Programmation par contraintes
Cours 4 : Problèmes de Satisfaction de Contraintes
Résolution d'un CSP

Odile PAPINI

POLYTECH
Université d'Aix-Marseille
odile.papini@univ-amu.fr
http://odile.papini.perso.luminy.univ-amu.fr/sources/PCC.html

Plan du cours 4

Bibliographie

Livres :

- **K. Marriott and P. Stuckey. Programming with constraints. MIT Press 1998**
- **F. Fages. Programming Logique par contraintes. Ellipes, 1996**
- **K. R. Apt. Principles in Constraint Programming. Cambridge Univ Press, 2003**

Supports de cours :

- **Support de cours : Christine SOLNON https ://perso.liris.cnrs.fr/christine.solnon/Site-PPC/e-miage-ppc-som.htm**
- **Roman Bartak Charles university : http ://kti.mff.cuni.cz/ bartak/constraints/**

Problèmes de Satisfaction de Contraintes (CSP)

Définition

un CSP est un problème modélisé sous forme de contraintes posées sur des variables prenant valeur dans un domaine

un CSP est quadruplet (X, D, C, R) où :

- $X = \{X_1, \cdots, X_n\}$: un ensemble de variables
- $D = \{D_1, \cdots, D_n\}$: un ensemble de domaines,
  $X_i \in D_i$, $1 \leq i \leq n$
- $C = \{C_1, \cdots, C_m\}$: un ensemble de contraintes,
- $R = \{R_1, \cdots, R_m\}$: un ensemble de relations,
  à chaque contrainte C_j est associée une relation R_j, $1 \leq j \leq m$

Résolution d'un CSP

hypothèse : domaines finis

algorithmes génériques de résolution de CSP

- algorithmes complets
- algorithmes incomplets

Autres algorithmes pour :

- CSP numériques linéaires sur les réels
- CSP numériques linéaires sur les entiers
- CSP numériques non linéaires

Algorithmes de résolution d'un CSP

algorithmes génériques de résolution de CSP

- recherche systématique
  - genere_et_teste (GET)
  - retour_arrière (SRA) ou (backtracking (BT))
- techniques de filtrage
  - consistance de noeud (NC), d'arc (AC), de chemin (PC) $\cdots$
- techniques de propagation de contraintes
  - forward_checking (FC)
  - look_ahead (LH)
- techniques basées sur l'ordre des variables et des valeurs
  - heuristiques

Approche Génère_Et_Teste

principe

- Recherche systématique d'une solution
  - génération d'un affectation totale
  - test de la satisfaction de toutes les contraintes

