TraAM – intelligence artificielle

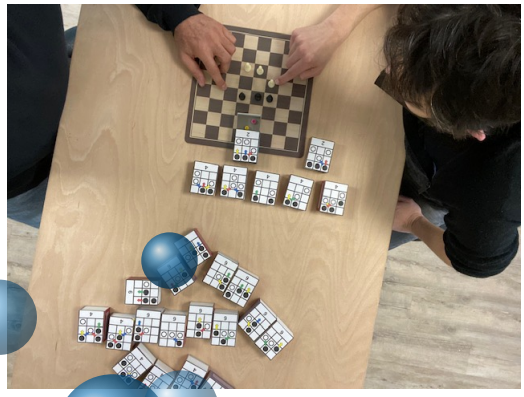
Thème National

Apprentissage machine à partir du jeu Hexap♙on

Prolongement par le groupe TraAM IA ac-nice.fr

Les mathématiques : moteur de l'intelligence artificielle

TraAM 2021/2022



Le groupe de travail

Académie de Nice

Contexte

Ce document présente, en collaboration avec l'académie d'Aix-Marseille, un projet d'« apprentissage machine » : une machine apprenant à (bien) jouer toute seule.
L'objectif principal est de présenter l'un des mécanismes d'intelligence artificielle appelé **apprentissage par renforcement**. En utilisant comme exemple le jeu **Hexapion** [1], nous illustrons l'apprentissage par renforcement utilisé en intelligence artificielle pour voir émerger un algorithme optimal et nous interroger sur la notion d'« intelligence » dans ce système (particulièrement sur l'intelligence d'un système de boîtes d'allumettes !).

La machine que nous présentons est une machine «physique» en boîtes d'allumettes, manipulable par le public, pour mieux comprendre son fonctionnement. Sa modélisation sur ordinateur est au cœur du projet.

Public

Élèves de première en spécialité `NSI`

Objectifs

Savoirs mathématiques : Système de numération et changement de bases.
Compétences informatiques (en lien avec la spécialité `NSI`) :

- Analyser et modéliser un problème en termes de flux et de traitement d'informations ;
- Décomposer un problème en sous-problèmes, reconnaître des situations déjà analysées et réutiliser des solutions ;
- Concevoir des solutions algorithmiques ;
- Traduire un algorithme dans un langage de programmation, en spécifier les interfaces et les interactions, comprendre et réutiliser des codes sources existants, développer des processus de mise au point et de validation de programmes ;
- Mobiliser les concepts et les technologies utiles pour assurer les fonctions d'acquisition, de mémorisation, de traitement et de diffusion des informations.

1. CC-by- Bastien Masse -2019 à partir du jeu hexapawn de Martin Gardner (1962) et d'une mise en activité par « https ://www.wikidebrouillard.org/ »)

En particulier :

- On mène une réflexion sur le choix de structures de données adaptées ;
- On travaille sur les algorithmes de changement de base ;
- On utilise une bibliothèque externe en vue de la réalisation d'une `IHM`.

Extraits du programme de référence

BO N°1 du 22 janvier 2019 détaillant le programme de `NSI` (première)

Prérequis

Compétences mathématiques : dénombrement à l'aide d'un arbre.
Compétences informatiques :

- Notions d'algorithme et de programmation.
- Les ingrédients des programmes : notion de variable informatique, séquence d'instructions, boucles, instructions conditionnelles.
- Représentation des données : **types construits**

1 Présentation de l'activité à partir du jeu Hexap♙on

La règle du jeu : [2]

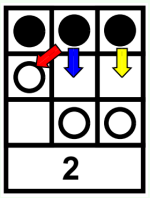
- Un pion ne peut avancer que d'une case en ligne droite ;
- Un pion peut en manger un autre en diagonale (dans ce cas-là, il prend la place du pion mangé qui est retiré du jeu).

Il y a **trois manières de gagner** à l'Hexapion :

- Amener un de ses pions sur la ligne opposée dans le camp adverse ;
- Prendre tous les pions de l'adversaire ;
- Bloquer l'adversaire (qui ne peut plus prendre ou avancer à son tour).

Pour préparer votre jeu, il vous faut :

- 24 boites d'allumettes
- 24 visuels (imprimables ici)
- 10 perles vertes ; 14 perles rouges ; 13 perles bleues ; 18 perles jaunes
- 3 pions blancs, 3 pions noirs



2. https://etincelle.blog/2021/03/18/un-jeu-a-construire-soi-meme-pour-tout-comprendre-sur-lintelligence-artificielle/

Préparation : Sur chacune des boites d'allumettes, vous collez l'une des configurations pouvant se présenter dans le jeu. **Une situation = 1 boite d'allumette**.

Les boites d'allumettes représentent donc les « coups » que « l'ordinateur » va jouer. Dans chaque boite, on dispose des billes de différentes couleurs (1 couleur pour chaque case de la grille qu'on peut jouer). On met dans la boite autant de billes que de « coups » possibles lorsqu'on se trouve dans la situation dessinée sur la boite d'allumette.

