

Programmation par Contraintes

Module du Master "Systèmes Informatiques Intelligents" 2ème année

CHAPITRE III

Résolution d'un CSP binaire discret

Mr ISLI

Faculté d'Informatique

Département Intelligence Artificielle et Science des Données

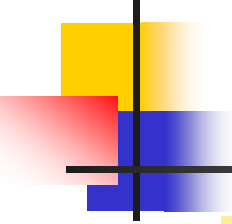
Université des Sciences et de la Technologie Houari Boumediène

BP 32, El-Alia, Bab Ezzouar

DZ-16111 ALGER

https://perso.usthb.dz/~aisli/TA_PpC.htm

amar.isli@usthb.edu.dz



CHAPITRE III

Résolution d'un CSP binaire discret

Résolution d'un CSP binaire

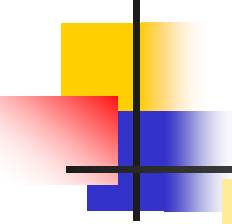
- Le CSP admet-il des solutions ?
- Le CSP admet-il des solutions ? Si oui, en exhiber une
- Le CSP admet-il des solutions ? Si oui, les exhiber toutes

CHAPITRE III

Résolution d'un CSP binaire discret

Algorithmes de résolution

- Algorithmes de consistance locale
 - Propagation de contraintes : incomplets en général mais de complexité polynômiale
 - Consistance de nœud : consistance des sous-CSP de taille 1
 - consistance d'arc : consistance des sous-CSP de taille inférieure ou égale à 2
 - Complets pour certaines classes de CSP
- Algorithmes complets naïfs (recherche systématique d'une solution)
 - Générer et tester (GET)
 - Simple retour arrière (SRA)



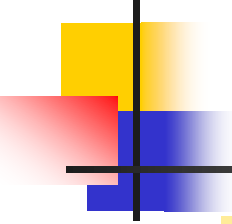
CHAPITRE III

Résolution d'un CSP binaire discret

Algorithmes de résolution

■ Algorithmes complets intelligents

- Quand une nouvelle variable est instanciée, filtrage des domaines des variables non encore instanciées
 - Pour chaque variable non encore instanciée, supprimer de son domaine toutes les valeurs incompatibles avec la valeur venant d'être choisie pour la variable en cours d'instanciation (**forward-checking**) :
 - Anticipe les situations auxquelles fait face SRA : **forward-checking = anticipation**
 - Appliquer un algorithme de consistance d'arc au CSP résultant de la nouvelle instanciation de variable (**look-ahead**) :
 - Forward-checking = forme simplifiée de look-ahead
 - Filtrage au niveau des nœuds de l'arbre de recherche : meilleur que celui du forward-checking (nouveaux domaines=sous-ensembles de ceux que donnerait FC)
 - Mais plus gourmand en temps
 - Compromis entre filtrage et complexité
- Instancier-et-propager (forme de branch and bound)



CHAPITRE III

Résolution d'un CSP binaire discret

Algorithmes de résolution d'un CSP binaire

- Heuristiques
 - Ordre d'instanciation des variables (statique, dynamique)
 - Ordre sur les valeurs du domaine de la variable en cours d'instanciation (statique, dynamique)

CHAPITRE III

Résolution d'un CSP binaire discret

Algorithme « générer et tester » (GET)

1. Initialement, aucune instanciation n'est marquée
2. Considérer une instanciation $i=(e_1, \dots, e_n)$ non marquée
3. Marquer i
4. Si i satisfait toutes les contraintes du CSP :
 - Retourner i
5. Si toutes les instanciations sont marquées
 - Retourner « échec : le CSP n'admet pas de solution »
6. Aller en 2.

CHAPITRE III

Résolution d'un CSP binaire discret

Algorithme GET :

fonction GET(A) : booléen

début

si toutes les variables de X sont instanciées alors

 si A est solution alors retourner VRAI sinon retourner FAUX finsi

sinon

 choisir une variable X_i de X qui n'est pas encore instanciée

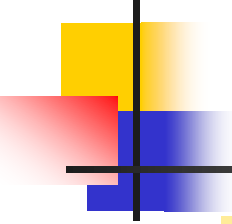
 pour toute valeur V_i du domaine $D(X_i)$ faire

 si GET(AU{(X_i, V_i)}) alors retourner VRAI finsi fait

 retourner FAUX

finsi

fin



CHAPITRE III

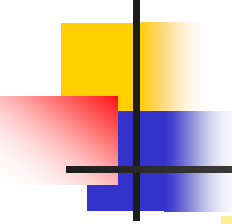
Résolution d'un CSP binaire discret

Algorithme GET : **inconvenient majeur**

- Si inexistence de solution ou unique solution consistant en la toute dernière instantiation
 - Parcours exhaustif de toutes les instantiations possibles : exploration de tout l'espace de recherche $D(X_1) \times \dots \times D(X_n)$.

