Projet d'Intelligence Artificielle

Résolution d'un kakuro par l'implémentation d'un CSP

1) Compilation et exécution du solveur

Utilisation du programme ***makefile*** pour la compilation.

- **make** : compilation complète du projet.
- **make clean** : suppression des fichiers .o ainsi que l'exécutable du projet.

Exécution du programme en ligne de commande.

- **./main <grille_de_jeu>** où <grille_de_jeu> est un fichier présent dans le dossier *input*.

2) Présentation des fichiers

Les fichiers de code sont présents dans le dossier *src*.

-

3) Déroulement du programme

- {main} → vérification de l'usage de l'exécutable
- {parseur} → lecture du fichier d'entrée
- {parseur} → création de la grille de jeu à partir du code fourni
- {parseur} → initialisation du CSP
- {backtrack} → résolution du CSP
- {main} → affichage des statistiques, à savoir le nombre de tests de contrainte, le nombre de nœuds de l'arbre de recherche ainsi que le temps d'exécution.
- {main} → nettoyage de la mémoire dynamique

4) Structures de données utilisées

Utilisation de la structure fournie pour la grille de jeu.

Concernant le CSP, j'utilise trois structures, deux relatives aux contraintes, une permettant de gérer les variables.

Les structures de contraintes sont équivalentes, à l'exception d'une variable près. J'ai choisi de les séparer afin d'économiser une opération de test supplémentaire : on pourrait en effet utiliser une énumération *type* par exemple afin de distinguer les deux au sein d'une même structure comme ci-dessous :

```
struct s_contrainte
{
        enum e_type type_contrainte;
        int somme;
        int arite;
        int* portee;
};
```

Vous noterez l'utilisation d'un pointeur d'entiers *portee* contenant les numéros des variables sur lesquelles portent ladite contrainte. Celui-ci aurait pu être remplacé par un pointeur de structures *s_variable* directement, ce qui n'a pas été fait dans un souci de facilité d'implémentation.

Cette implémentation permet de conserver une complexité en temps O(1). En effet la contrainte de différence impose que la longueur d'un mot soit comprise entre 1 et 9. Donc le nombre total de contraintes dans lequel apparaît une variable est borné :

- 8 contraintes de somme horizontale,
- 8 contraintes de somme verticale,
- 16 contraintes de différence.

La structure modélisant une variable du CSP comprend, outre les champs qui n'ont pas besoin d'être précisés :

- la taille du domaine afin de permettre l'implémentation d'heuristiques,
- les 3 contraintes dans lesquelles elle peut apparaître (afin d'éviter de multiplier les structures).

5) Heuristiques

- Filtrage des domaines d'une variable en retirant les valeurs supérieures ou égales à la contrainte de somme qu'elle possède.
- Type dom/deg à savoir le rapport de la taille du domaine sur le nombre de contraintes dans lesquelles apparaît la variable. (todo)
- N'instancier les variables seules soit sur une ligne soit sur une colonne. (todo)