USTHB - FEI - Département d'Informatique
Master 2 ''Systèmes Informatiques Intelligents'' 2019/2020
Module ''Programmation par Contraintes''
Date : 25/01/2020 <13h00-14h15>

Corrigé de l'examen

Exercice 1 (12 points=2+2+(2+2+2+1+1))

I. Donner un CSP de taille supérieure ou égale à 3, trivialement consistant d'arc mais inconsistant.

II. Donner un CSP de taille supérieure ou égale à 4, trivialement consistant de chemin mais inconsistant.

III. On considère le CSP binaire discret P=(X,D,C) suivant :

- $X=\{X_1,X_2,X_3\}$
- $D(X_1)=D(X_2)=D(X_3)=\{4,5,6\}$
- $C=\{c_1 : X_1<X_2, c_2 : X_2<X_3\}$

1. Appliquez l'algorithme de recherche SRA (Simple Retour Arrière) au CSP P
2. Appliquez l'algorithme de recherche FC (Forward-Checking) au CSP P
3. Appliquez au CSP P l'algorithme de recherche Look-Ahead avec AC3 comme procédure de filtrage
4. Donnez la représentation graphique de P
5. Donnez la représentation matricielle de P

Pour chacun des trois algorithmes de recherche de la partie III, il est demandé de respecter l'ordre statique X_1, X_2, X_3 d'instanciation des variables ; et l'ordre statique 4,5,6 de choix des valeurs du domaine de la variable en cours d'instanciation. Il est également demandé d'expliquer tous les échecs éventuels.

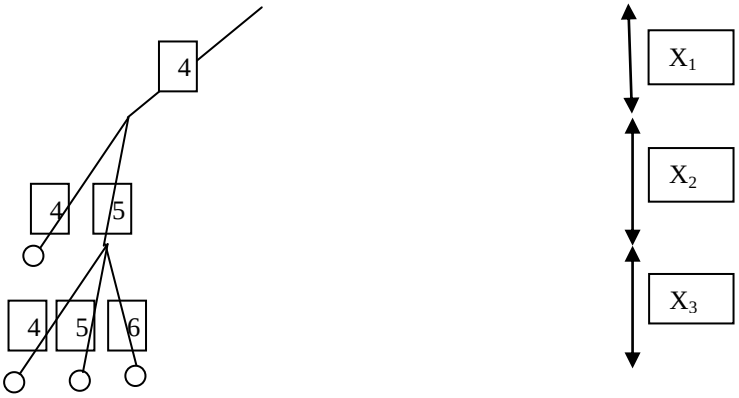
Solution :

I. Le CSP P modélisant l'instance suivante de coloriage d'un graphe : coloriage d'un triangle avec deux couleurs. $P=(X,D,C)$ avec $X=\{X_1,X_2,X_3\}$, $D(X_1)=D(X_2)=D(X_3)=\{1,2\}$, $C=\{c_1 : X_1\neq X_2, c_2 : X_1\neq X_3, c_3 : X_2\neq X_3\}$. P est trivialement inconsistant (on ne peut pas colorier un triangle avec seulement deux couleurs, sans que deux des trois sommets aient la même couleur). Néanmoins, P est consistant d'arc : pour toute paire (X_i,X_j) de variables, pour toute valeur V_i de $D(X_i)=\{1,2\}$, il existe V_j dans $D(X_j)=\{1,2\}$ telle $(X_i,X_j)=(V_i,V_j)$ satisfait la contrainte sur X_i et X_j (V_j support de V_i).

II. Le CSP P modélisant l'instance suivante de coloriage d'un graphe : coloriage du graphe complet à 4 sommets, K4, avec trois couleurs. $P=(X,D,C)$ avec $X=\{X_1,X_2,X_3,X4\}$, $D(X_1)=D(X_2)=D(X_3)= D(X_4)=\{1,2,3\}$, $C=\{c_1 : X_1\neq X_2, c_2 : X_1\neq X_3, c_3 : X_1\neq X_4 , c_4 : X_2\neq X_3, c_5 : X_2\neq X_4, c_6 : X_3\neq X_4\}$. P est trivialement inconsistant (on ne peut pas colorier le graphe complet à 4 sommets avec seulement trois couleurs, sans que deux des quatre sommets aient la même couleur). Néanmoins, P est consistant de chemin.