Algorithme SRA : **amélioration de GET**

- Si la valeur choisie pour la variable en cours d'instanciation est incompatible avec les valeurs choisies pour les variables déjà instanciées → retour arrière
 - Instancier avec une autre valeur du domaine si possible
 - Si toutes les valeurs déjà essayées
 - Retourner échec si toute première variable
 - Retourner à la variable précédente sinon (retour arrière chronologique)
- Sinon :
 - Si toute dernière variable retourner "succès"
 - sinon passer à la variable suivante et réitérer le processus



CHAPITRE III

Résolution d'un CSP binaire discret

Algorithme « Simple Retour Arrière » (SRA)

fonction SRA(A) : booléen

début

choisir une variable X_i non encore instanciée

pour toute valeur V_i du domaine $D(X_i)$ **faire**

si $AU\{(X_i, V_i)\}$ satisfait les contraintes unaires sur X_i et toutes les contraintes binaires sur X_i et les variables déjà instanciées **alors**

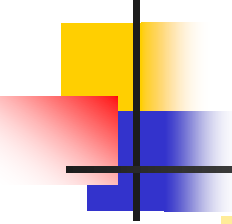
si $AU\{(X_i, V_i)\}$ instanciation complète **alors** retourner VRAI

sinon si SRA($AU\{(X_i, V_i)\}$) **alors** retourner VRAI **finsi finsi finsi**

fait

retourner FAUX

fin



CHAPITRE III

Résolution d'un CSP binaire discret

Exemple : le problème des quatre reines

- Echiquier de 16 cases (4x4)
- 4 lignes L1, L2, L3 et L4
- 4 colonnes C1, C2, C3 et C4
- 4 reines R1, R2, R3 et R4 :
 - La reine R1 se déplace sur la colonne C1
 - La reine R2 se déplace sur la colonne C2
 - La reine R3 se déplace sur la colonne C3
 - La reine R4 se déplace sur la colonne C4

CHAPITRE III

Résolution d'un CSP binaire discret

Exemple : le problème des quatre reines

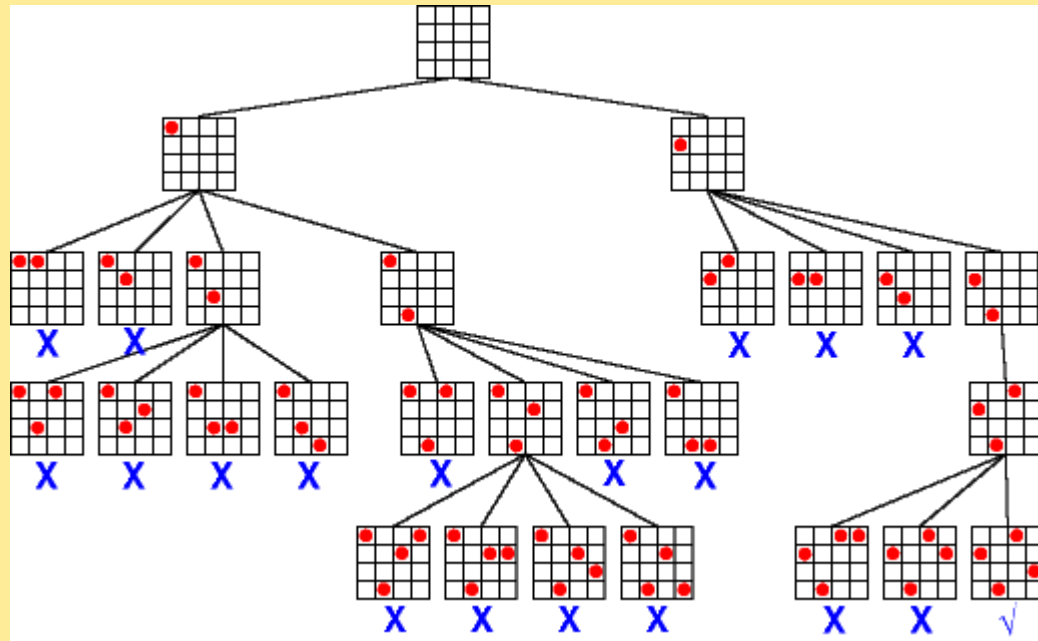
- Modélisation à l'aide d'un CSP $P=(X,D,C)$
 - $X = \{X_1, X_2, X_3, X_4\}$
 - $D = \{D(X_1), D(X_2), D(X_3), D(X_4)\}$ avec
 - $D(X_1)=D(X_2)=D(X_3)=D(X_4)=\{1,2,3,4\}$
 - $C=C_{LI} \cup C_{DD} \cup C_{DA}$
 - $C_{LI}=\{X_i \neq X_j, i \neq j, i \in \{1,2,3,4\}, j \in \{1,2,3,4\}\}$
 - $C_{DA}=\{X_i - i \neq X_j - j, i \neq j, i \in \{1,2,3,4\}, j \in \{1,2,3,4\}\}$:
(i, X_i) et (j, X_j) ne doivent pas être sur la même parallèle à la 1ère bissectrice des axes (diagonale ascendante)
 - $C_{DD}=\{X_i + i \neq X_j + j, i \neq j, i \in \{1,2,3,4\}, j \in \{1,2,3,4\}\}$:
(i, X_i) et (j, X_j) ne doivent pas être sur la même parallèle à la 2ème bissectrice des axes (diagonale descendante)

