Université des Sciences et de la Technologie Houari Boumediène
Faculté d'Electronique et d'Informatique
Département d'Informatique
LMD Master 2ème Année ''Systèmes Informatiques Intelligents'' 2010/2011
Module ''Programmation par Contraintes''
Date : 18/01/2011

Corrigé de l'examen

Exercice 1 <5 points>:
On considère le CSP discret binaire P=(X,D,C) suivant :

- $X = \{X_1, X_2, X_3\}$
- $D(X_1)=D(X_2)=D(X_3)=\{(0,1),(2,1),(0,4),(2,4)\}$
- $C = \{c_1, c_2, c_3\}$ avec
  - c_1 : oblique(X_1, X_2)
  - c_2 : oblique(X_1, X_3)
  - c_3 : oblique(X_2, X_3)

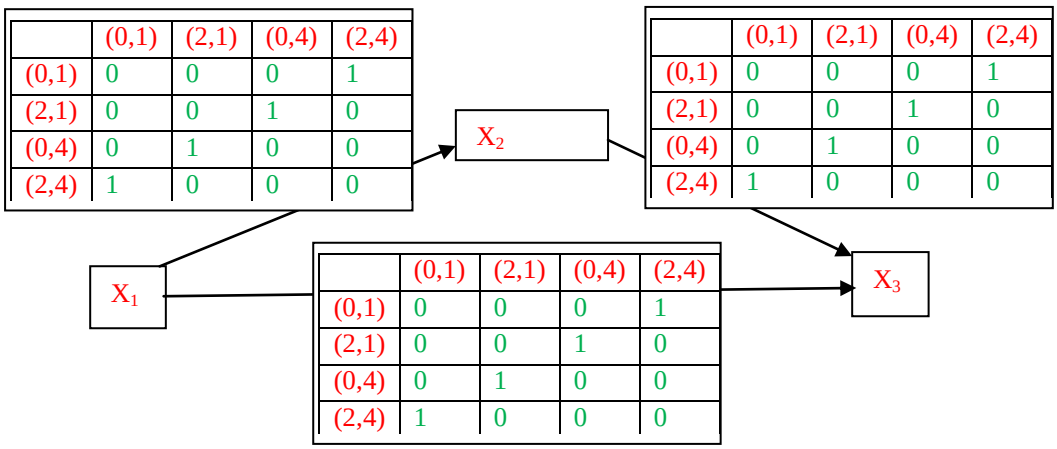
Pour tous points A et B du plan, donnés par leurs coordonnées (a_1,b_1) et (a_2,b_2), respectivement :

oblique(A,B) si et seulement si ($a_1 \neq a_2$ et $b_1 \neq b_2$)

1) Donnez une représentation graphique de P.
2) Donnez une représentation matricielle de P.
3) Appliquez l'algorithme de consistance d'arc AC3 au CSP P.

Solution :

1) Une représentation graphique de P :



2) Une représentation matricielle M_P de P :

	X_1	X_2	X_3
X_1	$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$	$\begin{bmatrix}0&0&0&1\\0&0&1&0\\0&1&0&0\\1&0&0&0\end{bmatrix}$	$\begin{bmatrix}0&0&0&1\\0&0&1&0\\0&1&0&0\\1&0&0&0\end{bmatrix}$
X_2	$\begin{bmatrix}0&0&0&1\\0&0&1&0\\0&1&0&0\\1&0&0&0\end{bmatrix}$	$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$	$\begin{bmatrix}0&0&0&1\\0&0&1&0\\0&1&0&0\\1&0&0&0\end{bmatrix}$
X_3	$\begin{bmatrix}0&0&0&1\\0&0&1&0\\0&1&0&0\\1&0&0&0\end{bmatrix}$	$\begin{bmatrix}0&0&0&1\\0&0&1&0\\0&1&0&0\\1&0&0&0\end{bmatrix}$	$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$

3) Application de la consistance d'arc AC3 au CSP P :

L'algorithme utilisera une file F qu'il initialisera aux arcs sur lesquels il y a une contrainte : $F=\{(X_1,X_2),(X_2,X_1),(X_1,X_3),(X_3,X_1),(X_2,X_3),(X_3,X_2)\}$. L'algorithme prendra à tour de rôle les différents arcs de la file pour propagation. Mais la propagation ne modifiera aucun domaine pour la simple raison que le CSP est déjà consistant d'arc, qui est très visible au niveau des éléments de la représentation matricielle M_P : ces éléments sont des matrices carrées booléennes 4x4 et chacune de ces matrices booléennes présente sur chacune de ses lignes (au moins) un 1 !