Ensuite, place au jeu. ☞ **Durée de l'activité : 15-30 min**

Phase 1 : Manipuler

Déroulement d'une partie : L'animateur joue le rôle de l'IA [3]. Il a les pions noirs. Le joueur humain a les pions blancs et commence la partie car il faut de la "matière" pour que l'IA puisse fonctionner. L'animateur joue pour l'IA en utilisant les boites d'allumettes qui représentent des états du jeu. (Les positions équivalentes du point de vue des actions de l'IA ne sont pas représentées, par exemple : seule la boite avec le pion de gauche avancé est représentée, c'est l'équivalent de la situation avec le pion droit avancé).

Tour 1 : Le joueur humain avance un pion.

Tour 2 : L'IA joue avec une des boites marquées ②.

L'animateur prend la boite d'allumette correspondant à la situation de jeu et tire une perle **au hasard**. Il joue le coup de la couleur indiquée et garde la perle (il la pose sur la boite par exemple).

Tour 3 : Le joueur humain gagne ou bien la partie continue :

> Si le joueur humain gagne, l'IA estime que son coup était mauvais, on retire la perle du jeu, **c'est ainsi que l'IA apprend !** Si la partie continue, l'IA estime que son coup n'était ni bon ni mauvais, on remet la perle dans la boite.

Tour 4 : L'IA joue avec une des boites marquées ④.

L'animateur tire une perle au hasard et joue pour l'IA ; deux résultats possibles :

> L'IA gagne, dans ce cas, l'IA estime que son coup était bon, on remet la perle dans la boite. Le jeu peut continuer. l'IA estime que son coup n'était ni bon ni mauvais, on garde la perle (il la pose sur la boîte par exemple).

Tour 5 : Ce tour possède la même mécanique de jeu que le tour 3.

> Si l'IA perd, la perle du tour 4 est retirée du jeu **sauf si c'est la dernière perle**. Les autres perles sont remises dans leurs boîtes.

Etc...

3. https://files.inria.fr/mecsci/classcodeIAI/pdf/ressources_complementaires/m2_IAI_fichepedagosup-Hexapion.pdf

2 Analyser et modéliser

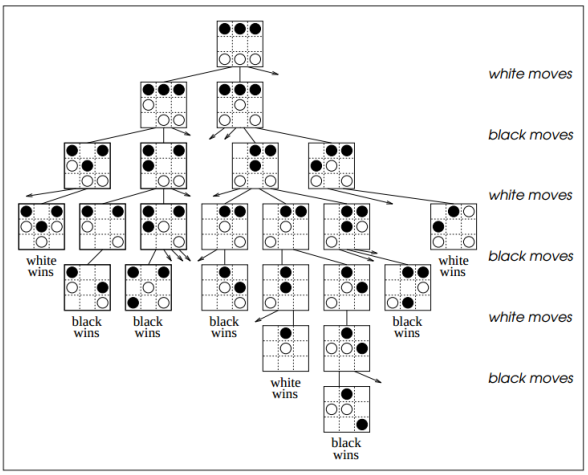
Phase 2 : Verbaliser - Où est l'intelligence dans tout cela ?

En fait, à chaque fois que « l'ordinateur » perd, on retire la dernière perle qu'il a jouée. Comme cela, il « apprend » de ses erreurs. Puisque les perles correspondantes aux mauvais coups sont retirées, « l'ordinateur » ne conserve que les coups qui le font gagner. L'ordinateur est très fort au bout de quelques parties. Incroyable, non ?

Explications :

L'Hexapion est un jeu dit « résolu » dont le résultat (gagner, perdre ou match nul) peut être correctement prédit à partir de n'importe quelle position, en supposant que les deux joueurs jouent à la perfection.

Les boîtes d'allumettes représentent l'arbre des possibles du jeu dans son intégralité.



Chaque fois que l'IA perd, c'est une des branches qui mène à la défaite de l'IA qui est coupée donc il y a de moins en moins de perles dans les boîtes, c'est-à-dire de moins en moins de possibilités de perdre pour l'IA. S'il ne reste plus qu'une perle par boîte, il est impossible de gagner contre l'IA. Dans ce système d'IA, c'est l'humain qui crée la donnée nécessaire au système pour apprendre : c'est un **apprentissage supervisé** puisque c'est nous qui fixons les règles de ce qu'est un « bon coup » ou un « mauvais coup ». Notre IA possède également deux atouts de l'informatique :

- Une mémoire parfaite
- La capacité de répétition

Ainsi la machine ne fait jamais deux fois la même erreur.