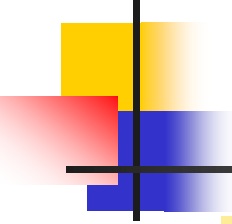
CHAPITRE III

Résolution d'un CSP binaire discret

Résolution du problème des quatre reines

- algorithme SRA





CHAPITRE III

Résolution d'un CSP binaire discret

Algorithme SRA : **inconvenient majeur**

- Améliore l'algorithme GET
 - Réduction de l'espace de recherche : les valeurs affectées à la variable en cours d'instanciation sont réduites à celles satisfaisant les contraintes avec les variables déjà instanciées
- Mais la détection des conflits reste tardive
- **Solution** → **FC** : dès qu'une nouvelle variable est instanciée, supprimer des domaines des variables non encore instanciées les valeurs qui ne sont pas compatibles avec la valeur choisie
→ anticipation

CHAPITRE III

Résolution d'un CSP binaire discret

Algorithme « Forward Checking » (FC)

Sous forme d'une fonction réursive $FC(A, Dom)$

- $P=(X,D,C)$ CSP dont il faut tester la consistance
- A et Dom : instantiation (partielle) + évolution des domaines
- A ensemble dont les éléments sont de la forme (X_i, V_i)
 - contient les valeurs V_i affectées aux variables X_i déjà instanciées
- A l'appel de la fonction FC :
 - A vide (aucune variable n'est instanciée)
 - $Dom=D$
- FC fonction booléenne retournant VRAI ssi P est consistant

CHAPITRE III

Résolution d'un CSP binaire discret

Algorithme « Forward Checking » (FC)

fonction $FC(A, Dom)$: booléen

début

si toutes les variables de X sontinstanciées alors retourner VRAI

sinon

choisir une variable X_i de X qui n'est pas encore instanciée

pour toute valeur V_i de $Dom(X_i)$ satisfaisant les contraintes unaires sur X_i faire

$D' = Dom$ // $D'(X_j) = Dom(X_j)$, pour toute variable X_j

$D'(X_i) = \{V_i\}$; $non_instanciées = \{X_j \in X : X_j \text{ non encore instanciée}\} \setminus \{X_i\}$; $domaine_vide = faux$

tant que ($non_instanciées \neq \emptyset$) et ($domaine_vide = faux$) faire

considérer une variable X_j de l'ensemble $non_instanciées$

$non_instanciées = non_instanciées \setminus \{X_j\}$

$D'(X_j) = \{V_j \in Dom(X_j) \mid \{(X_i, V_i), (X_j, V_j)\} \text{ satisfait les contraintes sur } X_i \text{ et } X_j\}$

si $D'(X_j)$ vide alors $domaine_vide = vrai$ finsi fait

si ($domaine_vide = faux$) alors si $FC(A \cup \{(X_i, V_i)\}, D')$ alors retourner VRAI finsi finsi fait

