USTHB - FEI - Département d'Informatique
Master 2 ''Systèmes Informatiques Intelligents'' 2018/2019
Module ''Programmation par Contraintes''
Date : 16/12/2018 <9h40-11h10>

Interrogation
-Corrigé-

Exercice 1 (10 points)

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

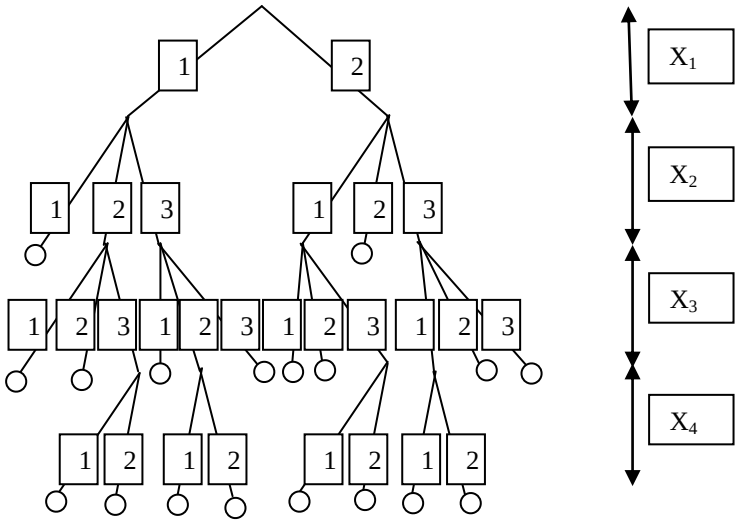
- $X=\{X_1,X_2,X_3,X_4\}$
- $D(X_1)=D(X_4)=\{1,2\}$, $D(X_2)=D(X_3)=\{1,2,3\}$
- $C=\{c_1 : X_1\neq X_2, c_2 : X_1\neq X_3, c_3 : X_1\neq X_4, c_4 : X_2\neq X_3, c_5 : X_2\neq X_4, c_6 : X_3\neq X_4\}$

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) Donnez la représentation graphique de P
4) Donnez la représentation matricielle de P
5) Appliquez l'algorithme de consistance d'arc AC3 au CSP P
6) Appliquez au CSP P l'algorithme de recherche Look-Ahead avec comme procédure de filtrage l'algorithme de consistance de chemin PC2

Pour chacun des trois algorithmes de recherche, il est demandé de respecter l'ordre statique X_1, X_2, X_3, X_4 d'instanciation des variables ; et l'ordre statique 1, 2, 3 de choix des valeurs du domaine de la variable en cours d'instanciation. Il est également demandé d'expliquer tous les échecs éventuels.

