AI Project : The Connection Game
Étude préliminaire

Souaibou Dine BARRY & Zayd ADOUAN
Polytech Nantes
Nantes, France

1 FORMALISATION EN TERMES DE (E, i, F, O, P, C)

1.1 Ensemble des états E

L'ensemble E est l'ensemble de toutes les configurations possibles d'une grille $N \times M$, où chaque cellule peut être vide ou contenir une tuile. Chaque cellule $M[i][j]$ est représentée par une liste $[L, U, R, D]$, où $L, U, R, D \in \{0, 1\}$, indiquant si les côtés gauche, haut, droit, et bas de la tuile sont ouverts (1) ou fermés (0).

Formellement :

$$E = \left\{ M \in (\{0,1\}^4)^{N \times M} \mid M[i][j] = [L, U, R, D] \right\}$$

avec la contrainte suivante :

$$\forall M[i][j] \neq [0,0,0,0], \exists k \in \{(i{+}1, j), (i{-}1, j), (i, j{+}1), (i, j{-}1)\}, M[k] \neq [0,0,0,0]$$

Cette contrainte impose que toute tuile non vide soit adjacente à au moins une autre tuile non vide.

1.2 État initial i

L'état initial $i = M_0 \in E$ est défini avec les conditions suivantes :

- Si $i = 0$ et $j = 0$, alors $M_0[0][0] = [t, x, y, z]$ où $t, x, y, z \in \{0, 1\}$ et $3 \geq t + x + y + z \geq 2$, avec $x \neq z$, signifiant que la première cellule doit avoir au minimum 2 ouvertures et au maximum 3.
- Si $i = N-1$ et $j = M-1$, alors $M_0[N-1][M-1] = [w, u, a, o]$ où $w, u, a, o \in \{0, 1\}$ et $2 \leq w + u + a + o \leq 3$ avec $u \neq 0$
- Conditions pour les cellules en bordure :

$$\begin{cases} M[i][0], & i \in [1, N-2] \\ M[0][j], & j \in [1, M-2] \\ M[i][M-1], & i \in [1, N-2] \\ M[N-1][j], & j \in [1, M-2] \end{cases} = [t, x, y, z] \text{ avec } t, x, y, z \in \{0,1\} \text{ et } t{+}x{+}y{+}z \leq 3$$

  $M[0][M-1]$ et $M[N-1][0] = [t, x, y, z]$ avec $t, x, y, z \in \{0, 1\}$ et $(t + x + y + z = 2$ et $x \neq z)$ ou $t + x + y + z = 0$

  ($x \neq z$ permet d'éviter d'avoir des tuiles de la forme '-' dans les coins)

1.3 États finaux F

$$F = \{M \in E \mid \forall (i, j), (l_{i,j} = r_{i,j-1} \text{ et } r_{i,j} = l_{i,j+1} \text{ et } u_{i,j} = d_{i-1,j} \text{ et } d_{i,j} = u_{i+1,j})\}$$

Ces conditions assurent :

- Pour les tuiles adjacentes horizontalement :
  - $l_{i,j} = r_{i,j-1}$: Le côté gauche de la tuile $M[i][j]$ doit être égal au côté droit de la tuile située à gauche $(M[i][j-1])$
  - $r_{i,j} = l_{i,j+1}$: Le côté droit de la tuile $M[i][j]$ doit être égal au côté gauche de la tuile située à droite $(M[i][j+1])$
- Pour les tuiles adjacentes verticalement :
  - $u_{i,j} = d_{i-1,j}$: Le côté supérieur de la tuile $M[i][j]$ doit être égal au côté inférieur de la tuile située au-dessus $(M[i-1][j])$
  - $d_{i,j} = u_{i+1,j}$: Le côté inférieur de la tuile $M[i][j]$ doit être égal au côté supérieur de la tuile située en dessous $(M[i+1][j])$

