1 Introduction

2 Eternity II

2.1 Le jeu

Eternity II est un jeu sorti en 2008 qui repose sur un principe assez simple, c'est un puzzle de 16 par 16 qu'il faut réassembler. Il est composé de 256 pièces carrés, qui ont, sur chaque arête une couleur donnée. [images tt ca tt ca]. Afin de pouvoir assembler le puzzle, il suffit placer les pièces de façon à ce que les faces adjacentes soient de même couleur. Comme un puzzle classique, il y a des pièces de coin et de bord. Ceux-ci sont reconnaissables car ils possèdent une ou deux arêtes grises. Par contre, la où ca devient complexe, c'est qu'une pièce n'as pas une place prédéterminée (comme dans un puzzle), c'est à dire qu'elle peux se situer n'importe où sur le plateau.

2.2 Le défi

A ce jour, personne n'a réussi à résoudre ce puzzle (même grâce à l'aide de supercalculateurs) malgré les différentes stratégies mise en place. Pourquoi ? Car derrière ce jeu anodin se cache l'un des plus grand problème du monde actuel : les problèmes NP-difficiles. Ceux-ci sont fait de tel sorte que même en connaissant leur structure ou fonctionnement, il est pratiquement impossible d'en déduire un algorithme de résolution. L'une des solutions les plus fiables à ce jour est de tester tout les cas possible (qui est évidemment très important).

Exemple 2.1. Le nombre de combinaisons pour Eternity II s'élève à 10^{545}, c'est à dire environ 10^{450} fois le nombre d'atomes dans l'univers connu (estimé à au plus 10^{80}) ! ! !

2.3 La recette secrète d'Eternity II

Pour rendre ce problème combinatoire, il est nécessaire de respecter plusieurs conditions.

Chaque pièce est unique l'unicité des pièces est importante, car si une pièce est en double, cela veux dire que la pièce peux être placée à deux endroits différents (ce qui simplifie le problème)

Le ratio de nombre de couleurs par nombre de pièces est calculé [joindre graphique tt ca tt ca] : il faut qu'il y ait assez de couleur pour que chaque pièce soit unique, mais pas assez pour que l'on puisse déterminer les pièces adjacentes

Exemple 2.2. Supposons qu'il y ait trop de couleurs. Cela veux dire qu'une pièce à peu de voisins (car chaque pièce est unique, par conséquent les couleurs sont distribués uniformément à travers les pièces), si la pièce à très peu de voisins, je peux déduire des groupement de pièces assez facilement.
Donc je simplifie mon problème.

2.4 Etat de l'art

Nombreux sont ceux qui ont essayé de résoudre le problème... plusieurs moyens ont été mis en place, grâce au solveur SAT (solveur de satisfiabilité booléenne dont le fonctionnement est assez complexe pour ne pas être abordé ici), par une approche graphe ou encore par bruteforce.

Il est estimé que actuellement, l'approche la plus performante est la résolution par bruteforce, car c'est elle qui est la plus rapide.

[à développer]

3 Sujet / problématique

Le but du stage est donc de déterminer si, malgré les dires, on ne pourrait pas trouver une méthode de résolution plus efficace que la bruteforce, qui repose sur une grande quantité d'information pré-calculées, cette méthode est appellée smartforce par la suite.

Ces informations pré-calculées serviront différentes causes, mais deux objectifs principaux peuvent en être explicités :

- les ouvertures
- les finales

Afin de comprendre le principe des ouvertures et des finales, il est important de pouvoir se représenter un arbre de possibilités où chaque branche de l'arbre est une combinaison spécifique. [là faut une bonne image explicite]

3.1 Ouvertures

Le but des ouvertures, est de pré-calculer jusqu'à un certain niveau toutes les combinaisons possibles, et de les stocker. Cela permet par la suite, de pouvoir paralléliser les calculs afin de tester plusieurs sous instances du problème

3.2 Finales

De la même façon, pré-calculer les fins possibles permet de connaitre plus rapidement si la combinaison actuelle est possible ou non.

3.3 Difficultés préliminaires

afin de résoudre le problème, la principale difficulté c'est que le jeu de base est trop complexe et ne permet pas de déterminer si l'approche actuelle est adapté. Par conséquent, on utilisera des instances d'Eternity II, qui sont des plateaux de plus petite taille (4x4, 5x5 ...) qui ont les mêmes propriétés que le jeu de base.

4 Approche

Dans cette partie, nous expliquerons quels sont les différents outils et stratégies mis en place pour permettre la résolution du problème. Dans un premier temps nous avons eu besoin de mettre en place une valeur étalon, qui nous permet de savoir si les differentes méthodes de smartforce sont bonnes ou pas. Cette valeur étalon est un programme de bruteforce qui nous fournit le nombre total de noeuds, le nombre de solution, le nombre de noeuds à la premiere solution et les tous les timers correspondant.

4.1 Bruteforce

Afin de pouvoir partir sur de bonnes bases, plusieurs différents méthodes de parcours (quel chemin prendre pour résoudre mon plateau) ont été utilisé, afin de voir quel parcours est le plus performant pour la résolution brute. En sachant que le nombre de noeuds/sec est la même, l'unité de mesure est le nombre de noeuds.

Les differents types de parcours sont :

rowscan on pose les pieces en lignes horizontales sur le plateau

diagonal on pose les pieces en diagonal

spiral in on dispose les pièces en spirale en partant de l'exterieur [image]

spiral out idem que spiral in mais en partant de l'interieur vers l'exterieur

Ces différents types de parcours ont été testés sur plusieurs instances de taille variable.

4.2 Smartforce

Une fois la valeur étalon fixée, il est maintenant facile de mettre en place un autre approche du problème qui à pour principe de cumuler une grande quantité de donnée pour faire face au nombre exponentiel de possibilités.

Les différents types de données (nommés modèles) sont comme différents points de vues du problème. Ils sont plus ou moins utiles, mais la force réside dans leur union. Mais surtout, ils permettent de mettre en place le concept d'ouvertures et finales.

4.2.1 CaPi

L'approche CaPi (abbréviation de Cases/pieces) est l'approche la plus naive, elle permet de définir quelle piece peut être placée sur telle case et inversement, quelle case peux avoir telle piece.

Cette approche est l'interaction la plus basique de notre problème. C'est aussi celle-ci qui est utilisée en bruteforce.

4.2.2 BoCo

L'approche BoCo (Bordure/Couleur) est bien plus fine : si l'on connait quelle piece est sur telle case, on sait quelle couleur peux se placer sur telle bordure [de la case]. Elle permet d'implémenter un système de mis à jour bien plus performant car ne nombre de couleurs est bien plus petit que le nombre de pièces.

Exemple 4.1. Si une couleur disparait, alors tt les pièces ayant cette couleur ne peuvent plus être placés à cette case, par conséquent, les autres bords de la case ont (probablement) des couleurs qui disparaissent aussi (propagation de la disparition).

4.2.3 Corolles

Grace aux visions CaPi et BoCo, il est possible de pré-calculer des zones du plateau nommés corolles, ceux-ci contiennent tous les cas possibles dans cette zone donnée.

Le nombre de cas possible étant très important, il est nécessaire de le classer. Les corolles peuvent êtres identifiés grâce à plusieurs critères.

- taille du plateau
- l'orientation de la corolle
- La pièce (et sa rotation) à l'origine de la corolle
- La case à l'origine de la corolle
- La taille de la corolle

position et orientation des corolles

4.2.4 BoCoDiag

5 Resultat

6 Manuel d'utilisation

7 Manuel Technique