Université de Bourgogne

Rapport Algo et complexité

Enseignants :Dominique Michelucci et Alexis Guyot Auteur : DIARRA Ibrahim

13 mai 2022

Table des matières

1 Introduction

Le sujet qui a été choisi est le sujet numéro 1. Il concerne le jeu Eternity II. Ce dernier est un puzzle de 256 pièces(plateau de 16x16) inventé par Christopher Monckton et publier par Tomy en juillet 2007. Une récompense de 2 millions de dollar était promise à la première personne qui présentera une solution au casse-tête dans le délai imparti (2 ans et demi) . Personne (ni humains, ni machine) n'a réussi le challenge. Le jeu fait partie de la catégorie des puzzles appelé "bounded unsigned edge-matching problème" ou "problème de correspondance de bord non signé borné "en français . Plus simplement, Il y a deux règles à respecter pour qu'une configuration soit solution du jeu.

1. Les bords connectés de deux pièces doivent avoir la même valeur (couleur,motif, signe etc...).
2. Les cotés(celles d'une piece) adjacents aux bords du plateau ont tous la même couleur(couleur qui ne doit pas apparaître ailleur que sur les bords).

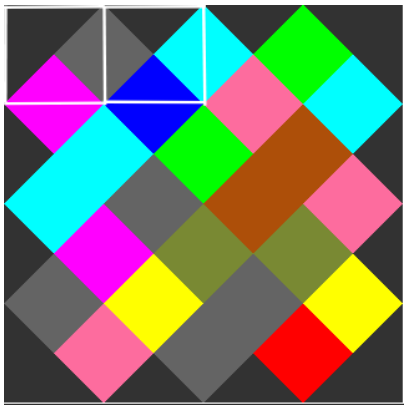

FIGURE 1 – 4x4 bounded unsigned edge-matching problem

Le but du projet est le suivant :

1. Dans un premier temps, génerer un puzzle aléatoire et mélanger les pieces.
2. Dans un second temps, résoudre le puzzle à l'aide d'une méthode de recherche arborescente avec retour en arriere.

Nous avons deux contraintes que nous tenterons de satisfaire . En effet, notre programme doit pouvoir résoudre un puzzle 12x12 avec une dizaine de couleurs.
Pour répondre a cette problematique, nous suivrons les étapes suivantes :

— Tous d'abord nous allons analyser le problème.
— Ensuite nous allons étudier notre implementations et nos résultats.

2 Annalyse du problème

Pour pouvoir trouver une solution à un problème, il faut déja le comprendre.
Ce jeu semble être très simple au premier abort. Les règles sont simple à comprendre et à respecter. Alors on est en droit de se demander pourquoi personne n'a trouver une solution et remporter les 2 million de dollar de récompense. Comme on peut s'y attendre, cela n'est pas aussi simple et on va très vite comprendre pourquoi.

2.1 En quoi la recherche de solution est difficile voir impossible(avec la technologie et les connaisances actuelle)

2.1.1 Dèscription complète du jeu

Selon [4] ,[3] et [1] Comme indiquer plus haut, ce jeu est un puzzle de type problème de correspondance de bord non signé borné de 256 piece(grille de 16x16). Il a été conçu specialement pour être difficile à résoudre par une recherche informatique par force brute. Chaque pièce du puzzle a ses bords d'un côté marqués de différentes combinaisons de forme et couleur.En realité, on peut uniquement utiliser les couleurs et se passer des formes car une combinaison de couleur et de formes peut etre vu comme une couleur. Par exemple la couleur rouge associé la forme étoile sera representé par le rouge et la couleur rouge et la forme carré sera representé par du marron etc...

Chaque pièce peut être utilisée dans 4 orientations. Il y a 22 couleurs, sans compter les bords gris. Cinq des couleurs se trouvent exclusivement dans les 60 paires de bords. Plus précisement dans les losanges de l'anneau le plus à l'extérieur. Les 17 autres couleurs sont utilisées dans les 420 paires de bords "intérieures" restantes. Les couleurs sont utilisées uniformément, chacune des 5 couleurs de bordure étant utilisée dans exactement 12 paires d'arêtes, et chacune des 17 couleurs intérieures utilisées pour 24 paires d'arêtes (5 couleurs) ou 25 paires d'arêtes (12 couleurs). Le nombre total de paires de bords est de 480. L'une des cinq couleurs de bordure ne se trouve sur aucune pièce d'angle, tandis que les 17 couleurs intérieures sont utilisées au moins une fois sur une pièce de bordure.