1.4 Opérateurs O

L'ensemble O est limité à une seule opération de rotation horaire pour chaque tuile :

$$O = \{\text{Rotation horaire de } [L, U, R, D] \longrightarrow [D, L, U, R]\}$$

1.5 Fonction de transition P

Notons :

- $M[i][j]$ désigne la tuile à la ligne i et à la colonne j
- Les côtés des tuiles sont représentés par :
  - $u_{i,j}$ pour le côté supérieur (up)
  - $l_{i,j}$ pour le côté gauche (left)
  - $r_{i,j}$ pour le côté droit (right)
  - $d_{i,j}$ pour le côté inférieur (down)

$$P = \left\{ M \in F \mid \forall (i, j), \begin{cases} (l_{i,j} = r_{i,j-1}) \text{ et } (r_{i,j} = l_{i,j+1}) \text{ et} \\ (u_{i,j} = d_{i-1,j}) \text{ et } (d_{i,j} = u_{i+1,j}) \end{cases} \right\}$$

avec les conditions de bordure suivantes :

(1) Première colonne : Aucune tuile de la première colonne ne doit avoir un côté supérieur ouvert.

$$\forall i \in [0, N-1], \quad u_{i,0} = 0$$

(2) Première ligne (sauf la première cellule) : Aucune tuile de la première ligne, à l'exception de la première cellule, ne doit avoir un côté gauche ouvert.

$$\forall j \in [1, M-1], \quad l_{0,j} = 0$$

(3) Dernière colonne (sauf la dernière cellule) : Aucune tuile de la dernière colonne, à l'exception de la dernière cellule, ne doit avoir un côté droit ouvert.

$$\forall i \in [0, N-2], \quad r_{i,M-1} = 0$$

(4) Dernière ligne : Aucune tuile de la dernière ligne ne doit avoir un côté inférieur ouvert.

$$\forall j \in [0, M-1], \quad d_{N-1,j} = 0$$

Cette fonction assure que :

- Pour que deux cellules soient connectées de manière sûre, les côtés adjacents doivent être soit tous les deux ouverts (valeur 1), soit tous les deux fermés (valeur 0)

— Les conditions de bordure sont respectées pour garantir qu'il n'y a pas de fuites sur les bords du plateau, sauf aux points d'entrée et de sortie spécifiés

1.6 Fonction de coût C

$$C = \sum \text{nombre de rotations horaires} \times 1$$

Le coût est égal au nombre total de rotations horaires effectuées, avec un coût unitaire de 1 pour chaque rotation.

2 STRUCTURE DE L'ARBRE DE RECHERCHE

Pour une grille $N \times M$, analysons la structure de l'arbre de recherche selon les trois critères demandés :

2.1 Taille de l'arbre

La taille de l'arbre est $4^{N \times M}$, car chaque cellule de la grille peut avoir quatre rotations possibles. Par exemple, pour une grille 3×3, nous aurions 4^9 états possibles.

2.2 Symétrie

L'arbre de recherche est symétrique. Cette symétrie se manifeste par le fait que chaque action (rotation) peut être annulée en effectuant trois rotations ce qui représente une rotation inverse. En pratique, cela signifie que :

— Si nous pouvons aller de l'état A à l'état B, nous pouvons toujours revenir de B à A
— Cette propriété permet de revenir en arrière si une branche de recherche ne mène pas à une solution

2.3 Facteur de branchement

Le facteur de branchement est de $4^{N \times M}$, car à chaque état, nous pouvons effectuer une rotation horaire sur n'importe quelle tuile de la grille. Ce facteur élevé indique que l'espace de recherche grandit rapidement avec la taille de la grille.

3 STRATÉGIE DE GÉNÉRATION DE PUZZLES

Pour générer un puzzle, une approche consiste à :

(1) Partir d'une configuration sûre où toutes les connexions sont satisfaites.
(2) Appliquer des rotations aléatoires dans le sens horaire sur certaines tuiles pour casser les connexions sûres.
(3) Veiller à ce que les modifications créent un puzzle jouable mais difficile.
(4) Pour structurer la difficulté, il peut être utile de concentrer les rotations dans des zones spécifiques de la grille.