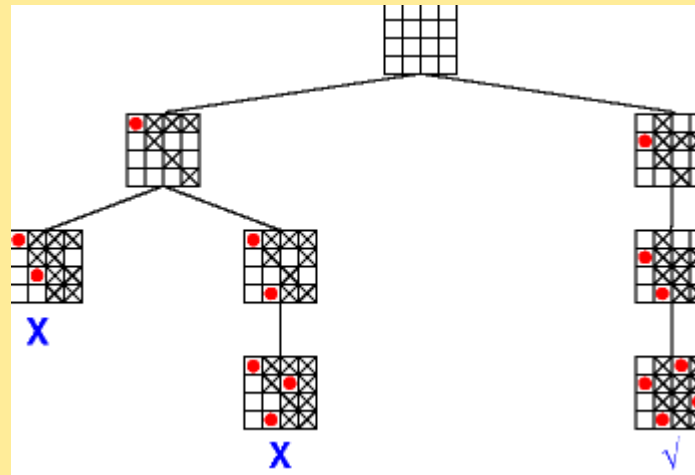
retourner FAUX // la solution partielle A ne peut pas être étendue à la variable X_i

finsi fin

CHAPITRE III

Résolution d'un CSP binaire discret

Exemple :

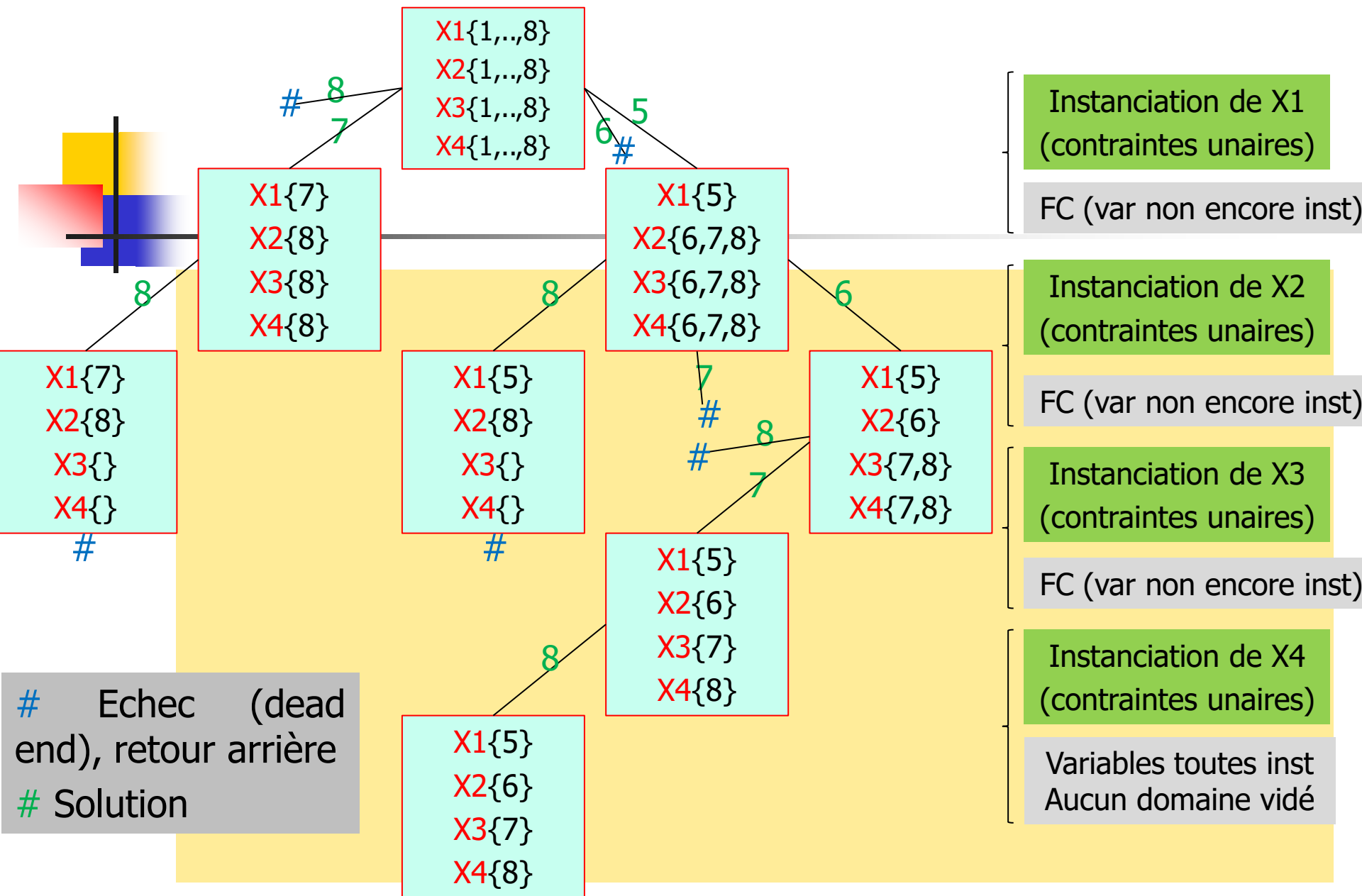


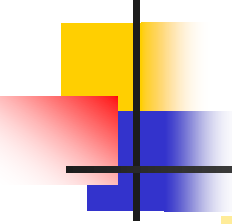
CHAPITRE III

Résolution d'un CSP binaire discret

Exemple : appliquer l'algorithme FC au CSP $P=(X,D,C)$

- $X=\{X_1,X_2,X_3,X_4\}$
- $D(X_1)=D(X_2)=D(X_3)=D(X_4)=\{1,2,3,4,5,6,7,8\}$
- $C=\{c_1 : X_1 \text{ impair} ; c_2 : X_2 \text{ pair} ; c_3 : X_3 \text{ impair} ; c_4 : X_4 \text{ pair} ;$
 $c_5 : X_1 < X_2 ;$
 $c_6 : X_2 < X_4 ;$
 $c_7 : X_1 < X_4 ;$
 $c_8 : X_1 < X_3 ;$
 $c_9 : X_3 < X_4 ;$
 $c_{10} : X_2 < X_3\}$



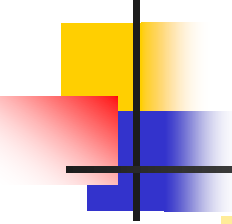


CHAPITRE III

Résolution d'un CSP binaire discret

Algorithme FC : **inconvenient majeur**

- Anticipe des situations d'échec que SRA ne détecte que plus tardivement
- Mais ne supprime que les valeurs incompatibles avec la valeur choisie pour la variable en cours d'instanciation :
 - **Conséquence** : pour X_i et X_j non encore instanciées, $D(X_i)$ peut avoir des valeurs n'ayant plus de support dans $D(X_j)$
- **Solution** → **look-ahead** : filtrage avec un algorithme de consistance d'arc



CHAPITRE III

Résolution d'un CSP binaire discret

Filtrage durant la recherche récursive d'une solution :

- Filtrage à priori
 - Prétraitement
 - Avant le début effectif de la recherche
 - Aucune variable n'est encore instanciée
 - Réduction de l'espace de recherche
 - CSP surcontraint et inconsistant : inconsistance généralement détectée par le prétraitement
