Projet Programmation 2
2014/15
Troisième partie

Emile Contal & Stefan Schwoon

1 Des générateurs d'instances pour les jeux

L'objet de la troisième partie du projet est de générer des instances de nos jeux résolubles par un raisonnement fixé. Selon la complexité des règles utilisées, les instances seront plus ou moins difficiles à résoudre par le joueur. Pour la plupart des jeux, programmer un tel générateur nécessite de développer un solveur partiel. Dans ce cas une technique classique consiste à construire séquentiellement l'instance de manière gloutonne :

1. on fixe une cellule aléatoirement,
2. on utilise toutes les règles du niveau de difficulté pour éventuellement déduire le contenu d'autres cellules,
3. si on rencontre une contradiction on recommence à zéro,
4. sinon on continue avec une autre cellule.

Dans d'autres cas comme le démineur, on pourra préférer partir d'une instance aléatoire puis vérifier si elle est résoluble par le solveur, et la rejeter sinon. Vous pouvez trouver plus de détails sur la documentation de la collection des petits jeux ici : `http://www.chiark.greenend.org.uk/~sgtatham/puzzles/devel/writing.html#writing-generation`.
Vous pourrez également trouver des exemples de règles de déduction adaptées à un certain niveau de difficulté en regardant directement le code source de ces jeux, notamment les commentaires, `http://tartarus.org/~simon-git/gitweb/?p=puzzles.git;a=tree`.

Vous devez programmer un générateur pour au moins :
- *Flip* (générateur simple, pas besoin de solveur) ;
- *Démineur* ;
- un [1] des quatre jeux suivants : *Solo*, *Tents*, *Tower*, *Unruly* ;

1. Sauf pour le groupe de 1. Deux pour le groupe de 3.

2 Exemple de règles pour le Démineur

Voici un exemple de règles de déduction que l'on peut utiliser pour le démineur :

Facile.
- Lorsque le nombre de voisins non dévoilés est égal au numéro de la case, tout les voisins sont des mines.
- Lorsque le nombre de mines connues parmi les voisins est égal au nombre de la case, tout les voisins non dévoilés n'ont pas de mines

Normal. Soient c_1, c_2 deux cases, V_1, V_2 leurs voisins respectifs, et m_1, m_2 leur nombre de mines.
- Lorsque $V_1 \cap V_2 \neq \emptyset$ et que $|V_1 \setminus V_2| = m_1 - m_2$, alors $V_1 \setminus V_2$ ne comporte que des mines et $V_2 \setminus V_1$ ne comporte aucune mine.
- Lorsque $V_1 \subset V_2$, le nombre de mines de dans $V_2 \setminus V_1$ est égal à $m_2 - m_1$.

Difficile. Si il existe $c_1, \ldots, c_n$ tels que l'ensemble de leurs voisins non déterminés soient disjoints deux à deux et que la somme $\sum_{i=1}^{n} m'_i$ de leur nombre de mines non déterminées est égale au nombre restant de mines à trouver, alors toutes les cases hors de ces voisins n'ont pas de mine.

3 Critères d'évaluation

Les critères de la partie précédente restent valides. Merci de vous référer au sujet de la partie 1 en ce qui concerne l'évaluation du rapport, de la soutenance, des fonctionnalités du code et de l'organisation du code. L'aspect spécifique à cette partie est l'efficacité des algorithmes implémentés. Un générateur qui demanderait un temps de calcul rébarbatif ne serait pas satisfaisant.

4 Dates importantes

- Le code et le rapport en pdf (généré par latex) seront à rendre avant le mardi 26 mai à 23h59.
- La soutenance pour la troisième partie sera le vendredi 29 mai.