4 MÉTHODE D'EXPLORATION DE L'ARBRE DE RECHERCHE

Compte tenu de nos analyses précédentes, notamment la taille importante de l'arbre ($4^{N \times M}$) et son caractère symétrique, nous proposons d'utiliser l'algorithme A* pour explorer l'arbre de recherche.

4.1 Justification du choix

— **Complétude :** A* garantit de trouver une solution si elle existe, ce qui est essentiel pour déterminer si un puzzle est valide
— **Optimalité :** L'algorithme trouve la solution avec le minimum de rotations, ce qui est pertinent puisque notre fonction de coût est basée sur le nombre de rotations
— **Efficacité :** A* est plus efficace que :
  — Le BFS (Breadth-First Search) qui utilise beaucoup de mémoire car il doit stocker toutes les configurations possibles à chaque niveau de l'arbre
  — Le DFS (Depth-First Search) qui ne garantit pas de trouver la solution la plus courte, mais renvoie la première solution qu'il rencontre
— **Gestion de la symétrie :** L'algorithme permet d'éviter la redondance en gardant une trace des états déjà visités

4.2 Éléments d'implémentation

— **État :** Représentation de la grille à un instant donné
— **Fonction de coût g :** Nombre de rotations effectuées
— **Fonction heuristique h :** Estimation du nombre minimal de rotations restantes
— **Liste fermée :** Pour éviter de revisiter les mêmes états

5 ÉLABORATION D'HEURISTIQUES

Pour améliorer l'efficacité de la recherche, nous proposons plusieurs types d'heuristiques :

5.1 Contraintes de connexité

— **Détection précoce d'incohérences :** Vérifier immédiatement si une rotation crée des connexions impossibles avec les tuiles voisines
— **Vérification des bordures :** S'assurer que les tuiles en bordure respectent les contraintes (côtés fermés sauf entrée/sortie)

5.2 Fonctions d'évaluation d'états

— **Nombre de connexions sûres :** Compter le nombre de paires de tuiles adjacentes formant des connexions valides
— **Distance à l'objectif :** Estimer le nombre minimal de rotations nécessaires pour atteindre un état final
— **Connectivité :** Évaluer si un chemin existe entre l'entrée et la sortie

5.3 Ordonnancement des actions

— **Priorité aux tuiles critiques :** Commencer par les tuiles proches de l'entrée et de la sortie
— **Rotations progressives :** Tester d'abord une seule rotation avant d'essayer les autres possibilités
— **Zones connectées :** Traiter en priorité les zones où plusieurs tuiles sont déjà correctement connectées

6 ÉTUDE DE LA VARIANTE

Dans cette variante du jeu, certaines pièces ne sont pas initialement sur le plateau mais sur le côté, à placer correctement. Cette

modification impacte notre modélisation et la complexité comme suit :

6.1 Impact sur la modélisation (E, i, F, O, P, C)

- **États** E **:** L'ensemble des états doit maintenant inclure :
  - La configuration du plateau
  - La liste des pièces disponibles sur le côté
  - Les positions vides où placer ces pièces
- **Opérateurs** O **:** Deux types d'opérations sont maintenant possibles :
  - Rotation des pièces déjà placées
  - Placement d'une nouvelle pièce (avec une orientation)

6.2 Impact sur la complexité

La complexité est significativement augmentée car :

- Pour chaque position vide, nous devons considérer :
  - Le choix de la pièce à placer parmi celles disponibles
  - Les 4 orientations possibles pour cette pièce
- Si k est le nombre de pièces à placer, le facteur de branchement devient :

$$4^{N \times M - k} \times (k! \times 4^k)$$

où :

- $4^{N \times M - k}$ représente les rotations des pièces déjà placées
- $k!$ représente les différentes façons de placer les k pièces
- 4^k représente les orientations possibles pour ces k pièces