- Filtrage après chaque instanciation d'une nouvelle variable
 - Instancier puis propager

CHAPITRE III

Résolution d'un CSP binaire discret

Consistance de nœud (node-consistency)

- Un CSP $P=(X,D,C)$ est consistant de nœud si pour toute variable X_i de X , et pour toute valeur v de $D(X_i)$, l'instanciation partielle $\{(X_i,v)\}$ satisfait toutes les contraintes unaires de C portant sur X_i

Principe d'un algorithme de consistance de nœud

- filtrage des domaines
 - pour chaque variable X_i
 - supprimer de $D(X_i)$ toute valeur v telle que l'instanciation partielle $\{(X_i,v)\}$ viole les contraintes unaires portant exclusivement sur X_i

CHAPITRE III

Résolution d'un CSP binaire discret

fonction NC((X,D,C))

début

pour i=1 à n faire

$D(X_i) = \{v \in D(X_i) : \{(X_i, v)\} \text{ satisfait toutes les contraintes unaires portant sur } X_i\}$

fait

retourner (X,D,C)

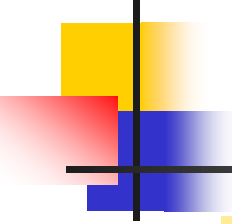
fin

CHAPITRE III

Résolution d'un CSP binaire discret

consistance d'arc (arc-consistency)

- Un CSP $P=(X,D,C)$ est consistant d'arc si :
 - Il est consistant de nœud
 - pour tout couple (X_i, X_j) de variables, et pour toute valeur v_i de $D(X_i)$, il existe une valeur v_j de $D(X_j)$ telle que l'instanciation partielle $\{(X_i, v_i), (X_j, v_j)\}$ satisfait toutes les contraintes binaires de C portant exclusivement sur X_i et X_j



CHAPITRE III

Résolution d'un CSP binaire discret

consistance d'arc (arc-consistency)

- Exemple de CSP consistant d'arc mais inconsistent :
 - CSP modélisant l'instance suivante du problème de coloriage d'un graphe : coloriage d'un triangle avec deux couleurs

