

Projet d'Intelligence Artificielle

MP : XMS1IE042

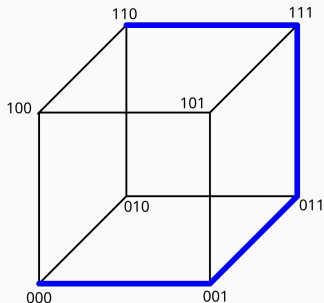
Alexandre Bruckert
alexandre.bruckert@univ-nantes.fr

21 novembre 2025

Nantes Université

Description du jeu

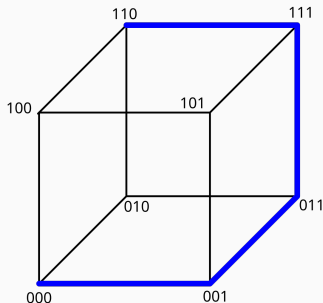
Le problème *snake in the box* est un problème classique (et important!) en théorie des graphes et théorie des codes. Il consiste à trouver la plus longue **chaîne induite** (ou **serpent**) possible dans un graphe représentant un hypercube.



Description du jeu

Une **chaîne induite** est un chemin élémentaire dans un graphe non-orienté, tel que :

- le premier noeud et le dernier noeud de la chaîne n'admettent qu'un voisin appartenant à la chaîne
- tous les autres noeuds de la chaîne n'admettent que deux voisins appartenant à la chaîne.



Description du jeu

On crée un jeu à deux joueurs à partir de ce problème.

Deux joueurs construisent chacun un serpent dans le même graphe G , à partir de deux sommets initiaux distincts, choisis aléatoirement.

À chaque tour, le joueur actif ajoute un sommet et l'arête correspondante **à la tête de son propre serpent**, à condition que :

- le sommet ajouté n'appartienne pas déjà à un serpent;
- après l'ajout, pour tout sommet v qui appartient à un serpent, le nombre de ses voisins qui appartiennent à un serpent (lui-même ou celui de l'adversaire) reste inférieur à 2.

Description du jeu

Le premier joueur qui ne peut pas jouer de coup légal perd.

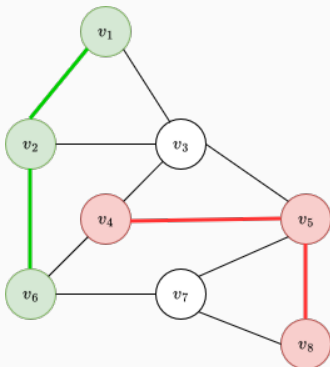


Figure 1 : Un exemple de position finale du jeu. Le serpent du joueur 1 (v_1, v_2, v_6) ne peut plus être étendu, car v_4 est déjà sur le serpent du joueur 2, et v_7 a trois voisins appartenant à des serpents.

Objectifs du projet

Dans ce projet, vous devrez proposer et implémenter une (ou plusieurs!) stratégie pour ce jeu. Pour ce faire, vous trouverez sur Madoc et Gitlab le code permettant de modéliser le jeu (classes implémentant le jeu, UI, etc).

Note importante : Sauf autorisation préalable, vous ne devrez utiliser aucune librairie autre que celles déjà importées dans le code fourni !

Description du projet

Ce projet sera divisé en 3 parties.

- Partie 1 : Commencer par le commencement.
 1. Familiarisez vous avec le code fourni dans les différents fichiers. Assurez vous de bien comprendre le fonctionnement de chaque classe, ainsi que le fonctionnement général.
 2. Implémentez une stratégie jouant de manière aléatoire.
 3. Concevez et implémentez une procédure de comparaison permettant d'évaluer la supériorité d'une approche sur une autre. Pour ce faire, vous devriez faire jouer plusieurs fois les stratégies, en inversant le joueur commençant à chaque fois. Vous implémenterez cette procédure dans un nouveau fichier *evaluate.py*.

- Partie 2 : Ajouter de l'intelligence.
 1. Implémentez une stratégie "gloutonne", basée sur un ensemble de règles heuristiques.
 2. Implémentez une stratégie MinMax permettant d'améliorer les performances du joueur aléatoire et du joueur glouton. Vous ferez particulièrement attention à la profondeur autorisée dans la recherche!
 3. Modifiez cette stratégie en utilisant un élagage $\alpha - \beta$

- Partie 3 : Améliorer les performances.
 1. Proposez et ajoutez des heuristiques permettant d'améliorer les performances. Vous ferez attention de bien justifier et décrire vos choix.
 2. Implémentez une stratégie de recherche de Monte Carlo. Décrivez cette approche, ainsi que les différences avec les approches précédentes en termes de performances.
 3. (Bonus) Si il vous reste du temps, vous pouvez considérer des modèles d'apprentissage pour la création d'une fonction d'évaluation (Q-learning, par exemple).

Consignes

- Pour ce projet, vous travaillerez en **binômes**.
- Vous commencerez par **cloner** (ou mieux : **forker**) le dépôt sur lequel vous trouverez le code :

```
git clone https://gitlab.univ-nantes.fr/bruckert-a/FST_AI_Project_2025
```

- A la fin de chaque séance, vous devrez **commit et push** les avancées que vous aurez faites.
- Vous n'avez (normalement) que les fichiers ***strategy.py*** et ***utils.py*** à modifier. Si vous souhaitez des modifications au reste du projet (bug report, fonctionnalités supplémentaires), ouvrez une issue sur gitlab.

- Vous soumettrez sur Madoc un document pdf créé avec \LaTeX , décrivant et expliquant en détails votre travail.
- Ce document devra contenir un lien vers un dépôt Gitlab public, où vous aurez déposé l'intégralité de votre code.
- Vous devrez également, dans ce document, répondre aux questions préliminaires à la fin de ces slides.
- Ce projet est à rendre pour le vendredi 9 janvier 2026, mais une stratégie fonctionnelle sera attendue pour la dernière séance, semaine 51.

Le projet sera évalué selon plusieurs critères :

- La qualité et la complétude du rapport ($\sim 35\%$)
- La qualité de la solution proposée et de son implémentation ($\sim 35\%$)
- Les performances de la solution proposée lors de la dernière séance et sur un benchmark ($\sim 15\%$)
- La présentation de votre solution lors de la dernière séance. ($\sim 15\%$)

Il est obligatoire d'être présent à au moins 4 séances sur 6, en plus de la dernière séance. 5 points par séance supplémentaire manquée seront retirés de la note finale.

- Vous pouvez vous aider de LLMs pour développer votre code à la condition que vous puissiez comprendre et justifier chacun des choix techniques.
- Vous pouvez également vous aider de LLMs à des fins de relecture et correction pour votre rapport.
- Dans tous les cas, vous devez indiquer votre utilisation de LLM dans chaque document et fichier en ayant bénéficié.
- Vous êtes responsables de ce que vous rendez!

Questions préliminaires

1. Une partie peut-elle se terminer par un match nul ? Pourquoi ?
2. Caractérisez le jeu en termes de (i) son information disponible ; (ii) son caractère aléatoire
3. Exprimez une borne supérieure sur la complexité de l'espace des états du jeu (c-à-d le nombre de positions possibles pour le jeu).
4. Le **facteur de branchement moyen** d'un jeu est le nombre moyen de branches filles pour chaque noeud parent de l'arbre d'état du jeu. Donnez une estimation (même grossière) du facteur de branchement moyen pour ce jeu.
5. Que peut-on en conclure ?