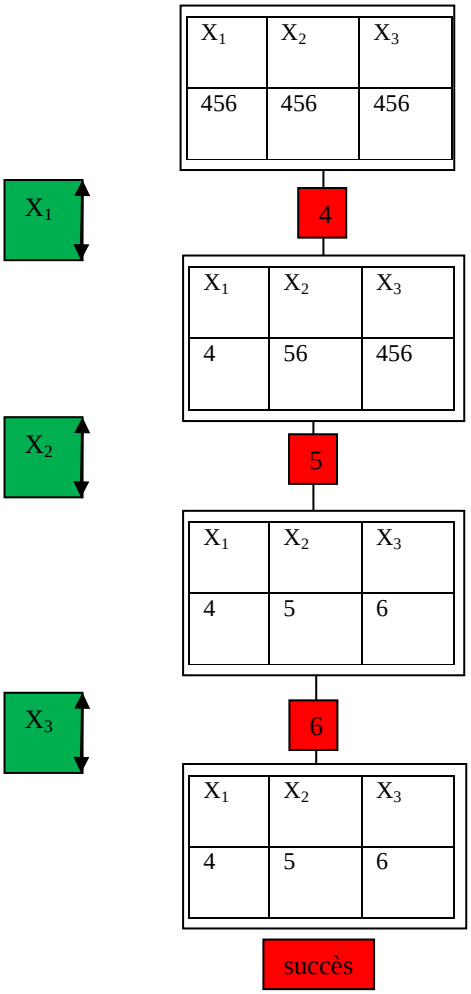
III.

1) SRA :



Les 3 premières feuilles de l'arbre de recherche, de gauche à droite, correspondent aux échecs de l'algorithme SRA, pour lesquels nous donnons ci-dessous, respectivement, une contrainte non satisfaite entre la variable en cours d'instanciation et une variable déjà instanciée : c_1, c_2, c_2. La feuille la plus à droite correspond au succès de SRA.

2) FC :

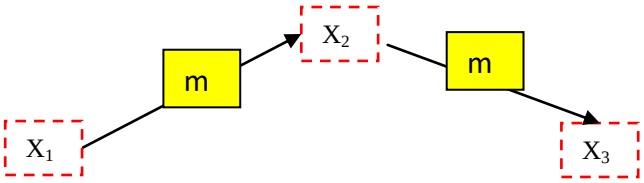


3) LookAhead :

X_1	X_2	X_3
4	5	6

Le prétraitement (application de AC3 au CSP de départ) aboutit à des domaines singletons : le CSP est donc consistant, avec solution donnée par les domaines. L'arbre de recherche se réduit donc à sa racine.

4) **Représentation graphique :** Gp=(X,E,l)



Avec $m=\begin{bmatrix}0 & 1 & 1\\ 0 & 0 & 1\\ 0 & 0 & 0\end{bmatrix}$

5) **Représentation matricielle M_P :**

	X_1	X_2	X_3
X_1	I_3	m	U_{3x3}
X_2	m_2	I_3	m
X_3	U_{3x3}	m_2	I_3

Avec $m_2=\begin{bmatrix}0 & 0 & 0\\ 1 & 0 & 0\\ 1 & 1 & 0\end{bmatrix}$ (m_2 est la transposée de m)

Exercice 2 (3 points=1,5+1,5)

1. Comment transformer un STP en un graphe orienté pondéré de telle qu'on ait : le STP est consistant si et seulement si le graphe n'a pas de circuits de longueur négative ?
2. Soit le TCSP P=(X,C) suivant :

$$\mathbf{X}=\{X_0,X_1,X_2\},\ \mathbf{C}=\{c_1 : (X_1-X_0)\in[3,12],\ c_2 :(X_2-X_0)\in[5,20],\ c_3 : (X_2-X_1)\in[4,7]\}$$

Le TCSP est-il consistant de chemin ? Expliquez.

Solution :

1. Soit P=(X,C) un STP avec $X=\{X_0,X_1,\ldots,X_n\}$, X_0 étant la variable ''origine du monde''. Le graphe demandé est G=(X,E,l) avec X l'ensemble des sommets, E l'ensemble des arcs et l la fonction de pondération. E et l sont construits comme suit :
   a. Initialiser E à l'ensemble vide : $E=\emptyset$
   b. Pour toute contrainte $(Xj-Xi)\in Cij$, avec Cij de l'une des formes $[a,+\infty[$ ou $[a,b]$, créer l'arc (Xj,Xi) et l'étiqueter (le pondérer) par -a :
      $E=E\cup\{(Xj,Xi)\}$, $l(Xj,Xi)=-a$
   c. Pour toute contrainte $(Xj-Xi)\in Cij$, avec Cij de l'une des formes $]-\infty,b]$ ou $[a,b]$, créer l'arc (Xi,Xj) et l'étiqueter par b :

$$E=E\cup\{(Xi,Xj)\},\ l(Xi,Xj)=b$$

La procédure est restreinte aux STP dont les contraintes sont de la forme $(Xj-Xi)\in Cij$ avec les bornes finies de Cij fermées. Mais elle est généralisable.