Il y a 4 pièces d'angle (avec deux côtés noir), 56 pièces de bordure (avec un côté noir) et 196 pièces intérieures (avec quatre côtés colorés). Chaque pièce a un arrangement unique de couleurs, et aucune des pièces n'est symétrique en rotation, de sorte que chacun des 1024 choix de pièces et d'orientation se traduit par un motif différent de couleurs de bord.

Une piece qui doit être placée dans une position et une orientation spécifiées près du centre du plateau etait fourni.

Deux puzzles d'indices étaient disponibles avec le lancement du produit, qui, s'ils sont résolus, donnent chacun une position de pièce sur le puzzle principal de 256 pièces. Clue Puzzle 1 est un puzzle carré de 36 pièces (6×6) et Clue Puzzle 2 est un puzzle rectangulaire de 72 pièces (12×6). Deux puzzles d'indices supplémentaires de mêmes dimensions ont été mis à disposition en 2008 : le 36 pièces Clue Puzzle 3 et le 72 pièces Clue Puzzle 4. Le livre de règles [3] stipule que le puzzle peut être résolu sans utiliser les indices.

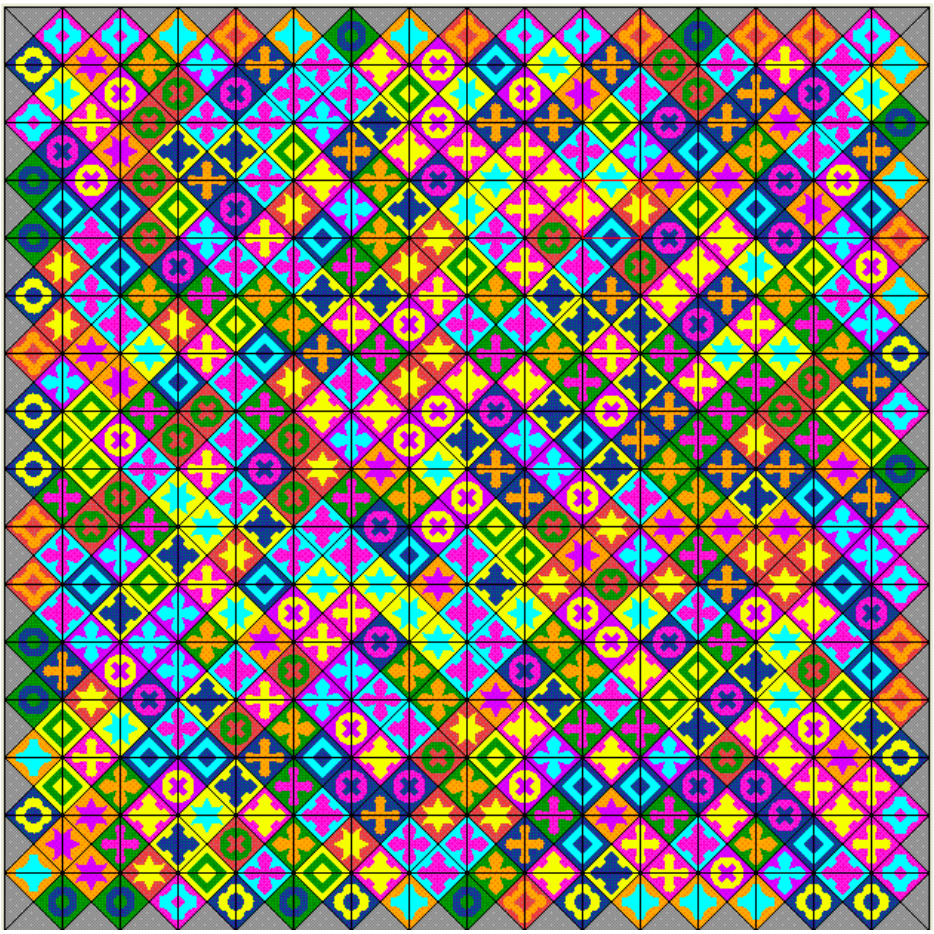

FIGURE 2 – Exemple de plateau eternity2 résolut

2.1.2 Tentatives et Methodes de résolution

Pour resoudre ce problème, plusieur pistes ont été explorer par les chercheurs.

Pierre Schauss et Yves Deville ont également travaille sur le projet en utilisant une aproche par programmation par contraintes et avec un voisinage de très grande taille. [5] Leur meilleur score a été de 463/480. Le score est exprimé en nombres de jonctions interieur completer. Selons leur rapport, deux logiciels de calcul distribués on été développés utilisant une approche brute-force de type backtracking :

— Le premier a été développé par Dave Clark. Environ 1000 ordinateurs calculaient de manière permanente pour ce projet. Ce projet était le plus populaire et à notre connaissance, il est celui qui a soumis la meilleure solution avec un score de 463/480. Ce projet a duré 6 mois. Dave Clark a maintenant abandonné car il ne croyait plus dans l'approche brute-force pour résoudre le puzzle.

