TD noté : Apprentissage d'heuristique

Durée : 4h30

Prof. S. Cussat-Blanc, F. Garrido-Lucero, B. Gaudou, U. Grandi
Prof. TD C. Contet, M. Garouani, J. Mailly, N. Verstaevel

Dans ce TP, vous allez associer les techniques d'apprentissage aux techniques de recherche heuristique afin d'obtenir une version simplifiée des algorithmes ayant permis à AlphaGo de battre le champion du monde de Go en 2017. Vous aurez besoin de tout le matériel étudié en TD/TP.

Pré-consignes

- Les téléphones sont interdits.
- L'accès à internet est autorisé. Vous pouvez consulter des documentations ou des forums si besoin. Cependant, **l'utilisation de LLMs comme ChatGPT est interdite**. Bien sûr, toute communication avec des personnes, à l'intérieur ou à l'extérieur de la salle, est également interdite.
- Vous devez travailler sur Google Colab en **désactivant l'assistant intelligent Gemini**. Pour cela, allez dans Outils $\rightarrow$ Paramètres $\rightarrow$ Assistance IA, décochez les deux premières cases et cochez la dernière. Attention toute activation de cette outil à un moment donné de l'épreuve entrainera immédiatement la rédaction d'un PV de fraude.
- À la fin du TD, vous devez rendre votre travail via ce lien ou directement sur Moodle.
- Le notebook à rendre doit inclure toutes les explications et analyses nécessaires. Il doit être excutable de bout en bout à partir après un redémarage des instances. En particulier, pensez à utiliser des blocs de texte et à bien commenter votre code.

Prise en main

Nous allons travailler sur le jeu du taquin 3×3, qui consiste à ordonner les tuiles du plus petit chiffre au plus grand, comme illustré sur la figure ci-dessous.

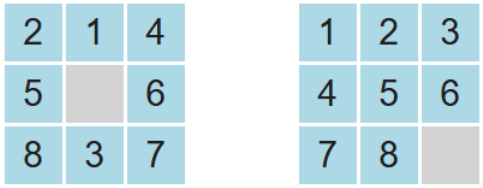

FIGURE 1 – À gauche, l'état initial ; à droite, l'état but.

Pour le résoudre, nous utiliserons l'algorithme A^* avec une **heuristique** que nous calculerons en exploitant les techniques de machine learning étudiées en cours. Rappelez-vous qu'une heuristique est une fonction h qui associe à chaque état E un coût $h(E) \in \mathbb{R}$.

Question 1. Importez le fichier Excel `taquin_1000_states` sur Colab.

La base de données contient 1000 taquins, représentés par leur état initial (col0, col1, ..., col8), quatre heuristiques différentes ($h1, h2, h3, h4$) et le nombre minimal de mouvements nécessaires pour les résoudre (`target`). `Target` correspond au véritable coût d'un état. L'algorithme A^* parfait serait celui qui utilise ces vrais coûts comme heuristique. Cependant, ces derniers ne peuvent être connus qu'après avoir résolu le taquin. Vous allez donc devoir les estimer à partir des observations contenues dans la base de données.

Les quatre heuristiques sont :

- $h1$: Nombre de tuiles mal placées.
- $h2$: Distance de Manhattan.
- $h3$: Distance de Manhattan avec conflits linéaires. Cette amélioration de la distance de Manhattan prend en compte les conflits linéaires. Un conflit linéaire se produit lorsque deux conditions se produisent :
  - — Deux tuiles sont dans la même ligne ou colonne que leur destination finale.
  - — Elles sont dans le mauvais ordre : la tuile de plus petite valeur est à droite ou en dessous de celle ayant une plus grande valeur.

  Par exemple, dans la figure 1, l'état initial présente deux conflits linéaires : entre 2 et 1 (ils sont dans la bonne ligne mais 2 est devant 1) et entre 8 et 7. Étant donné un état E, $h3$ est défini par :

$$h3(E) = h2(E) + 2 \times \text{\# Conflits linéaires}$$

- $h4$: Nombre de tuiles mal placées pondérée. Elle compte le nombre de tuiles mal placées par rapport à l'état final, tout en tenant compte du fait que certaines tuiles sont plus difficiles à déplacer que d'autres. On considère :
  - — La tuile centrale (5) a un poids de 0.8.
  - — Les tuiles des coins (1, 3, 7) ont un poids de 1.2.
  - — Les tuiles des bords (2, 4, 6, 8) ont un poids de 1.

  Pour l'état initial de la figure 1, nous obtenons :

$$h4(E) = 1 + 1.2 + 1 + 0.8 + 1 + 1.2 + 1.2 = 7.4$$

Question 2. Écrivez des fonctions qui prennent un taquin comme entrée (un état) et calculent $h3$ et $h4$. Attention, la tuile vide ne doit pas être considérée dans les calcules.

Estimation d'une heuristique par Machine learning

Dans cette partie, l'objectif est de créer une heuristique en utilisant les algorithmes de machine learning vus en cours. Le dataset d'entrainement est le fichier Excel `taquin_1000_states` fourni. A partir de ce dataset, vous devrez construire le meilleur modèle possible estimant la `target` en

considérant soit l'état initial du taquin uniquement, soit les heuristisques $h1, h2, h3, h4$ uniquement ($h3, h4$ si vous avez réussi leur implémentation), soit une combinaison des états initiaux et des heuristiques.

Question 3. Importer et analyser le dataset fourni. Créer trois datasets contenant soit les états initiaux, soit les heuristiques, soit les deux.

Question 4. Optimiser les principaux algorithmes vus en cours afin qu'ils soient le plus adaptés possible aux datasets proposés.

Question 5. Présenter les résultats des différents algorithmes et choisir en justifiant le meilleur algorithme, les meilleurs hyper-paramètres et les meilleures entrées pour prédire le nombre de coût avant la résolution.

Performance de l'algorithme A^* avec votre heuristique

Dans cette dernière partie, nous comparerons votre heuristique à BFS, DFS et aux quatre heuristiques de la base de données.

Question 6. Générez entre 10 et 100 taquins (en fonction du temps que vous avez à disposition) de manière aléatoire. Assurez-vous que ces taquins soient tous solubles.

Question 7. Résolvez les taquins avec BFS, DFS, et A^* en utilisant au moins cinq heuristiques parmi :

- $h1$ (nombre de tuiles mal placées)
- $h2$ (distance de Manhattan)
- $h3$ (distance de Manhattan avec conflits linéaires)
- $h4$ (nombre de tuiles mal placées pondérée)
- Les trois meilleures heuristiques obtenues avec les méthodes de machine learning.

Indiquez clairement les heuristiques sélectionnées et justifiez votre choix.

Question 8. Créez trois tableaux sous format *DataFrame* où :

1. Le premier tableau contient le numéro de l'état initial et le nombre de nœuds explorés par chaque algorithme.
2. Le deuxième tableau contient le numéro de l'état initial et le temps d'exécution de chaque algorithme.
3. Le troisième tableau contient le numéro de l'état initial et la longueur du chemin trouvé par chaque algorithme pour atteindre l'état but.

Dans chaque tableau, indiquez clairement si l'algorithme n'a pas trouvé de solution à cause du **budget d'exploration**.

Question 9. Imprimez les trois tableaux en utilisant *tabulate* avec l'option `tablefmt='pretty'`.

Question 10. En proposant différentes statistiques sur ce tableaux, analyser la performance de ces heuristiques.