TD2 : Problèmes de Satisfaction de Contraintes

Exercice 1. Backtrack

On peut représenter une exécution de l'algorithme de Backtrack vu en cours en développant l'arbre de recherche dont les nœuds représentent une assignation (de plus en plus complète) ; la racine de l'arborescence représentant l'assignation vide. Chaque niveau de l'arbre est dédié à une contrainte. Pour faciliter la représentation on se limitera à indiquer sur chaque nœud la valeur assignée à la contrainte correspondant au niveau courant (l'assignation courante étant donc « lue » en remontant le chemin du nœud courant à la racine). Lorsqu'une assignation viole une contrainte, on indique par un **×** que le nœud engendre un « backtrack » en **précisant la** (ou les) **contrainte**(s) **violée**(s). On s'arrête à la première solution trouvée.

1) Exécuter l'algorithme Backtrack sur le problème de coloration de carte vu en cours en considérant à chaque choix de variables à assigner l'ordre `WA,Q,T,SA,NSW,V,NT` et à chaque choix de valeurs l'ordre `R,G,B`.
2) Même question en considérant les variables dans l'ordre `WA,NT,NSW,Q,V,SA,T` et les valeurs dans l'ordre `R,G,B`.
3) Quel impact l'ordre d'affectation des variables a t'il sur l'arbre de recherche ? Et celui des valeurs ? Discuter de l'importance du choix de ces ordres pour la recherche d'une solution et pour celle de toutes les solutions.
4) Calculer un ordre d'application des variables en appliquant l'heuristique min-width (et en cas d'égalité suivre l'ordre lexicographique) puis tracer l'exécution de Backtrack avec cet ordre.
5) Calculer un ordre d'application des variables en appliquant l'heuristique max-card (et en cas d'égalité max-degree puis suivre l'ordre lexicographique si encore égalité) puis tracer l'exécution de Backtrack avec cet ordre.
6) Préciser à partir de l'ordre d'application des variables calculées dans la question précédente, quelles contraintes doivent « effectivement » être testées à chaque niveau de l'arbre de recherche.
7) Proposer alors un algorithme pour le test de consistance qui optimise le nombre de contraintes à vérifier à chaque étape.

Exercice 2. Cryptogrammes

Les puzzles arithmétiques (ou cryptogrammes) sont des opérations où une lettre correspond à un chiffre. On donne l'opération en lettres, il faut trouver celle en chiffres.

```
  L I O N N E
+   T I G R E
-------------
= T I G R O N
```

La résolution de ces puzzles peut être modélisée comme un CSP.

1) Modéliser le cryptogramme précédent par un CSP. Vous donnerez d'abord les contraintes sous la forme d'équations linéaires puis quand cela est raisonnable donnerez la version « en extension » des contraintes.
2) Représenter graphiquement le réseau de contraintes (on dessinera le biparti d'incidence variables/contraintes).

Exercice 3. N-reines

Le problème des n-reines consiste à positionner *n* reines sur un échiquier de *n*x*n* cases de telles sortes qu'elles ne se menacent pas 2 à 2. On rappelle qu'aux échecs une reine peut se déplacer d'autant de cases qu'elle veut sur sa ligne, sa colonne et ses diagonales. Par conséquent deux reines ne doivent pas partager une même ligne, colonne ou diagonale. Proposer une modélisation CSP pour le problème des 4 reines. Vous donnerez les contraintes en extension puis réfléchirez à un langage permettant d'exprimer les contraintes en intension et permettant une généralisation du problème pour un nombre quelconques de reines.