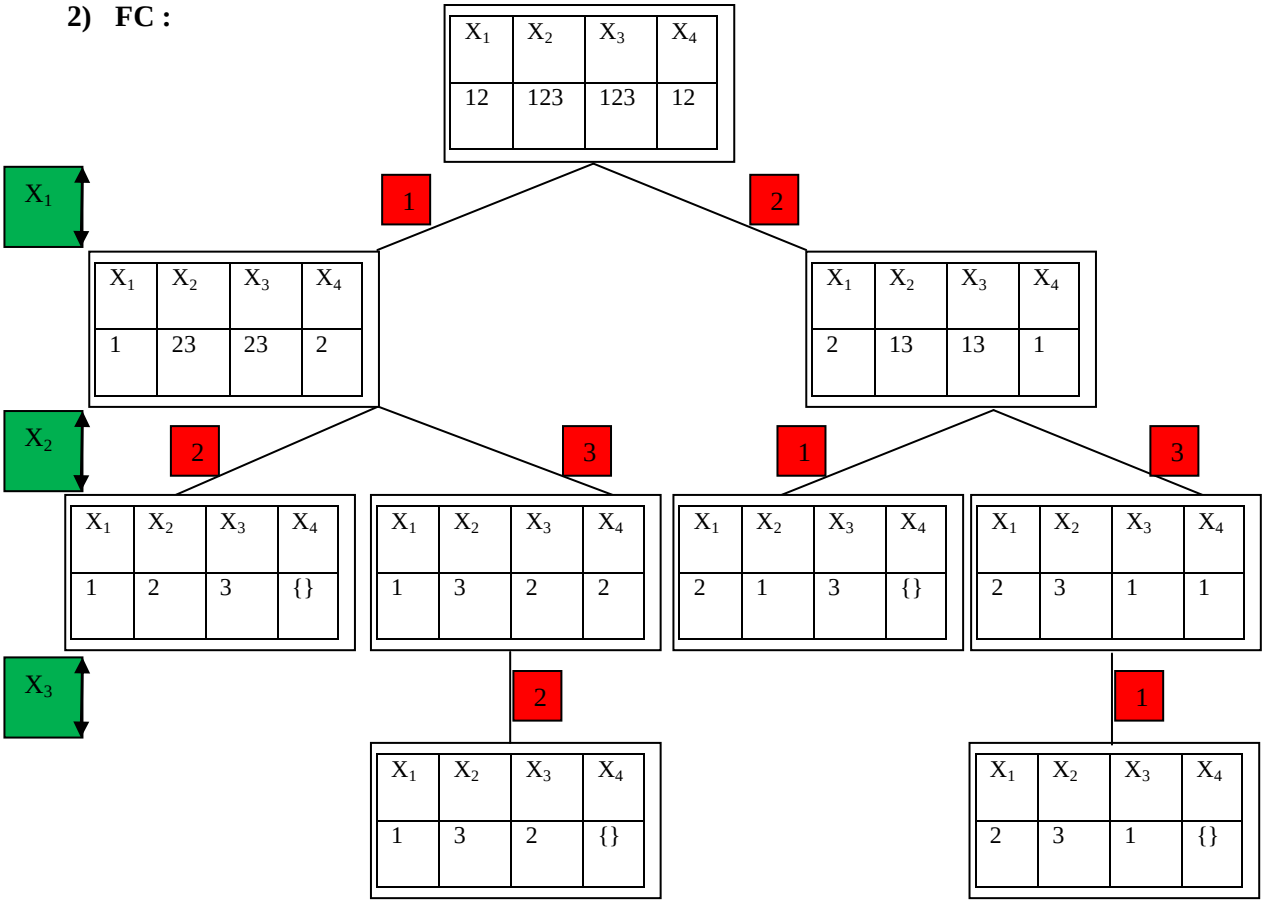
Solution :

1) SRA :

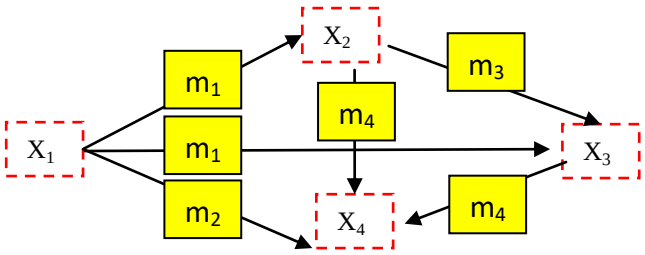


Les 18 feuilles de l'arbre de recherche correspondent aux échecs de l'algorithme SRA, pour lesquels nous donnons ci-dessous, de gauche à droite, respectivement, une contrainte non satisfaite entre la variable en cours d'instanciation et une variable déjà instanciée : c_1, c_2, c_4, c_3, c_5, c_2, c_3, c_6, c_4, c_4, c_2, c_5, c_3, c_1, c_6, c_3, c_2, c_4.

2) FC :



3) Représentation graphique : Gp=(X,E,I)



Avec $m_1=\begin{bmatrix}0 & 1 & 1\\ 1 & 0 & 1\end{bmatrix}$, $m_2=\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$, $m_3=\begin{bmatrix}0 & 1 & 1\\ 1 & 0 & 1\\ 1 & 1 & 0\end{bmatrix}$ et $m_4=\begin{bmatrix}0 & 1\\ 1 & 0\\ 1 & 1\end{bmatrix}$

4) Représentation matricielle M_P :

	X_1	X_2	X_3	X_4
X_1	I_2	m_1	m_1	m_2
X_2	m_4	I_3	m_3	m_4
X_3	m_4	m_3	I_3	m_4
X_4	m_2	m_1	m_1	I_2

Avec $I_2=\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$ et $I_3=\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}$

5) AC3 initialisera sa file Q à tous les arcs (X_i,X_j) sur lesquels il y a une contrainte, dans les deux sens : Q={(X_1,X_2), (X_2,X_1), (X_1,X_3), $(X_3,X1)$, (X_1,X_4), (X_4,X_1), (X_2,X_3), (X_3,X_2), (X_2,X_4), (X_4,X_2), (X_3,X_4), (X_4,X_3)}. Toutes les paires (X_i,X_j) de Q vérifient ce qui suit : toute valeur V_i de $D(X_i)$ a un support dans $D(X_j)$. Donc AC3 n'aura aucun effet sur P : il ne fera aucune modification. Le CSP, tel qu'il est donné, est déjà consistant d'arc.