Algorithme Génère_et_Teste : GET

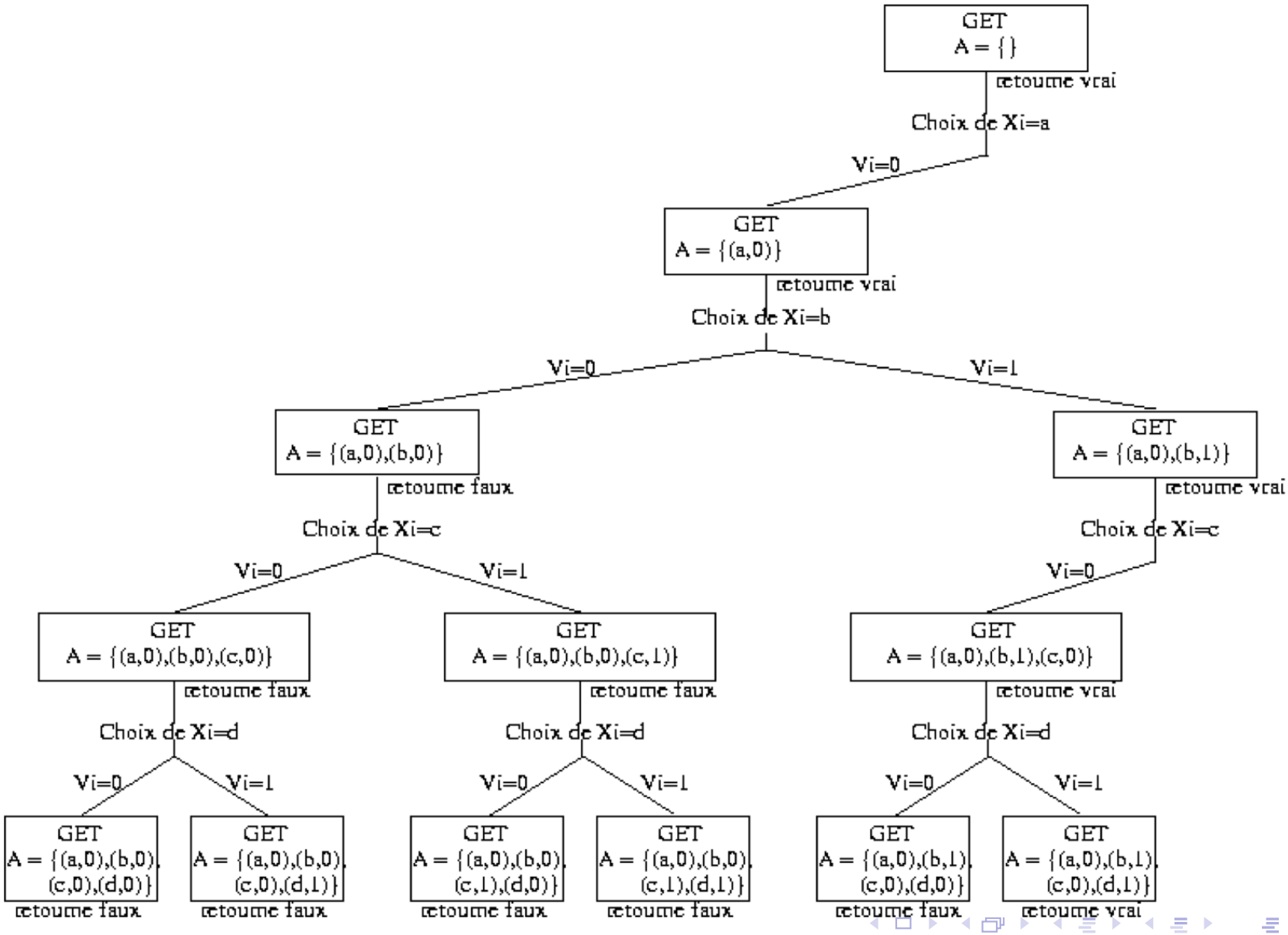
fonction GET(A,(X,D,C)) : booléen
début
si toutes les variables de X sont affectées **alors**
  si A est consistante **alors**
    retourner VRAI
  sinon
    retourner FAUX
  finsi
sinon
  choisir une variable X_i de X qui n'est pas encore affectée
  pour toute valeur V_i appartenant à D_i **faire**
    si GET(A $\cup \{(X_i, V_i)\}$, (X,D,C)) = VRAI **alors**
      retourner VRAI
    finsi
  fin pour
  retourner FAUX
finsi
fin

Algorithme Génère_Et_Teste : GET

exemple

- $X = \{X_1, X_2, X_3, X_4\}$
- $D = \{D_1, D_2, D_3, D_4\}$ avec $D_1 = D_2 = D_3 = D_4 = \{0, 1\}$
- $C = \{X_1 \neq X_2, X_3 \neq X_4, X_1 + X_3 < X_2\}$
- $R = \{R_1, R_2, R_3\}$

Recherche de solutions de ce CSP par l'algorithme GET


GET
A = {}
retourne vrai
Choix de Xi=a
Vi=0
GET
A = {(a,0)}
retourne vrai
Choix de Xi=b
Vi=0
Vi=1
GET
A = {(a,0),(b,0)}
retourne faux
Choix de Xi=c
GET
A = {(a,0),(b,1)}
retourne vrai
Choix de Xi=c
Vi=0
Vi=1
Vi=0
GET
A = {(a,0),(b,0),(c,0)}
retourne faux
Choix de Xi=d
GET
A = {(a,0),(b,0),(c,1)}
retourne faux
Choix de Xi=d
GET
A = {(a,0),(b,1),(c,0)}
retourne vrai
Choix de Xi=d
Vi=0
Vi=1
Vi=0
Vi=1
Vi=0
Vi=1
GET
A = {(a,0),(b,0),(c,0),(d,0)}
retourne faux
GET
A = {(a,0),(b,0),(c,0),(d,1)}
retourne faux
GET
A = {(a,0),(b,0),(c,1),(d,0)}
retourne faux
GET
A = {(a,0),(b,0),(c,1),(d,1)}
retourne faux
GET
A = {(a,0),(b,1),(c,0),(d,0)}
retourne faux
GET
A = {(a,0),(b,1),(c,0),(d,1)}
retourne vrai