Exercice 2 <5 points>:

Soit G=(V,E) un graphe : V est l'ensemble des sommets et E l'ensemble des arêtes.

1) Modélisez à l'aide d'un CSP discret binaire P=(X,D,C) le problème de 3-coloriage de G défini, pour rappel, comme suit : peut-on colorier les sommets de G avec trois couleurs de telle sorte que deux sommets adjacents n'aient pas la même couleur ?
2) Comment passer du graphe G à la représentation graphique de P ? Expliquez.

Solution :

1) G=(V,E), avec $V=\{V_1,\ldots,V_i,\ldots,V_n\}$. Le CSP P=(X,D,C) modélisant le problème de 3-coloriage de G est comme suit :
   - $X=\{X_1,\ldots,X_i,\ldots,X_n\}$ (les variables de P sont les sommets de G, mais on renomme ici et on appelle les variables $X_1,\ldots,X_n$ pour rester en conformité avec les notations du cours : la variable X_i correspond donc au sommet V_i)
   - Le domaine est commun à toutes les variables, et est donné par l'ensemble couleurs=$\{cl_1,cl_2,cl_3\}$ des trois couleurs à utiliser pour le coloriage de G : $D(X_i)$=couleurs, pour tout i=1,…,n
   - L'ensemble C des contraintes de P est construit comme suit :
     i. On initialise C à l'ensemble vide : $C=\emptyset$
     ii. Pour toute arête (V_i,V_j) de G, on ajoute à C la contrainte $X_i \neq X_j$: $C=C\cup\{X_i\neq X_j\}$
2) Le passage du graphe G à la représentation graphique de P est quasi-automatique, et se fait comme suit :
   - Transformer chaque arête (V_i,V_j) (non orientée) en l'arc (V_i,V_j) ou (exclusif) en l'arc (V_j,V_i)
   - Pour tout i, renommer le sommet V_i en X_i
   - Etiqueter chacun des arcs du graphe orienté ainsi obtenu par la matrice carrée 3x3 booléenne suivante :

	Cl_1	Cl_2	Cl_3
Cl_1	0	1	1
Cl_2	1	0	1
Cl_3	1	1	0

Exercice 3 <2,5 points>:

Expliquez le principe de fonctionnement de l'algorithme de consistance de chemin PC3 appliqué à un CSP qualitatif de directions cardinales.

Solution :

Soit P=(X,C) un CSP qualitatif de directions cardinales. L'algorithme de consistance de chemin PC3 appliqué à P utilise une file F qu'il initialise à toutes les paires (X_i,X_j) de variables sur lesquelles il y a une contrainte. Si la file contient une paire (X_i,X_j), elle ne contient pas la paire (X_j,X_i) ; on peut supposer que les paires (X_i,X_j) qu'elle contient vérifie i<j. L'algorithme répète les étapes suivantes jusqu'à ce que la file devienne vide, ou qu'il détecte la relation vide indiquant l'inconsistance de P :

- Il prend une paire (X_i,X_j) de la file pour propagation

- Pour toute variable X_k, avec $k \notin \{i,j\}$, l'algorithme révise éventuellement les étiquettes des arcs (X_i,X_k) et (X_j,X_k) ; et si l'étiquette d'un des arcs est modifiée, l'arc est mis dans la file s'il n'y est pas déjà :
    - temp=C[i,k]
    - $C[i,k]=C[i,k] \cap (C[i,j] \circ C[j,k])$
    - si(temp≠C[i,k]){

      $C[k,i]=(C[i,k])^{-1}$
      $F=F \cup \{(X_i,X_k)\}$
      }
    - temp=C[j,k]
    - $C[j,k]=C[j,k] \cap (C[j,i] \circ C[i,k])$
    - si(temp≠C[j,k]){

      $C[k,j]=(C[j,k])^{-1}$
      $F=F \cup \{(X_j,X_k)\}$
      }

L'algorithme détaillé est comme suit (voir cours) :

fonction REVISE((Xi,X_k,Xj),(X,D,C)) : booléen
début
    temp=C_{ik}
    $C_{ik}=C_{ik} \cap C_{ij} \circ C_{jk}$
    si temp=C_{ik} alors retourner FAUX
    sinon{
        $C_{ki}=(C_{ik})^{-1}$
        retourner VRAI
        }
    finsi
fin

fonction PC3((X,C))
début
Q ← {(X_i,X_j) : (i<j) et il existe une contrainte entre X_i et X_j}
tantque Q≠∅ faire
    Prendre une paire (Xi,Xj) de variables de Q
    supprimer la paire de Q : Q ← Q\{(X_i,X_j)}
    pour toute variable X_k différente de X_i et de X_j faire
        si REVISE((X_i,X_k,X_j),(X,C)) alors Q←Q∪{(X_k,X_i)}
        finsi
        si REVISE((X_j,X_k,X_i),(X,C)) alors Q←Q∪{(X_k,X_i)}
        finsi
    fait
fin-tantque
retourner (X,C)
fin