— Le deuxième est un projet français qui semble avoir moins de membres. Ils n'ont pas encore publié leur meilleur score."

Plusieurs autres traveaux utilisant du backtraking existe. Mais aucun n'a vraiment obtenue une solution complète au problème.

Pour se rendre compte de la difficulter de l'exercice, calculons la complexité du jeux.

2.1.3 Complexité

La difficulté de ce type de problème reside dans le nombre exponentiel de candidats potentielle et donc le temps necessaire avec les ordinateurs dont on dispose actuelement pour les verifier tous et trouver une qui fonctionne . En effet, le seule moyen pour trouver une solution est d'essayer de disposer les pieces encore et encore j'usqu'a tomber sur une solution. Si on a vraiment beaucoup de chance, on tombe sur une des solutions(ou la solution car certaines configurations en ont une seule) rapidement. Si on n'a pas de chance, on revient en arrière et on change la dernière piece placer par une des nombreuses autre possibilité et on reitère encore et encore. La complexité d'une tel méthode de résolution est monstrueuse. En fonction de la dèscription des pieces du jeux et des règles, nous déduisons les éléments suivant :

- Les quatres piece d'angle peuvent être sur n'importe lequel des 4 emplacements. Ce qui coorespond à toute les permutations de 4 éléments qui nous est donné par 4 !.
- Les pieces de bord sont au nombre de 56 en ne comptant pas les 4 précédents. Ce qui coorespond encore une foie au nombre de permutations de 56 éléments qui est 56 !.
- Enfin les pieces au centre sont au nombre de 196. Ce qui coorespond encore une foie au nombre de permutations de 196 éléments qui est 196 !.
- Avec les rotation, il ya 4^{256} possibilité en plus.

Tous ceci permet de dire que le nombre de configuration possible peut être borner par la valeur $4! * 56! * 196! * 4^{256}$.

4 !=24 ; 56 !=7.109*10^{74}; 196! $= 5.1 * 10^3$65. Ce nombre depasse de loin le nombre d'atome dans l'univert.

Un autre moyen d'estimer cette complexité serait le nombre de noeux parcourut dans le pire des cas dans l'arbre des possibles.

Pour une configuration n*n=N sans prendre en compte les rotations possible de chaque piece(c'est à dire si on explore toute des branches de l'arbre des possibilités en suposant que chaque piece est unique) le nombre de noeud parcouru est de l'ordre de S(N)

$$\begin{aligned} \mathrm{A}_0 &= 1 \\ A_i &= (N-i)A_{i-1} \end{aligned}$$

$$S(N) = \sum_{i=0}^{N} A_i$$

Pour N=3 S(N)=18

Si nous prenons en compte les rotations, cette complexité explose d'avantage.

$$\begin{aligned} \mathrm{A}_0 &= 1 \\ A_i &= 4 * (N-i)A_{i-1} \end{aligned}$$

$$S(N) = \sum_{i=0}^{N} A_i$$

IL faut également noté que contrairement à la taille du plateau, l'augmentation du nombre de couleur diminue les candidats potentiel pour une case du plateau donnée. Au vu des complexités, on comprend mieux pourquoi ce jeu est aussi difficile. Il fait partie de la classe np en théorie des complexités. En effet, il est possible de verifier une solution donnée en temps polynomial mais tres difficile d'en trouver une quand le nombre de piece est grand. Le probleme meme np-complet selon [2].

3 Etude de notre implementation et analyse de nos résultats

La méthode que nous utilisons est l'approche par force brute. Nous explorons les branche de l'arbre des possibilités jusqu'a trouver une configuration exact. Les piece sont positionner tant qu'on le peut .Une foie bloquer, on revient sur le choix de la dernère piece poser. Ensuite on reitère l'operation.

3.1 Mise en place de la méthode

Notre implementation n'est pas le jeu originale mais s'en approche . Le nombre de couleurs et de pieces est variable pour permettre l'etude de differentes configuration. La contrainte que chaque piece est unique n'a pas été respecter. La grille peut aller jusqu'a 12*12 et le nombre de couleur peut aller jusqu'a 20.
Une foie q'une configuration correcte à été generer de manière aleatoire,les pieces sont melangé en les déplaçant et en les tournant de manière aleatoir.
Ensuite un la resolution peut commencer.