6) Look Ahead commencera par un prétraitement par PC2, qui détectera l'inconsistance de P. Ceci peut être justifié par les trois opérations de consistance de chemin suivantes, qui transforment l'entrée [2,3] de M_P en matrice nulle :

- $M_P[2,4]=M_P[2,4]\cap M_P[2,1]\circ M_P[1,1]\circ M_P[1,4]=m_4\cap m_4\circ m_2$

$$=\begin{bmatrix}0 & 1\\ 1 & 0\\ 1 & 1\end{bmatrix}\cap\begin{bmatrix}0 & 1\\ 1 & 0\\ 1 & 1\end{bmatrix}\circ\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}=\begin{bmatrix}0 & 1\\ 1 & 0\\ 1 & 1\end{bmatrix}\cap\begin{bmatrix}1 & 0\\ 0 & 1\\ 1 & 1\end{bmatrix}=\begin{bmatrix}0 & 0\\ 0 & 0\\ 1 & 1\end{bmatrix}=m_5$$

- $M_P[4,3]=M_P[4,3]\cap M_P[4,1]\circ M_P[1,1]\circ M_P[1,3]=m_1\cap m_2\circ m_1$

$$=\begin{bmatrix}0 & 1 & 1\\ 1 & 0 & 1\end{bmatrix}\cap\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}\circ\begin{bmatrix}0 & 1 & 1\\ 1 & 0 & 1\end{bmatrix}=\begin{bmatrix}0 & 1 & 1\\ 1 & 0 & 1\end{bmatrix}\cap\begin{bmatrix}1 & 0 & 1\\ 0 & 1 & 1\end{bmatrix}=\begin{bmatrix}0 & 0 & 1\\ 0 & 0 & 1\end{bmatrix}=m_6$$

- $M_P[2,3]=M_P[2,3]\cap M_P[2,4]\circ M_P[4,4]\circ M_P[4,3]=m_3\cap m_5\circ m_6$

$$=\begin{bmatrix}0 & 1 & 1\\ 1 & 0 & 1\\ 1 & 1 & 0\end{bmatrix}\cap\begin{bmatrix}0 & 0\\ 0 & 0\\ 1 & 1\end{bmatrix}\circ\begin{bmatrix}0 & 0 & 1\\ 0 & 0 & 1\end{bmatrix}=\begin{bmatrix}0 & 1 & 1\\ 1 & 0 & 1\\ 1 & 1 & 0\end{bmatrix}\cap\begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 1\end{bmatrix}=\begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}$$

Exercice 2 (4 points)

On considère l'instance de la table ci-dessous du problème d'ordonnancement de type job shop, qui consiste en deux jobs J1 et J2 devant passer chacun par deux machines M1 et M2 :

	1ère tâche : <machine, durée>	2ème tâche : <machine, durée>
Job J1	<M1,5>	<M2,3>
Job J2	<M2,4>	<M1,6>

Toutes les tâches sont non-préemptives. La date de début au plus tôt est t_d=2. La date de fin au plus tard est t_f=25.

On s'intéresse au problème de recherche d'une solution réalisable, c'est-à-dire satisfaisant toutes les contraintes mais ne donnant pas forcément l'optimum de l'instance.

1) Modélisez le problème à l'aide d'un CSP binaire discret P=(X,D,C)
2) Donnez la représentation graphique de P
3) Donnez la représentation matricielle de P

Solution :

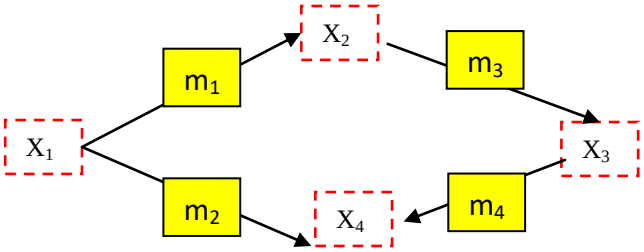
1) **Modélisation** à l'aide d'un CSP binaire discret P=(X,D,C) **:**

L'instance du problème de job shop est constituée de quatre tâches : T_1 (passage du job J1 par la machine M1), T_2 (passage du job J1 par la machine M2), T_3 (passage du job J2 par la machine M2), et T_4 (passage du job J2 par la machine M1). L'instance peut être modélisée par le CSP P=(X,D,C) suivant :

- X={ X_1,X_2,X_3,X_4}, la variable X_i (i de 1 à 4) désignant la date de début de la tâche T_i
- $D(X_1)$={2,…,20}, $D(X_2)$={2,…,22}, $D(X_3)$={2,…,21}, $D(X_4)$={2,…,19}