2. Le TCSP P, qui se trouve être un STP, n'est pas consistant de chemin : P_{02} n'est pas inclus dans la composition $P_{01}°P_{12}$: $P_{01}°P_{12}=[3,12]°[4,7]=[7,19]$

Exercice 3 (5 points=3,5+1,5) :

A. On représente une matrice mxn par une liste de m listes de longueur n. Ecrire un programme Prolog implémentant le prédicat **motif**(X,M1,M2) suivant, avec M1 et M2 deux matrices mxn. **motif**(X,M1,M2) ssi $\exists k\geq 0$, k minimal, tel que
$\forall(i,j)\in\{1,\ldots,m\}x\{1,\ldots,n\}$: **si M1[i,j]=X alors j+k>n ou M2[i,j+k]=X**
En d'autres termes, **motif**(X,M1,M2) ssi les X de M1 se retrouvent dans M2, décalés de k vers la droite, avec perte éventuellement si dépassement (décalage à droite).

B. Ecrire un programme Prolog faisant appel au solveur de contraintes sur domaines finis pour la résolution du CSP de l'exercice 1.III.

Solution :

A.

```
%
% motif(1,[[1,1,0,0],[0,1,1,0],[0,0,1,0]],[[1,1,1,0],[0,1,1,1],[0,0,1,1]]).
% motif(1,[[1,1,0,0],[0,1,1,0],[0,0,1,0]],[[0,0,1,1],[0,0,0,1],[0,0,0,0]]).
% motif(1,[[1,1,0,0],[0,1,1,0],[0,0,1,0]],[[0,1,1,0],[0,0,1,1],[1,1,0,1]]).
% motif(1,[[1,1,0,0],[0,1,1,0],[0,0,1,0]],[[0,0,0,1],[0,1,0,1],[1,1,1,0]]).
%
motif(X,M1,M2):-length(M1,L1),
        length(M2,L1),
        longueur_lignes(M1,L2),
        longueur_lignes(M2,L2),
        write('Matrice M1='),
        nl,
        imprimer_matrice(M1),
        nl,
        write('Matrice M2='),
        nl,
        imprimer_matrice(M2),
        decalage(X,M1,M2,0).
longueur_lignes([],_):-!.
longueur_lignes([L|M],X):-length(L,X),
        longueur_lignes(M,X).
imprimer_matrice([]):-!.
imprimer_matrice([L|M]):-write(L),
        nl,
        imprimer_matrice(M).
decalage(X,M1,M2,K):-decalage_lignes(X,M1,M2,K),
        !.
decalage(X,M1,M2,K):-K2 is K+1,
        decalage(X,M1,M2,K2).
decalage_lignes(X,[],_,K):-!,
```

```
        nl,
        write('Le motif X='),write(X),
        nl,
        write('La longueur K du décalage='),write(K).
decalage_lignes(X,[L1|M1],[L2|M2],K):-decalage_ligne(X,L1,L2,K),
        decalage_lignes(X,M1,M2,K).
decalage_ligne(_,_,L,K):-length(L,K2),
        K2moins1 is K2-1,
        K>K2moins1,
        !.
decalage_ligne(X,[X|L1],L2,K):-!,
        length(L3,K),
        append(L3,L4,L2),
        append([X],_,L4),
        !,
        append([_],L5,L2),
        decalage_ligne(X,L1,L5,K).
decalage_ligne(X,[_|L1],[_|L2],K):-decalage_ligne(X,L1,L2,K).
```

Déroulement :

```
1 ?- motif(1,[[1,1,0,0],[0,1,1,0],[0,0,1,0]],[[0,1,1,0],[0,0,1,1],[1,1,0,1]]).
Matrice M1=
[1,1,0,0]
[0,1,1,0]
[0,0,1,0]

Matrice M2=
[0,1,1,0]
[0,0,1,1]
[1,1,0,1]

Le motif X=1
La longueur K du décalage=1
true.

 2 ?- motif(1,[[1,1,0,0],[0,1,1,0],[0,0,1,0]],[[0,0,1,1],[0,0,0,1],[0,0,0,0]]).
Matrice M1=
[1,1,0,0]
[0,1,1,0]
[0,0,1,0]

Matrice M2=
[0,0,1,1]
[0,0,0,1]
[0,0,0,0]

Le motif X=1
La longueur K du décalage=2
true.

 3 ?-
```

B.

```
exo1III(Vars):-Vars=[X1,X2,X3],
        Vars in 4..6,
        X1#<X2,
        X2#<X3,
        label(Vars).
```