

Projet Evolution Artificielle

Année 2006-2007

Solveur de problème de pavage par polyominos*Le projet doit impérativement être réalisé par groupes de trois étudiants.***Description générale.**

Le projet consiste à implémenter en OCaml un solveur de problème de pavages par polyominos.

Un polyomino : c'est un ensemble quelconque de cases d'un échiquier $n \times m$.

Un problème de pavage d'un polyomino A par un ensemble B de tuiles : c'est trouver toutes les manières de couvrir sans chevauchement toutes les cases du polyomino A avec des copies translattées de tuiles de B . Dans ce projet, on suppose que l'on dispose d'un nombre illimité de tuiles de chaque sorte et que l'on n'effectue que des translations.

Elément de réalisation.

Le projet est divisé en trois parties indépendantes :

- Un éditeur de problème de pavages.
- Un solveur.
- Un éditeur de solutions.

1) **L'éditeur de problème :**

Il faut écrire un programme qui permette de rentrer facilement les données du problème qui sont :

- a) Le polyomino à paver.
- b) Le jeu de tuiles qui doit permettre le pavage.

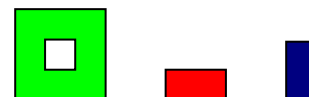
L'interface graphique devra permettre de dessiner facilement un polyomino (à la souris, on pourra sélectionner un case, sélectionner un rectangle de cases, effacer un case, effacer tout) et de charger et d'enregistrer les données dans un fichier sous la forme d'une chaîne de caractères comme suit :

3,3 : 0,0;1,0;2,0;0,1;2,1;0,2;1,2;2,2 : 2 : 0,0;1,0 : 0,0;0,1 :

où :

- les 2 premiers nombres en jaune correspondent aux dimensions de l'échiquier,
- les données en vert aux coordonnées des cases du polyomino à paver énumérées dans l'ordre suivant : $((x,y) < (x',y')$ ssi $y < y'$ ou $y = y'$ et $x < x'$).
- le nombre en rose au nombre de tuiles.
- et les données en rouge et en bleu aux coordonnées des cases des tuiles classé dans le même ordre que ci-dessus (ici il y a deux tuiles).

La description ci-dessus correspond au problème suivant :

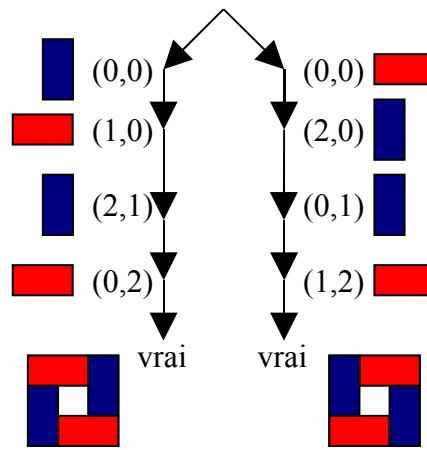


où le polyomino vert doit être pavé par des copies des tuiles rouge et bleu.

2) Le solveur :

Vous devez écrire un programme qui charge un problème et qui en trouve toutes les solutions. Pour trouver les solutions, on va construire un arbre dont chaque nœud interne sauf la racine stocke une position et un type de tuiles et où les feuilles contiennent un booléen indiquant si le chemin de la racine à la feuille correspond bien à un pavage. Plus précisément, on regarde pour chaque tuile de combien de manières différentes on peut recouvrir, avec une copie translaturée, la première case (pour l'ordre ci-dessus) du polyomino (La copie doit tenir entièrement dans le polyomino). S'il y a $k > 0$ possibilités en tout, alors dans l'arbre la racine a k fils et chacun de ses nœuds fils contiennent une position et le type de tuile utilisée, sinon on crée un fils affecté de vrai ou faux suivant que le polyomino à paver est vide ou non. On procède alors récursivement sur chacun des k sous-arbres (le nouveau polyomino à paver étant le polyomino précédant privé de la tuile que l'on vient de poser).

Par exemple, pour le problème ci-dessus, on veut construire l'arbre :



1) L'éditeur de solutions :

Ce programme doit permettre de donner le nombre de solutions du problème à partir de l'arbre ainsi que d'afficher la n -ième solution pour l'ordre préfixe sur l'arbre, d'afficher un diaporama des solutions, d'afficher une page contenant un nombre donné de solutions.

Le projet doit être le plus modulaire possible. En particulier, si vous avez le temps, vous pouvez rajouter des contraintes sur le nombre de tuiles utilisables de chaque sorte ou le nombre de tuiles de chaque sorte présentes dans la solution (il faut alors compléter la chaîne de caractère). Il doit être ergonomique. Par exemple, vous devez fournir un makefile pour la compilation.

Dossier de projet :

Chaque groupe devra rédiger un dossier de projet comprenant :

- une description générale du problème;
- une justification technique sur le choix des structures de données et des algorithmes essentiels;
- un listing commenté du programme;
- des essais d'exécution judicieusement choisis, "montrant" le bon fonctionnement du programme.

Hotline : En cas de problème concernant les objectifs à atteindre, vous pouvez envoyer un mail à olivier.bodini@lip6.fr