Analyser et modéliser à l'aide (au choix) :

- D'un Schéma fonctionnel
- D'un Organigramme
- D'une Narration de programme

3 Programmer un jeu hexapion

Phase 3 : Abstraire

Au niveau première NSI, l'interface graphique (GUI - bibliothèque `tkinter`) : `Hexapion.py` est fournie et **le mode « 2 joueurs » est fonctionnel**. Voir **entête du fichier `Hexapion.py`**

script `Python` - Entête du fichier `Hexapion.py`

```
#######################################
#
# Jeu Hexapion
# ---------------
#
# Auteur: Franck Lagrave
# Mars 2022
# Lycée Beaussier - TraAM Nice -> IA
#
#######################################

import tkinter as tk
from PIL import Image, ImageTk
from tkinter.messagebox import askquestion, showwarning, showinfo # IHM
from random import choice
from apprentissage import index_plateau, arbre_boite

EMPTY = 0
PAWN_W = 1
PAWN_B = 2

NBL, NBC = 4, 4
SIDE = 100
COLORS = ['Cornsilk', 'Sienna']
```

Il s'agit de proposer un développement de fonctions en `Python` (voir fichier `apprentissage.py`) et donc de rendre **le mode « 1 joueur » fonctionnel** permettant de jouer contre la machine et de l'entraîner en soumettant chacune des fonctions programmées par l'élève à un jeu de tests.

> **Remarque** :
> Le paramètre `nbc` de certaines fonctions - `nbc` *désignant le nombre de colonnes* - anticipe un éventuel prolongement du jeu avec une configuration de plateau $3 \times$ `nbc`

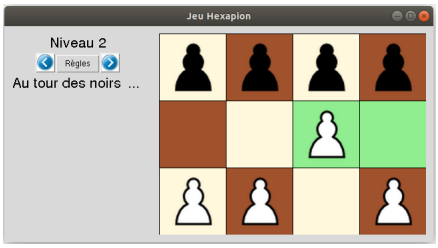


Fig. niveau de configuration avec `nbc = 4`

script Python - Extrait du fichier apprentissage.py

```python
#
# Fichier à compléter: apprentissage.py
#

EMPTY = 0  # case vide
PAWN_W = 1  # pion blanc
PAWN_B = 2  # pion noir

# les différents codes élèves seront validés par les jeux de tests
# (non exhaustifs...) fournis pour chaque fonction.


def conv_ternaire(n):
    """int -> str
    Bases de numération : écriture base 10 vers écriture base 3.

    hypothese : n est un entier naturel donné par son écriture décimale.

    renvoie: l'écriture en base 3 de l'entier naturel n
    sous forme d'une chaîne de caractères.
    """
    pass

# Jeu de tests à décommenter
# assert conv_ternaire(7397) == '101010222'
# assert conv_ternaire(8801) == '110001222'


def index_plateau(ch, nbc):
    """str*int -> str
    Calcul de l'index du plateau identifiant une boîte d'allumettes.

    nbc est le nb de colonnes du plateau.
    ch représente la situation du plateau par un parcours des lignes
    de gauche à droite et de haut en bas.

    hypothese: ch est une chaine de caractères de longueur 9 composée
    uniquement de "0","1","2"le nb de "1" le nb de "2" étant au max nbc.

    renvoie: l'index du plateau sous forme d'une chaîne de caractères.
    """
    pass

# Jeu de tests à décommenter
# assert index_plateau("222100011",3) == "8801"
# assert index_plateau("222010101",3) == "7397"
```

4 Bilan :

Les élèves vont programmer un plateau de jeu Hexapion et réfléchir à une modélisation des boîtes d'allumettes (notre machine «physique») à l'aide des structures de données de types construits (`p`-uplets, `p`-uplets nommés, tableau indexé, tableau donné en compréhension, dictionnaires par clés et valeurs) enseignées au niveau 1re `NSI`.

Pour **éviter les variables globales** dans le script `Hexapion.py` on encourage l'utilisation de **données mutables**. En effet, une technique puissante pour créer des programmes modulaires est d'incorporer des données qui peuvent changer d'état au fil du temps. De cette façon, un seul objet de données peut représenter quelque chose (une boîte d'allumette,...) qui évolue indépendamment du reste du programme.

Dans ce projet, l'élève développe un **apprentissage supervisé** en entraînant une machine ; chaque entraînement affinant la stratégie gagnante de la machine.

Évaluation :

- Présentation orale de l'IA et de ses contributions à la société actuelle.
- Présentation orale des solutions algorithmiques adoptées.
- Démonstration de la mise en œuvre des programmes produits.

Prolongements prévus

- Utilisation de l'archive `Hexapion_V2.zip`[4] :
  la possibilité de **lire et écrire dans un fichier texte** (ex : `boites2.txt`) en `Python` permet de sauvegarder et de charger l'apprentissage de la machine et évite de repasser systématiquement par une phase d'entraînement pour la machine.

  C'est l'occasion de visualiser l'apprentissage en demandant aux élèves de créer les dictionnaires `Python` représentant les boîtes d'allumettes avec leurs billes respectives à chaque tour de jeu.

- Débat sur l'IA - Oral de maturité

Bilan à posteriori et documents

Retour réflexif avec captations d'élèves.
Fiche élève – Fiche prof – liens vers des ressources en relation – Synthèse à posteriori.

4. `Hexapion_V2.zip` est volontairement non disponible pour les élèves lors de cette 1re activité