Universités d'Aix-Marseille I & III

Licence Informatique – Intelligence Artificielle

TD n°1 : Problèmes de satisfaction de contraintes

1. Modélisation

Modélisez les problèmes suivants sous forme de CSP :

1.1. Problème des pigeons :
Le problème des pigeons consiste à placer n pigeons dans un pigeonnier qui possède $n-1$ nids en respectant les deux règles suivantes :

(i) chaque pigeon doit occuper exactement un nid,

(ii) chaque nid doit être occupé par exactement un pigeon.

1.2. Problème des n-dames :
Le problème des n-dames consiste à placer n dames sur un "échiquier" $n \times n$ de telle manière qu'aucune ne puisse être en prise par rapport aux autres, c'est-à-dire qu'il n'y en ait pas plus d'une dame par rangée, colonne, diagonale et anti-diagonale. Afin de simplifier la modélisation du problème, on placera une dame par rangée. Le problème consiste alors à déterminer sur quelle colonne placer la dame de chaque rangée.

1.3. Problème cryptarithmétique SEND + MORE = MONEY :
Le problème consiste à associer à chaque lettre un chiffre de telle sorte que la somme soit valide et que chaque chiffre ne soit pas utilisé plus d'une fois.

2. Backtracking

2.1. Résolvez le problème des pigeons avec $n = 3$ en utilisant BT.

2.2. Cherchez une solution aux problèmes des 3-dames et 4-dames en utilisant BT.

3. Forward Checking

3.1. BT teste la compatibilité entre la valeur de la dernière variable instanciée et les valeurs des variables précédemment instanciées. FC fait à la place des tests de compatibilité entre la valeur de la dernière variable instanciée et les valeurs possibles des variables restant à instancier. Question-piège : pourquoi tester la compatibilité avec une valeur de variable x pas encore instanciée alors qu'il est possible qu'il se produise un échec avant qu'on ait instancié x ? Pourquoi espère-t-on quand même qu'il y aura moins de tests de contraintes avec FC ?

3.2. Existe-t-il des cas où BT est meilleur que FC (en terme de tests de contraintes) ? Si oui, donnez un exemple.

3.3. Résolvez le problème des pigeons avec $n = 3$ et $n = 4$ en utilisant FC.

3.4. Cherchez toutes les solutions des problèmes des 4-dames et des 5-dames en utilisant FC.

4. Backjumping

4.1. Vérifiez s'il existe une solution au problème des 6-dames avec la première dame en colonne 1, la deuxième en colonne 3 et la troisième en colonne 5 en utilisant BJ.

4.2. BJ visite-t-il des nœuds que ne visite pas BT ? FC visite-t-il des nœuds que ne visite pas BJ ?

4.3. BJ teste-t-il toujours moins de contraintes que BT ? FC teste-t-il toujours moins ou plus de contraintes que BJ ? En rajoutant les mécanismes de FC et BJ à BT, l'algorithme de recherche résultant est-il potentiellement meilleur que FC ?

5. Heuristique de choix de variable

Considérons un CSP à trois variables dont la première a un domaine de taille 1, la deuxième un domaine de taille 2 et la troisième un domaine de taille 3 et dont l'ensemble des contraintes est vide. Dans quel ordre vaut-il mieux instancier les variables pour que le nombre de nœuds générés soit le plus petit lorsqu'on génère toutes les solutions ? Déduisez-en une heuristique générale de choix des variables. Quelle est la complexité de cette heuristique ?

6. Heuristique de choix de valeur

Considérons qu'on se trouve en un nœud donné d'un arbre de recherche en train d'être développé grâce à FC. La prochaine variable x à instancier a été choisie. S'il n'existait aucune contrainte mettant en jeu les variables non instanciées, comment pourrait-on déterminer le nombre de solutions issu de chaque choix de valeur pour x ? Déduisez-en une heuristique générale de choix de valeur. Quelle est la complexité de cette heuristique ?

7. Systèmes d'équations

Soit le système de 3 équations à 3 inconnues suivant :

$$\left| \begin{array}{c} x \times y \leq 5 \\ x^3 - z^3 \geq 10 \\ y + z = 4 \end{array} \right.$$

avec x, y et z toutes définies sur l'ensemble $\{0, 1, 2, 3, 4\}$.

7.1 Représentez les 3 contraintes binaires (du CSP correspondant à ce système d'équations) sous forme explicite, c'est-à-dire, en indiquant pour chacune d'elles l'ensemble des couples de valeurs autorisés.

7.2 Cherchez s'il existe une solution telle que $x = 2$ en utilisant l'algorithme Backtracking : représentez l'arbre de recherche en indiquant la contrainte qui est la cause d'un échec lors d'un retour arrière. Vous pouvez choisir l'ordre d'affectation des variables qui vous convient le mieux.

7.3 Cherchez toutes les solutions du problème en utilisant le Forward-Checking : représentez l'arbre de recherche en indiquant quel est le domaine vide qui est la cause d'un échec lors d'un retour arrière. Vous pouvez choisir l'ordre d'affectation des variables qui vous convient le mieux.

7.4 Nous allons chercher une heuristique générale de choix de variable pour les systèmes d'équations en fonction du type des équations. On considère qu'il existe 3 types d'équations : $f(x_i, x_j) = k$, $f(x_i, x_j) \neq k$ et $f(x_i, x_j) \geq k$, où f est une fonction mathématique calculable quelconque et k est une constante.

Dans le cas général où on ne dispose pas d'autres informations que le type des équations, quel est le type d'équation susceptible d'autoriser le plus (respectivement : le moins) de couples de valeurs pour x_i et x_j ? Sachant qu'une heuristique de choix de variable a pour fonction de chercher une variable qui aura le moins de successeurs possibles, déduisez-en une heuristique générale de choix de variable pour ces systèmes d'équations.