CHAPITRE III

Résolution d'un CSP binaire discret

consistance d'arc (arc-consistency)

- Exemple de CSP non consistant d'arc :
 - CSP modélisant l'instance suivante du problème de coloriage d'un graphe : coloriage d'un triangle ABC
 - couleurs permises pour A et B : V et R
 - couleur permise pour C : V

CHAPITRE III

Résolution d'un CSP binaire discret

Principe d'un algorithme de consistance d'arc

- Rendre le CSP consistant de noeud
- Répéter tant que possible
 - Soit X_i une variable telle qu'il existe des valeurs v_i de $D(X_i)$ pour lesquelles:
 - il existe une variable X_j pour laquelle, pour toute valeur v_j de $D(X_j)$
 - l'instanciation partielle $\{(X_i, v_i), (X_j, v_j)\}$ viole les contraintes binaires portant exclusivement sur X_i et X_j
 - supprimer de $D(X_i)$ de telles valeurs v_i



CHAPITRE III

Résolution d'un CSP binaire discret

Algorithmes de consistance d'arc

■ REVISE :

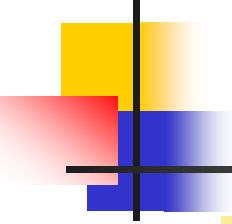
- réduit la taille des domaines
- supprime les valeurs qui ne satisfont pas les contraintes binaires

■ AC1 :

1. appliquer REVISE à tous les arcs sur lesquels il y a une contrainte
 2. Si aucun domaine n'a été modifié, la procédure prend fin
 3. Sinon aller à 1
- **Inconvénient** : réapplique REVISE à tous les arcs sur lesquels il y a une contrainte :
 - Même aux arcs non modifiés par la passe précédente
 - jusqu'à fermeture

■ AC3 :

- ne réapplique REVISE qu'aux arcs dont le domaine de la variable extrémité a été modifié : (X_k, X_i) tel que $D(X_i)$ modifié



CHAPITRE III

Résolution d'un CSP binaire discret

fonction REVISE($(X_i, X_j), (X, D, C)$) : booléen

début

DELETE $\leftarrow$ FAUX

pour tous les V_i de $D(X_i)$ faire

si il n'y a pas de V_j dans $D(X_j)$ telle que $\{(X_i, v_i), (X_j, v_j)\}$ satisfait les contraintes binaires entre X_i et X_j **alors**

supprimer V_i de $D(X_i)$

DELETE $\leftarrow$ VRAI

finsi

fin pour

retourner DELETE

fin

CHAPITRE III

Résolution d'un CSP binaire discret

fonction AC1((X,D,C))

début

(X,D,C)=NC((X,D,C))

$Q \leftarrow \{(X_i, X_j) : (i \neq j) \text{ et (il existe une contrainte entre } X_i \text{ et } X_j)\}$

répéter

$R \leftarrow \text{FALSE}$

pour tous les (X_i, X_j) de Q **faire**

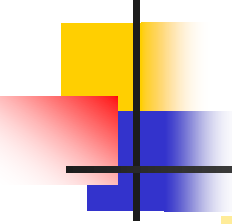
si REVISE($(X_i, X_j), (X, D, C)$) **alors** $R \leftarrow \text{TRUE}$ **fin si**

fin pour

jusqu'à non R

retourner (X,D,C)

fin



CHAPITRE III

Résolution d'un CSP binaire discret

fonction AC3((X,D,C))

début

$(X,D,C) = NC((X,D,C))$

$Q \leftarrow \{(X_i, X_j) : (i \neq j) \text{ et (il existe une contrainte entre } X_i \text{ et } X_j)\}$

tant que $Q \neq \emptyset$ **faire**

Prendre une paire (X_i, X_j) de variables de Q

supprimer la paire de Q : $Q \leftarrow Q \setminus \{(X_i, X_j)\}$

si REVISE($(X_i, X_j), (X,D,C)$) **alors**

$Q \leftarrow Q \cup \{(X_k, X_i) : \text{il existe une contrainte entre } X_k \text{ et } X_i \text{ et } X_k \neq X_i \text{ et } X_k \neq X_j\}$

finsi

fait

retourner (X,D,C)

fin

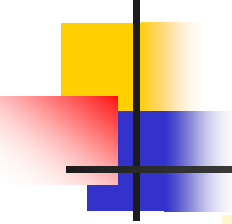
CHAPITRE III

Résolution d'un CSP binaire discret

Exemple 1 : appliquer les algorithmes de consistance d'arc AC1 et AC3 au CSP $P=(X,D,C)$

