Evaluation TP1 Intelligence artificielle – HMIN107

Au préalable :

Vérifier que votre programme calcule bien pour les méthodes backtrack et backtrallAll le nombre de nœuds explorés dans l'arbre de recherche, i.e. le nombre d'assignations partielles ou complètes explorées. En particulier, attention une assignation partielle jugée non consistante doit être comptée dans les assignations explorées même si elle n'a engendré aucun appel récursif.

Ajouter deux variables de classe (*public static int appel* et *public static int verif*) à votre classe ConstraintExt afin de pouvoir calculer lors de l'exécution d'un backtrack :

- le nombre d'appels à la fonction violation de la classes ContraintesExt
- le nombre total de fois où l'on vérifie effectivement dans la fonction violation qu'un tuple appartient aux tuples autorisés de la contraintes (on exclut donc les appels à violation qui concluent sans parcourir les tuples de la contrainte).

Par ailleurs, on souhaite pouvoir connaître le nombre de solutions calculées par searchAllSolutions.

Travail à faire

1) Réaliser une classe **`Demo`** comportant une méthode **`main`** effectuant les 8 tâches décrites dans la section *démonstration* ci-après.
2) Lancer l'exécution de ce programme.
3) Créer un fichier **`resultat`** au format *ascii* contenant :
   a) votre spécialité : Decol, Imagina, MIT, Aigle,
   b) une copie du résultat console de l'exécution de votre programme,
4) Créer une archive au format **`zip`** contenant :
   - vos fichiers **sources** (a priori ceux du répertoire src)
   - le fichier **`resultat`**
   - vos fichiers de données contenant les différents réseaux utilisés :
     - ceux demandés dans le TP : `coloration.txt, zebre.txt`
     - ceux de cette évaluation : `csp1.txt, csp2.txt, colore.txt, 10reines_Ext.txt, 10reines_Exp.txt, 20reines_Ext.txt, cryptoMoney.txt`
     - la version en intension du réseau : `cryptoMoneyIntension.txt`
5) Déposer l'archive **sur Moodle avant 18h00** (attention fermeture automatique) !

Démonstration

1. Résolution des réseaux de contraintes en extension des problèmes `csp1`, `csp2` et `coloration` en recherchant 1 solution puis toutes les solutions (fichiers `csp1.txt` et `csp2.txt` sur Moodle et votre fichier `coloration.txt` fait lors de l'étape 2 du TP). Votre programme doit afficher les informations suivantes :

   Réseau : `nom du réseau`
   searchSol :
   Première solution trouvée :
   Nombre de nœuds explorés :
   Nombre d'appels à violation :
   Nombre de vérifications de tuples :
   searchAllSol :
   Nombre de solutions trouvées :
   Nombre de nœuds explorés :
   Toutes les solutions trouvées :

2. Résolution du problème de coloration en avec contraintes en intension `dif` et `eq` (fichier `colore.txt` sur Moodle) en recherchant 1 solution puis toutes les solutions. Votre programme doit afficher :

   Réseau : colore
   searchSol :
   Première solution trouvée :
   Nombre de nœuds explorés :
   searchAllSol :
   Nombre de solutions trouvées :
   Nombre de nœuds explorés :
   Toutes les solutions trouvées :

3. Résolution du problème du zèbre (avec le fichier que vous avez dû faire lors de l'étape 5 du TP) en recherchant 1 solution puis toutes les solutions. Votre programme doit afficher les informations suivantes :

Réseau : zebre
searchSol :
Première solution trouvée :
Nombre de nœuds explorés :
searchAllSol :
Nombre de solutions trouvées :
Nombre de nœuds explorés :
Toutes les solutions trouvées :

4. Résolution des 10 reines en recherchant 1 solution puis toutes les solutions (fichier `10reines_Ext.txt` sur Moodle). Votre programme doit afficher les informations suivantes :

Réseau : 10reines_Ext
searchSol :
Première solution trouvée :
Nombre de nœuds explorés :
searchAllSol :
Nombre de solutions trouvées :
Nombre de nœuds explorés :

5. Résolution des 10 reines en intension en recherchant 1 solution puis toutes les solutions (fichier `10reines_Exp.txt` sur Moodle). Votre programme doit afficher les informations suivantes :

Réseau : 10reines_Exp
searchSol :
Première solution trouvée :
Nombre de nœuds explorés :
searchAllSol :
Nombre de solutions trouvées :
Nombre de nœuds explorés :

6. Résolution des 20 reines en extension en recherchant 1 solution seulement. Attention, il y a près de 40 milliards de solutions (fichier `20reines_Ext.txt` sur Moodle). Votre programme doit afficher les informations suivantes :

Réseau : 20reines_Ext
searchSol :
Première solution trouvée :
Nombre de nœuds explorés :

7. Résolution du cryptogramme SEND+MORE=MONEY (fichier `cryptoMoney.txt` sur Moodle) en recherchant 1 solution puis toutes les solutions. Votre programme doit afficher :

Réseau : cryptoMoney
searchSol :
Première solution trouvée :
Nombre de nœuds explorés :
searchAllSol :
Nombre de solutions trouvées :
Nombre de nœuds explorés :
Toutes les solutions trouvées :

8. Résolution du cryptogramme précédent en le reformulant en intension dans un fichier `cryptoMoneyIntension.txt` en recherchant 1 solution puis toutes les solutions. Votre programme doit afficher :

Réseau : CryptoMoneyIntension
searchSol :
Première solution trouvée :
Nombre de nœuds explorés :
searchAllSol :
Nombre de solutions trouvées :
Nombre de nœuds explorés :
Toutes les solutions trouvées :