- C={c_1 : $(X_2-X_1)\in[5,+\infty[$,c_2 : $(X_4-X_3)\in[4,+\infty[$,c_3 : $(X_4-X_1)\in]-\infty,-6]\cup[5,+\infty[$, c_4 : $(X_3-X_2)\in]-\infty,-4]\cup[3,+\infty[$}

2) **Représentation graphique :** Gp=(X,E,I)



Avec $m_1=\begin{bmatrix}0000011111111111111\\0000001111111111111\\0000000111111111111\\0000000011111111111\\0000000001111111111\\0000000000111111111\\0000000000011111111\\0000000000001111111\\0000000000000111111\\0000000000000011111\\0000000000000001111\\0000000000000000111\\0000000000000000011\\0000000000000000001\\0000000000000000000\\0000000000000000000\\0000000000000000000\end{bmatrix}$, $m_2=\begin{bmatrix}0000011111111111111\\0000001111111111111\\0000000111111111111\\0000000011111111111\\0000000001111111111\\0000000000111111111\\0000000000011111111\\1000000000001111111\\1100000000000111111\\1110000000000011111\\1111000000000001111\\1111100000000000111\\1111110000000000011\\1111111000000000001\\1111111100000000000\\1111111111000000000\\1111111111100000000\\1111111111110000000\\1111111111111000000\\1111111111111100000\end{bmatrix}$, $m_3=\begin{bmatrix}0001111111111111111\\0000111111111111111\\0000011111111111111\\0000001111111111111\\1000000111111111111\\1100000011111111111\\1110000001111111111\\1111000000111111111\\1111100000011111111\\1111110000001111111\\1111111000000111111\\1111111100000011111\\1111111110000001111\\1111111111000000111\\1111111111100000011\\1111111111110000001\\1111111111111000000\\1111111111111100000\\1111111111111110000\\1111111111111111000\end{bmatrix}$ **et**

$m_4=\begin{bmatrix}0000111111111111111\\0000011111111111111\\0000001111111111111\\0000000111111111111\\0000000011111111111\\0000000001111111111\\0000000000111111111\\0000000000011111111\\0000000000001111111\\0000000000000111111\\0000000000000011111\\0000000000000001111\\0000000000000000111\\0000000000000000011\\0000000000000000001\\0000000000000000000\\0000000000000000000\\0000000000000000000\\0000000000000000000\\0000000000000000000\end{bmatrix}$

3) **Représentation matricielle :**

	X_1	X_2	X_3	X_4
X_1	I_{19}	m_1	U_{19x20}	m_2
X_2	m_1^t	I_{21}	m_3	U_{21x18}
X_3	U_{20x19}	m_3^t	I_{20}	m_4
X_4	m_2^t	U_{18x21}	m_4^t	I_{18}

Exercice 3 (6 points) :

On représente une matrice mxn par une liste de m listes de n éléments chacune, la $i^{ème}$ liste, $i \in \{1,..,m\}$, représentant la $i^{ème}$ ligne de la matrice, le $j^{ème}$ élément d'une liste étant le $j^{ème}$ élément de la ligne représentée. Donnez un programme Prolog calculant la somme de deux matrices d'entiers ainsi que la transposée d'une matrice.

Solution :

```
somme([],[],[]):-!.
somme([[]|L1],[[]|L2],[[]|L3]):-!,somme(L1,L2,L3).
somme([[X|L1]|L2],[[Y|L3]|L4],[[Z|L5]|L6]):-Z is X+Y,
        somme([L1|L2],[L3|L4],[L5|L6]).
transposee([],[]):-!.
transposee(L,[]):-liste_de_listes_vides(L),!.
transposee(L1,[L2|L3]):-partitionner(L1,L2,M),
        transposee(M,L3).
liste_de_listes_vides([]):-!.
liste_de_listes_vides([[]|L]):-liste_de_listes_vides(L).
partitionner([],[],[]):-!.
partitionner([[X|L1]|L2],[X|C],[L1|M]):-partitionner(L2,C,M).
/*
1 ?- somme([[10,20,30],[10,20,30],[10,20,30]],[[1,2,3],[1,2,3],[1,2,3]],L).
L = [[11, 22, 33], [11, 22, 33], [11, 22, 33]].

2 ?- transposee([[10,20,30],[10,20,30],[10,20,30]],L).
L = [[10, 10, 10], [20, 20, 20], [30, 30, 30]].

3 ?-
*/
```