- $X=\{X_1,X_2,X_3,X_4\}$
- $D(X_1)=D(X_2)=D(X_3)=D(X_4)=\{1,2,3,4,5,6,7,8\}$
- $C=\{c_1 : X_1 \text{ impair} ; c_2 : X_2 \text{ pair} ; c_3 : X_3 \text{ impair} ; c_4 : X_4 \text{ pair} ;$
 $c_5 : X_1 < X_2 ;$
 $c_6 : X_2 < X_4 ;$
 $c_7 : X_1 < X_4 ;$
 $c_8 : X_1 < X_3 ;$
 $c_9 : X_3 < X_4 ;$
 $c_{10} : X_2 < X_3\}$

→ Voir Annexe 1



CHAPITRE III

Résolution d'un CSP binaire discret

fonction `Look_Ahead(A,(X,D,C))` : booléen

Début

`(X,D,C)=AC3((X,D,C))`

si inconsistance (vacuité d'un domaine) **alors** retourner FAUX **finsi**

si tous les domaines sont des singletons **alors** retourner VRAI

sinon

choisir une variable disjonctive X_i de X qui n'est pas encore instanciée

pour $j=1$ à n **faire** $D'(X_j)=D(X_j)$ **fait**

pour toute valeur V_i de $D(X_i)$ **faire**

$D'(X_i)=\{V_i\}$

si `Look_Ahead(AU{(Xi,Vi)},(X,D',C))` **alors** retourner VRAI **finsi**

fait

retourner FAUX //l'instanciation partielle A ne peut pas être étendue à la variable X_i

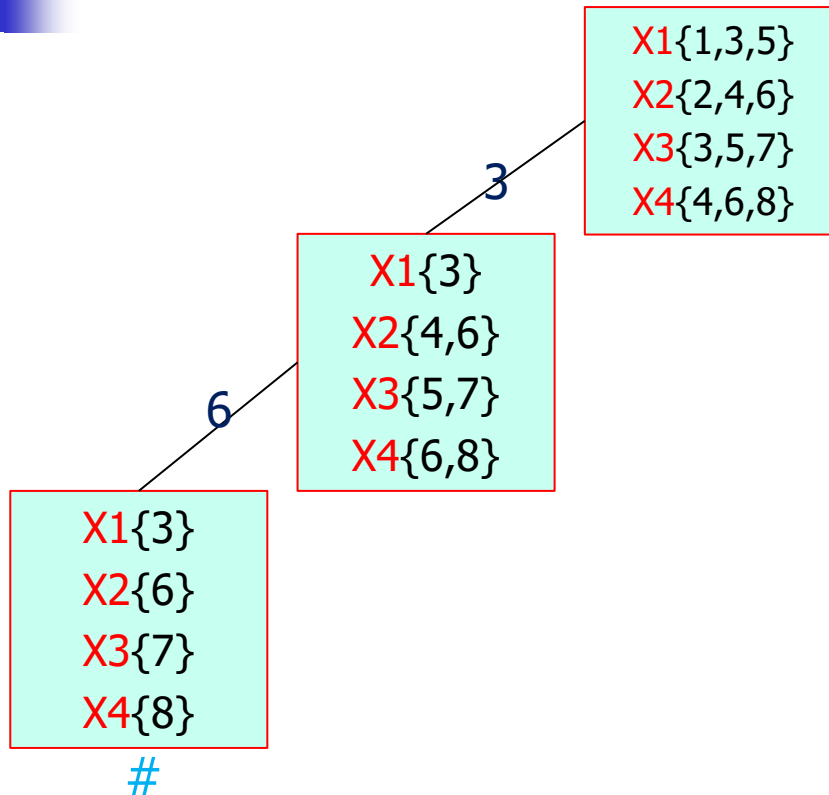
finsi

fin

CHAPITRE III

Résolution d'un CSP binaire discret

Prétraitement (AC3)



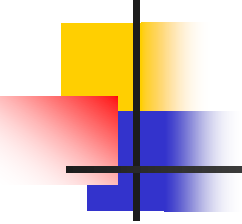
Instanciation de X1

Filtrage (AC3) :
 $Q = \{(X2, X1), (X3, X1), (X4, X1)\}$

Instanciation de X2

Filtrage (AC3) :
 $Q = \{(X1, X2), (X3, X2), (X4, X2)\}$

Raffinement singleton consistant d'arc donc consistant (solution)



