Documentation Jeu de Nim avec IA

Ce projet simule des parties du Jeu de Nim où deux Intelligences Artificielles s'affrontent pour apprendre les stratégies gagnante.

1. Présentation Fonctionnelle

Règles du jeu de base

- Le jeu commence avec 8 bâtons.
- À chaque tour, un joueur peut retirer 1 ou 2 bâtons.
- Variante "Misère" : Le joueur qui retire le dernier bâton perd la partie.

Concept de l'apprentissage

- L'IA n'est pas programmée avec une stratégie préétablie. Elle utilise une méthode inspirée du système MENACE (Matchbox Educable Noughts and Crosses Engine) :
- Exploration : L'IA choisit ses coups de manière probabiliste. Plus un "poids" est élevé pour une action, plus elle a de chances d'être choisie.
- Renforcement :
    - En cas de victoire, les actions effectuées durant la partie voient leur poids augmenter (Récompense).
    - En cas de défaite, les poids des actions effectuées sont diminués (Punition).
- Convergence : Au fil des simulations, l'IA finit par privilégier les coups qui mènent statistiquement à la victoire.

2. Documentation Technique

Architecture du code

Le projet est structuré en deux fichiers principaux :

- IA.py : Définit la classe Ia et la logique d'apprentissage.
- jeux_NIM_IA.py : Gère le déroulement des parties et l'interaction avec l'utilisateur.

Analyse de la classe Ia (Fichier IA.py)

Cette classe gère l'état interne et l'évolution d'un joueur virtuel.

- **Attributs :**
  - self.choix : Une liste de 8 listes (une pour chaque état de 1 à 8 bâtons). Chaque sous-liste contient deux entiers [poids_1, poids_2] initialisés à 50, représentant les probabilités de tirer 1 ou 2 bâtons.
  - self.historique : Dictionnaire enregistrant les coups joués durant la partie en cours.
  - self.palmares : Liste stockant le nombre de victoires et de défaites [V, D].
- **Méthodes clés :**
  - choisir(batons, choix) : Calcule la probabilité de l'action via un tirage aléatoire pondéré selon les poids actuels de l'état correspondant.
  - recompense() : Incrémente les poids des décisions prises dans historique et met à jour le palmarès.
  - punition() : Décrémente les poids des décisions prises (minimum 1) et met à jour le palmarès.

Analyse du moteur de jeu (Fichier jeux_NIM_IA.py)

Ce fichier contient la boucle principale de simulation.

- **definir_j1(liste_ia)** : Assure l'équité de l'entraînement en choisissant aléatoirement lequel des deux joueurs commence.
- **main()** :
  - Initialise deux instances d'IA (ia1 et ia2).
  - Demande à l'utilisateur le nombre de simulations à effectuer.
  - Exécute la boucle de jeu : retire les bâtons, change de tour, et appelle fin_jeu() pour mettre à jour les poids dès qu'un joueur retire le dernier bâton.
  - Affiche les poids finaux, permettant d'observer la stratégie apprise (ex: privilégier de laisser 1, 4 ou 7 bâtons à l'adversaire).