. Le script `localGame.py` ne sera pas utilisé pour le tournoi car chaque joueur n'est pas correctement isolé dans un processus indépendant. Durant le tournoi, les joueurs communiquerons en réseau avec l'arbitre, qui ne communiquera que des coups valides. Tout sera géré automatiquement via l'interface donnée mais pour cela vous devez bien respecter les consignes (sans quoi vous ne pourrez pas entrer dans le tournoi).

## 2 Modalités de rendu du tournoi

Le projet est à rendre en binômes. Les noms doivent être clairement spécifiés dans le fichier README donné dans votre archive. Cette archive (détaillée ci dessous) devra être rendue via Thor. La date de rendu et sera donnée sur le MOODLE du cours.

Vous devez rendre une archive contenant un répertoire ayant votre nom d'équipe (sans espaces). Cette archive devra contenir tous les fichiers nécessaires au lancement de votre IA (donc notamment `Goban.py`, `localGame.py`, ...). Si on lance `python localGame.py` (sans une ligne de modifiée) dans votre répertoire, cela doit lancer un match de votre I.A. contre elle même (le fait de ne pas modifier ce script permet de garantir que vous avez bien respecté les contraintes de l'API). C'est à partir de ce fichier que nous vérifierons, par exemple, quelle classe Python doit être chargée pour implanter l'interface playerInterface. Vous donnerez également, comme indiqué plus haut, un fichier README.TXT décrivant les techniques utilisées dans votre I.A. Même si ce fichier est simple, il est important de bien le renseigner pour que l'on puisse évaluer votre travail et l'originalité de votre approche. Vous devrez modifier le fichier `myPlayer.py` pour qu'il implante votre propre joueur. Attention : vous n'avez pas le droit d'écrire dans votre répertoire pendant le match. Le tournoi pouvant lancer plusieurs instances de votre I.A. dans différents matchs sur la même machine votre répertoire sera utilisé en lecture seule (vous pourrez écrire dans /tmp mais il faudra bien penser à nommer vos fichiers de manière unique pour autoriser plusieurs instances de votre programme à être lancées en parallèle). Les sorties standards seront sauvegardées pour afficher les matchs (mettez y des informations importantes). Par exemple, si vous voulez faire un match de votre IA contre le random player, il vous faudra copier `myPlayer` en `randomPlayer` et copier `localGame` en un nouveau fichier pour qu'il charge le joueur aléatoire avec le votre.

**Important :** pour le rendu, `localGame`, dans sa version originelle, non modifiée, doit ne lancer que votre I.A. contre elle-même et `myPlayer` doit implanter **votre** joueur, c'est à dire le joueur qui devra entrer dans le tounoi. Il est déconseillé, pour la version tournoi, de laisser un constructeur ayant des paramètres. Si vous décidez de laisser des paramètres (comme la profondeur maximal de votre minimax) à donner au constucteur, il faut absolument clarifier quels paramètres doivent être donnés.

Vous êtes libres d'écrire votre propre classe `Board` à la manière du script `Goban.py` pour accélérer votre I.A., mais il faut absolument laisser ce script intact pour que `localGame` se passe bien. Typiquement, si vous voulez le modifier, il faudra en faire une copie et renommer la classe et le fichier. Vous êtes libres d'organiser des tournois entre vous. Tous les matchs que nous lancerons seront accessibles sur internet. Attention, donc, à ce que vous écrivez.

```python
class PlayerInterface():
    # Returns your player name, as to be displayed during the game
    def getPlayerName(self):
        return "Not Defined"

    # Returns your move. The move must be a valid string of coordinates ("A1",
    # "D5", ...) on the grid or the special "PASS" move. a couple of two integers,
    # which are the coordinates of where you want to put your piece on the board.
    # Coordinates are the coordinates given by the Goban.py method legal_moves().
    def getPlayerMove(self):
        return "PASS"

    # Inform you that the oponent has played this move. You must play it with no
    # search (just update your local variables to take it into account)
    def playOpponentMove(self, move):
        pass

    # Starts a new game, and give you your color.  As defined in Goban.py : color=1
    # for BLACK, and color=2 for WHITE
    def newGame(self, color):
        pass

    # You can get a feedback on the winner
    # This function gives you the color of the winner
    def endGame(self, color):
        pass
```

FIGURE 2 – Interface à implémenter pour pouvoir jouer en tournoi