CHAPITRE III

Résolution d'un CSP binaire discret

- **K-consistance et k-consistance forte :**

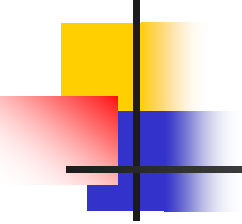
Soit $P=(X,D,C)$ un CSP de taille n , et k un entier entre 1 et n

- **K-consistance**

P est k -consistant si toute solution de tout sous-réseau de taille $k-1$ s'étend à toute $k^{\text{ème}}$ variable

- **K-consistance forte**

P est fortement k -consistant s'il est m -consistant pour tout entier m entre 1 et k



CHAPITRE III

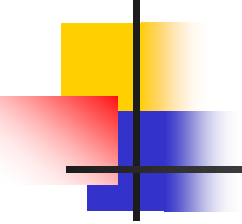
Résolution d'un CSP binaire discret

- **K-consistance et k-consistance forte :**

Soit $P=(X,D,C)$ un CSP de taille n :

- Si P est fortement n -consistant alors toute solution partielle s'étend à une solution globale
- Recherche d'une solution sans retour arrière

- Consistance de noeud = 1-consistance forte
- Consistance d'arc = 2-consistance forte
- Consistance de chemin = 3-consistance forte



CHAPITRE III

Résolution d'un CSP binaire discret

- Minimalité :

Soit $P=(X,D,C)$ un CSP de taille n :

- P est dit minimal si (il est consistant de noeud) et toute solution de tout sous-CSP de taille 2 peut s'étendre à une solution globale

CHAPITRE III

Résolution d'un CSP binaire discret

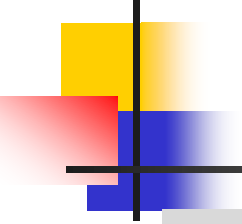
Consistance de chemin (path-consistency)

Un CSP $P=(X,D,C)$ est consistant de chemin si, pour tout chemin $(X_{i0}, \dots, X_{ij})$ de longueur j :

X_{i0}	X_{i1}	...	X_{ik}	$X_{i(k+1)}$	...	X_{ij}
v_0	v_1	...	v_k	$v_{(k+1)}$	...	v_j

Pour tous (v_0, v_j) tels que unaires+binaires sur X_{i0} et X_{ij} satisfaites, il existe $v_1, \dots, v_{(j-1)}$ tels que pour tout $k=0\dots(j-1)$:

$(v_k, v_{(k+1)})$ satisfait unaires+binaires sur X_{ik} et $X_{i(k+1)}$



CHAPITRE III

Résolution d'un CSP binaire discret

consistance de chemin (path-consistency)

On montre facilement qu'on peut se restreindre aux chemins de longueur 2 (triplets de variables).

CHAPITRE III

Résolution d'un CSP binaire discret

consistance de chemin (path-consistency)

Un CSP $P=(X,D,C)$ est consistant de chemin si :

X_{i0}

X_{i1}

X_{i2}

v_0

v_1

v_2

Pour tous (v_0, v_2) tels que unaires+binaires sur X_{i0} et X_{i2} satisfaites, il existe v_1 tel que :

(v_0, v_1) satisfait unaires+binaires sur X_{i0} et X_{i1}

(v_1, v_2) satisfait unaires+binaires sur X_{i1} et X_{i2}

v_1 support de la paire (v_0, v_2)

CHAPITRE III

Résolution d'un CSP binaire discret

Principe d'un algorithme de consistance de chemin

Filtrage des paires de valeurs permises :

- pour chaque paire (X_i, X_j) de variables
 - supprimer les paires permises (v_i, v_j) n'ayant pas de support dans le domaine d'une troisième variable X_k

X_i

X_k

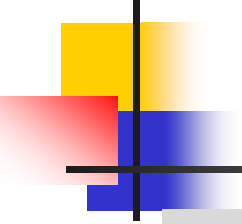
X_j

v_0

v_1

v_2

Supprimer la paire permise (v_0, v_2) si elle n'a aucune valeur v_1 de $D(X_k)$ comme support.



CHAPITRE III

Résolution d'un CSP binaire discret

consistance de chemin (path-consistency)

- Voir Annexe 2-1 pour la présentation des deux algorithmes de consistance de chemin PC-1 et PC-2, et de l'algorithme Look_Ahead avec PC-2 comme procédure de filtrage durant la recherche
- Voir Annexe 2-2 pour le déroulement des algorithmes PC-1 et PC-2