Université Pierre et Marie Curie

Résolution de problèmes

Résolution de Mots Croisés par un CSP.

Renaud ADEQUIN
Nadjet BOURDACHE

04/04/2016

1. Modélisation par un CSP et résolution

1. Pour résoudre ce problème, on propose une modélisation qui consiste à associer une variable à chaque mot de la grille. Les mots de la grille étants numérotés dans l'odre de leurs apparition dans la grille (d'abord les mots horizontaux puis les verticaux). On définit ensuite un ensemble de contraintes pour vérifier la cohérence de la grille générée.

   Variables :
   Pour m mots, on a m variables : Mot_i , $\forall\ i \in \{1, ..., m\}$.

   Domaine :
   Chaque mot de la grille doit appartenir au dictionnaire considéré, notons le $Dict$.
   $$\mathrm{D}(Mot_i) = \{X \in Dict : |X| = |Mot_i|\ \},\ \forall\ i \in \{1, ..., m\}\ .$$

   Contraintes :

   – Pour toute paire de mots Mot_i et Mot_j qui se croisent aux positions p pour Mot_i et q pour Mot_j, on définit la contrainte :
   $$Mot_i[p] \ = \ Mot_j[q].$$
   – Pour modéliser le fait qu'un même mot ne peut apparaître plus d'une fois dans la grille, il suffit d'ajouter la contrainte :
   $$\mathit{AllDiff}\ (Mot_1, Mot_2, \cdots, Mot_m)$$

2. D'un point de vue algorithmique, nous avons mis en œuvre une classe "Grille". Chaque objet de cette classe représente une instance de mots croisés, et contient une taille, un dictionnaire, une liste de tous les mots de la grille ayant une taille supérieure à 1 ainsi qu'une liste de cases noires.
   On pourra à partir d'un objet de cette classe, initialiser une grille à partir d'un fichier texte contenant une grille ou en générer une aléatoirement et la sauvegarder dans un fichier après résolution.

3. Les algorithmes AC3 et FC ont été développés tel qu'ils ont été définis en cours.

4. En revanche, pour le Conflict BackJumping, afin d'améliorer la résolution, nous avons :
   - Effectuer une arc-consistance sur les mots de la grille pour réduire les domaines avant de lancer l'algorithme.
   - Nous avons aussi ajouter un Check Forward avant l'appel récursif, ce qui nous permet d'éviter des appels inutiles et d'améliorer encore le temps de résolution des grilles les plus grandes.

5. Afin d'améliorer les trois algorithmes, nous avons utilisé une structure d'arbre pour sauvegarder et manipuler les dictionnaires et les domaines des différentes variables.
   Cette structure permet un parcours rapide du dictionnaire pour l'initialisation des domaines, et surtout, elle permet d'accélérer les algorithmes de filtrage, puisqu'elle peut permettre d'éliminer plusieurs mots d'un domaine en supprimant des sous arbres de ce dernier, nous évitant ainsi de devoir parcourir et tester chaque mot du domaine.

2. Expérimentations

1.

3. Extension au cas pondéré

1.