Exercice 4 <5 points>:

On considère le problème d'ordonnancement de quatre tâches T_1, T_2, T_3 et T_4 suivant :

a. T_1 et T_2 sont des tâches d'un même job J_1 et doivent être exécutées dans l'ordre T_1, T_2
b. T_3 et T_4 sont des tâches d'un même job J_2 et doivent être exécutées dans l'ordre T_3, T_4
c. Les tâches T_1 et T_3 doivent être exécutées sur une même machine m_1
d. Les tâches T_2 et T_4 doivent être exécutées sur une même machine m_2

e. Les durées des tâches T_1, T_2, T_3 et T_4 sont, respectivement 5, 6, 7 et 2

f. Toutes les tâches sont non-préemptives

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

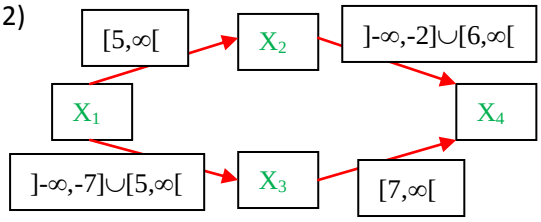
1) Modélisez le problème à l'aide d'un TCSP P=(X,C).
2) Donnez une représentation graphique de P.
3) Comment peut-on modifier l'algorithme Look_Ahead de telle sorte qu'il fournisse, pour un TCSP modélisant un problème d'ordonnancement, une solution réalisant l'optimum ? A quelle valeur doit-on initialiser l'optimum ? Expliquez.

Solution :

1) Le CSP P aura quatre variables X_1, X_2, X_3 et X_4; X_i, i=1…4, désignera le début de la tâche T_i. Les contraintes sont construites comme suit :
   - Les points a., e. et f. donnent la contrainte c_1 : $(X_2-X_1)\in[5,\infty[$
   - Les points b., e. et f. donnent la contrainte c_2 : $(X_4-X_3)\in[7,\infty[$
   - Les points c., e. et f. donnent la contrainte c_3 : $(X_3-X_1)\in]-\infty,-7]\cup[5,\infty[$
   - Les points d., e. et f. donnent la contrainte c_4 : $(X_4-X_2)\in]-\infty,-2]\cup[6,\infty[$

   Conclusion : $P=\{X_1,X_2,X_3,X_4\}$ et $C=\{c_1,c_2,c_3,c_4\}$

2)



3) L'algorithme Look_Ahead doit être modifié de telle sorte qu'il ne s'arrête pas à la première rencontre d'une solution réalisable. Tant que l'espace de recherche n'est pas entièrement exploré, il doit, à chaque fois qu'il trouve une nouvelle solution réalisable, comparer la valeur courante de l'optimum à l'optimum du raffinement convexe ayant donné lieu à la nouvelle solution : l'optimum est le maximum des $a_{0i}+d_i$, i=1…n ; a_{0i} est la borne inférieure de l'étiquette de l'arc (X_0,X_i) dans le raffinement, et d_i la durée de la tâche T_i. Il faut noter ici que l'algorithme Look_Ahead doit rajouter la variable ''origine du monde'' X_o qu'il reliera à chacune des autres variables, X_i, avec un arc étiqueté par [0,∞[(toutes les tâches doivent commencer au plus tôt à la même date que l'origine du monde). L'optimum doit être initialisé à la somme des durées des tâches (ici 5+6+7+2=20) : la solution réalisant une telle valeur de l'optimum est celle mettant côte à côte les tâches, sans laisser de vide entre deux tâches consécutives, et tout en respectant les contraintes de précédence (conjonctives) entre les tâches d'un même job !

Exercice 5 <2,5 points>:

Ecrivez un programme Prolog testant si une liste est palindrome. Vous supposerez que la liste vide est palindrome. Le programme ne doit pas faire appel à la bibliothèque Prolog.

Solution:

```
palindrome([]).
palindrome([X]).
palindrome([X|L]) :-inverse(L,R),concatener((R,[X],[X|L]).
inverse([],[]).
inverse([X],[X]).
inverse([X|L],R) :-inverse(L,S),concatener(S,[X],R).
concatener([],L,L).
concatener([X],L,[X|L]).
```

```
concatener([X|L],R,S):-concatener(L,R,T),concatener([X],T,S).
```