USTHB, FEI, Département d'Informatique

LMD Master 2 ''Systèmes Informatiques Intelligents'' 2018/2019

Module ''Programmation Par Contraintes''

Programmation par Contraintes : projet

Recherche récursive de solutions d'un CSP binaire discret :

Implémentation de Look-Ahead avec PC2 comme procédure de filtrage durant la recherche

Le but du projet est l'implémentation pour les CSP binaires discrets d'un des quatre algorithmes de recherche de solutions vus en cours, l'algorithme Look-Ahead, avec la politique suivante :

- Ordre d'instanciation des variables
  - **Version 1 :** tirage aléatoire de la prochaine variable à instancier
  - **Version 2 :** ordre dynamique donné par l'heuristique « instancier la variable dont le cardinal du domaine est le plus petit »
- Ordre statique de choix des valeurs du domaine : ordre croissant
- Filtrage durant la recherche avec la procédure de consistance de chemin PC2.

Le travail demandé est comme suit :

- Implémentation de l'algorithme de consistance de chemin PC2
- Implémentation de l'algorithme Look-Ahead de recherche récursive de solutions d'un CSP binaire discret, avec la politique décrite ci-dessus
- Explication des structures de données utilisées
- Comparaison des deux versions sur des CSP générés aléatoirement
- Laisser la possibilité à l'utilisateur d'entrer ses propres CSP

Autres détails : génération aléatoire des entrées sous forme de représentations matricielles, avec l'une des deux méthodes suivantes :

- **Méthode 1 :**
  - Générer aléatoirement le nombre n de variables dans un intervalle [a,b] (exemple [a,b]=[20,40])
  - Générer aléatoirement le cardinal p du domaine commun des variables dans un intervalle [c,d] (exemple [c,d]=[10,20])

- Sans contraintes, la représentation matricielle M_P à générer, qui est une matrice nxn, serait comme suit : matrice identité pxp sur la diagonale, matrice universelle pxp en dehors de la diagonale
- Générer aléatoirement chaque entrée [i,j] de M_P, avec i<j (**pour** k=1 à p **faire pour** l=1 à p **faire** générer aléatoirement M_P[i,j][k,l] dans {0,1} **fait fait**)
- Les entrées [i,j] de M_P, avec i>j (triangle inférieur, sans la diagonale), sont les transposées des entrées [j,i]
- Générer aléatoirement chaque entrée (i,i] de M_P (**pour** j=1 à p **faire** générer aléatoirement M_P[i,i][j,j] dans {0,1} **fait**)

- **Méthode 2 :**
  - Générer aléatoirement le nombre n de variables dans un intervalle [a,b] (exemple [a,b]=[20,40])
  - Générer aléatoirement le cardinal p du domaine commun des variables dans un intervalle [c,d] (exemple [c,d]=[10,20])
  - Sans contraintes, la représentation matricielle M_P à générer, qui est une matrice nxn, serait comme suit : matrice identité pxp sur la diagonale, matrice universelle pxp en dehors de la diagonale
  - Générer aléatoirement le nombre m de contraintes dans un intervalle [e,f] (exemple [e,f]=[20,30]) :
  - Pour k=1 à m :
    - Générer aléatoirement la paire (X_i,X_j) de variables sur laquelle doit porter la contrainte c_k, avec i≤j
    - Générer aléatoirement la contrainte c_k :

      <u>Si i<j :</u> pour k=1 à p **faire pour** l=1 à p **faire** générer aléatoirement M_P[i,j][k,l] dans {0,1} **fait fait**

      <u>Si i=j :</u> pour j=1 à p **faire** générer aléatoirement M_P[i,i][j,j] dans {0,1} **fait** (les entrées de M_P[i,i] en dehors de la diagonale sont à 0)

Remarque : Pour la méthode 1 de génération aléatoire des entrées, le nombre de paires de variables sur lesquelles il y a des contraintes est décidé par la génération aléatoire des entrées de M_P :

- pour chaque paire (X_i,X_j) de variables, avec i≠j, si l'entrée [i,j] n'est pas la matrice universelle, il y a des contraintes sur X_i et X_j, qui sont représentées par M_P[i,j]
- pour chaque variable X_i, si l'entrée [i,i] n'est pas la matrice identité, il y a des contraintes unaires sur X_i, qui sont représentées par M_P[i,i]

Toujours pour la méthode 1, la probabilité que la matrice booléenne générée pour une entrée [i,j] de M_P avec i<j (respectivement, avec i=j) soit la matrice universelle (respectivement, la

matrice identité) est faible. Il faut donc s'attendre à ce que les CSP générés par cette méthode aient un nombre important de variables sur lesquelles il y a des contraintes.

Travail à remettre :

- le solveur sur CD
- un rapport incluant
    - la description du solveur avec explication de la procédure principale LookAhead, de la procédure de filtrage PC2, et des structures de données utilisées
    - le résultat de la comparaison des deux versions sur des CSP générés aléatoirement avec une des deux méthodes ci-dessus