3.2 Structures de données

Les principales structure de données utiliser dans le cadre de ce projet sont ceux concernant l'arbre et la piece. Le plateau est tous simplement une tableau à deux dimentions de type piece. Le tableau a été choisie car il permet d'acceder directement à un élement.
Un type piece a été définit pour les pieces du jeu. Il se presente de la façon suivante.

```
type piece ={posx:int;posy:int;top:int; rigth :int; bottom:int;left:int;num:int }
```

il contient la position x et y des la piece, les valeur coorespondantes aux couleurs haut,droit, bas et gauche, et le numero de la piece.
Ce numero est important pour identifier une piece. Par exemple pour sa suppression rapide dans une liste on parcour la liste au plus une seule foi.

Le type arbre a également été definie. Ils se presente de la façons suivante.

```
type arbre={ pieces:piece list; possible:piece list array}
```

Il permet de stocker les information utile pour le backtracking. Il contient une liste de piece dont la tete est la dernière piece placer. Il contient également un tableau de liste de piece, que contiendra pour chaque piece poser la liste des autres possibilité trouver. Cela évitera la recherche d'autre solution à chaque retour en arriere.

3.2.1 Algorithme

Il est important de noté que la liste des autres possibilités pour une piece placer dans l'arbre ne tient pas compte de toute les rotations possibles d'une piece. C'est à dire que si une piece est déclarer possible alors ses autre rotation(qui sont peut être aussi possible) ne sont pas vérifié et la piece est ajouté dans l'orientation qui a le premier marcher. Cela a pour consequence quelque foi de ne trouvé aucune solution a la fin du parcour de l'arbre .

Passons maintenant à notre algorithme de backtracking.
Dans un premier temps les pieces sont triés pour récuperer lespieces d'angle. L'un de ces derniers est choisie et l'arbre est initialiser avec. Ensuite les pieces restantes sont passer à la fonction "Resoudre" qui se charge de placer la piece suivante tant que c'est possible. une foie qu'il a essayé toute les pieces, il appelle la fonction "back" qui éffectue le backtrack.

3.2.2 Mesures

N=n*n	NBCouleur	Temps
4	2	0.002329
4	4	0.003992
5	2	0.017813
5	4	0.228227
6	5	trop lent
6	6	0.060179
6	6	0.275285
8	7	trop lent
8	10	0.939152
8	11	0.017299
8	13	0.048581
8	20	0.012821
10	10	trop lent
10	15	0.764203
10	18	0.239522
12	20	2.461027

Les temps d'executions semble fortement dépendre du nombre de piece . Mais le nombre de couleur joue également un grand role. En effet on constate que plus ce dernier est grand et plus le temps est cour. On remarque également que le temps augmente de façon exponentielle quand le nombre de couleur et la taille du plateau son proche pour certaine valeurs.

3.2.3 Ameliorations possible

Une possibilité serait d'utiliser des indexs afin de ne plus parcourir toute la liste de piece pour touver la prochaine piece compatible.

4 Conclusion

Au terme de ce projet, nous pouvons affirmer qu'il a été des plus instructif. Il nous a permit d'appliquer et d'approfondire nos connaissances sur la complexité, la programmaton fonctionnel, le parcourt en profondeur d'arbre etc. Il nous a également permit de prendre ne main le language caml et de decouvrir les mécanismes de cette dernière.

Le jeux Eternity 2 fut un très bon moyen pour nous de découvrir les frontières de l'informatique d'aujourd'hui.

Références

[1] efreidoc.fr. https://efreidoc.fr/L2%20-%20PL2/SDD/Projets/2007-08%20%3A%20Eternity%20II/Groupe%203/2007-08.projet.eternityII.rapport03.sdd.pdf.

[2] Erik d. demaine et martin l. demaine. https://erikdemaine.org/papers/Jigsaw_GC/paper.pdf.

[3] Uk.description. https://web.archive.org/web/20071006033127/http://uk.eternityii.com/Download.ashx?id=19934.

[4] wikipedia. http:https://en.wikipedia.org/wiki/Eternity_II_puzzle#cite_note-:0-3.

[5] Pierre Schauss and Yves Deville. Hybridation de la programmation par contraintes et d'un voisinage à très grande taille pour Eternity II. In Gilles Trombettoni, editor, *JFPC 2008-Quatrièmes Journées Francophones de Programmation par Contraintes*, pages 115–122, Nantes, France, June 2008. LINA - Université de Nantes - Ecole des Mines de Nantes.