Universités d'Aix-Marseille I & III

Licence Informatique – Intelligence Artificielle

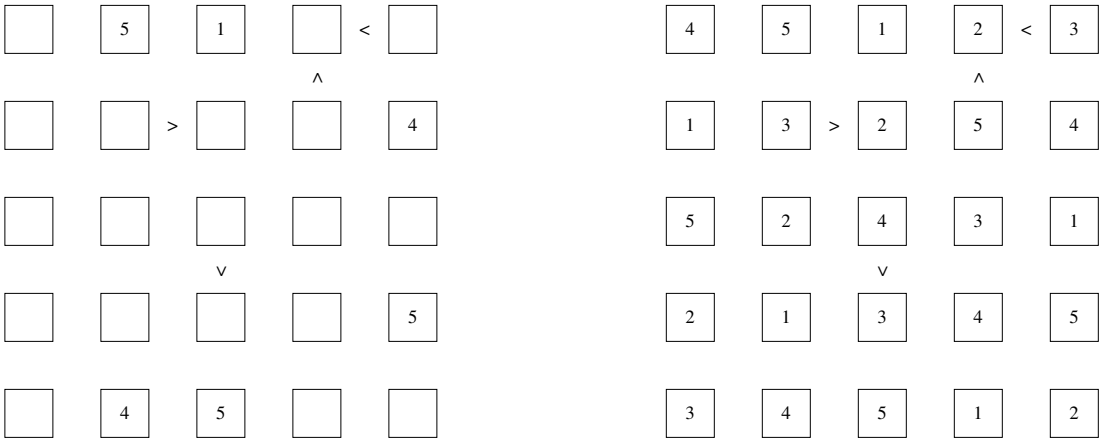
Projet

Dans ce projet, nous allons nous intéresser à un jeu logique proche du sudoku : le futoshiki. A l'image du sudoku, le futoshiki consiste à compléter une grille $n \times n$ avec des chiffres compris entre 1 et n de sorte que chaque chiffre ne soit présent qu'une et une seule fois par ligne et par colonne. Initialement, certaines cases peuvent être déjà remplies. De plus, des indices sont fournies sur les valeurs des cases par le biais de relation de supériorité ou d'infériorité entre les valeurs de deux cases voisines.

1 Présentation

Une instance futoshiki se définit par la donnée d'une grille carrée de $n \times n$ cases dont certaines peuvent contenir un chiffre entre 1 et n et d'un certain nombre d'indices. Un indice est simplement une relation de supériorité ou d'infériorité entre les valeurs de deux cases voisines. Les indices permettent donc de restreindre le nombre de possibilités pour les valeurs des cases concernées. L'objectif est de placer des chiffres de 1 à n dans chaque case vide de sorte que chaque chiffre ne soit présent qu'une seule fois par ligne et par colonne et que les conditions imposées par les indices soient satisfaites. Normalement, une vraie instance de futoshiki ne possède qu'une seule solution. Quant à la difficulté d'une grille, elle dépend bien entendu du nombre de cases remplies initialement et du nombre d'indices. A noter qu'il est possible qu'aucune case ne soit remplie initialement.

Un exemple d'instance futoshiki est donnée dans la partie (a) de la figure ci-dessous. La solution de cette grille est présentée dans la partie (b).



(a) (b)

Les objectifs de ce projet sont d'utiliser les techniques d'IA (ici de programmation par contraintes) pour trouver une solution d'une grille donnée.

2 Modélisation d'une instance

Pour rechercher une solution d'une instance futoshiki, nous allons modéliser le problème sous la forme d'un CSP binaire ou d'un CSP n-aire. Dans les deux cas, on associera une variable par case de la grille. Chaque

variable correspondant à une case vide aura pour domaine initial $\{1, 2, \ldots, n\}$. Chaque variable correspondant à une case ayant une valeur initiale aura un domaine réduit à cette valeur. Quant aux contraintes, nous utiliserons deux types de contraintes :

- des contraintes de différence (une contrainte de différence impose aux variables sur lesquelles elle porte d'avoir toutes des valeurs différentes). En pratique, dans la modélisation sous la forme d'un CSP binaire, deux variables seront liées par une contrainte binaire de différence si elles apparaissent dans une même ligne ou une même colonne. Dans le cas de la modélisation sous la forme d'un CSP n-aire, on aura une contrainte n-aire de différence par ligne et par colonne.
- des contraintes binaires correspondant aux indices qui lieront les variables correspondant à deux cases voisines reliées par un indice.

3 Travail à réaliser

Pour des raisons d'efficacité, tous les algorithmes de résolution seront implémentés de façon itérative. Le travail qui vous est demandé est le suivant :

(i) Implémenter une version binaire et une version n-aire de l'algorithme Backtrack,

(ii) Implémenter une version binaire et une version n-aire de l'algorithme Forward-Checking,

(iii) Pour chaque modélisation, comparer l'efficacité de Backtrack et de Forward-Checking en termes de temps, de nombre de nœuds et de nombre de tests de contraintes. Puis comparer les deux modélisations entre elles. Quelques grilles de jeu vous seront données ultérieurement pour effectuer ces comparaisons. Vous pouvez bien entendu tester sur vos propres grilles.

(iv) Proposer des heuristiques de choix de variables.

(v) Proposer des améliorations de l'algorithme Forward-Checking en tenant des spécificités du jeu.

(vi) Comparer l'efficacité des heuristiques et des améliorations proposées avec votre Forward-Checking initial doté d'une heuristique de type *dom/deg* en termes de temps, de nombre de nœuds et de nombre de tests de contraintes.

4 Rendu du travail

Vous devrez fournir les code sources en C des différents algorithmes ainsi qu'un rapport. Ce rapport devra contenir :

1. Un manuel d'utilisation de votre programme.
2. Les algorithmes Backtrack et Forward-Checking en version binaire et n-aire.
3. Les résultats et/ou courbes de comparaisons entre Backtrack et Forward-Checking d'une même modélisation.
4. Les résultats et/ou courbes de comparaisons entre les modélisations.
5. Vos heuristiques de choix de variables.
6. L'algorithme de votre Forward-Checking amélioré.
7. Les résultats et/ou courbes de comparaisons entre les différentes versions de Forward-Checking.
8. Pour chaque comparaison, vous donnerez une interprétation des résultats obtenus.