Algorithme Génère_Et_Teste : GET

inconvenients

sur l'espace de recherche

- ensemble des affectations complètes
- inconsistance découverte au dernier moment
- croissance exponentielle de la taille de l'espace de recherche

améliorations

- ne développer que des affectations partielles consistantes
- réduire la taille des domaines

Approche Simple retour arrière

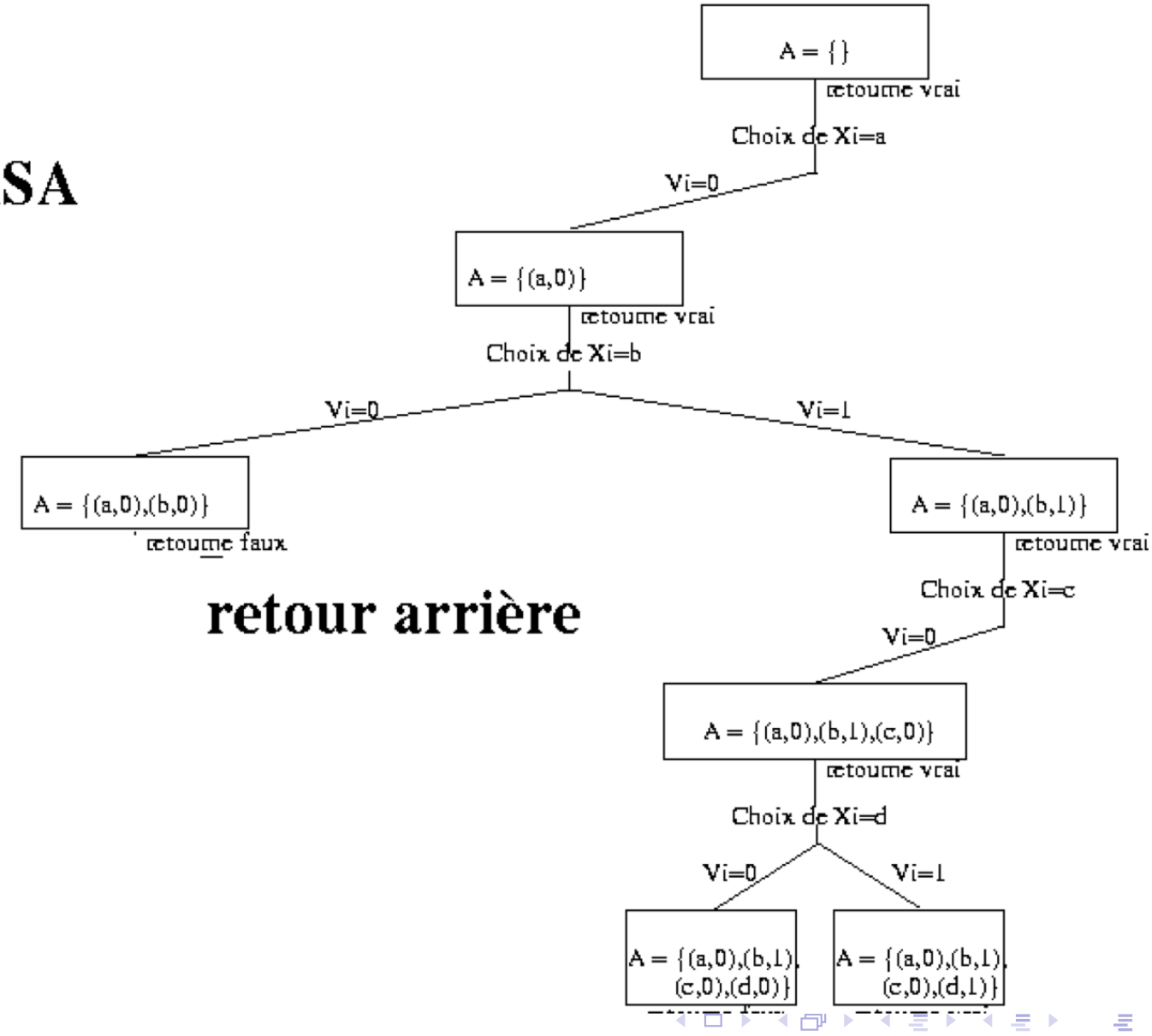
principe

- Construction d'une solution
  - génération d'un affectation partielle consistante
  - extension de l'affectation partielle avec l'affection d'une nouvelle variable
  - si cette extension est inconsistante on retourne en arrière on modifie l'affectation de la nouvelle variable

Algorithme Simple_Retour_Arrière : SRA

fonction SRA(A,(X,D,C)) : booléen
début
si A n'est pas consistante alors retourner **alors**
  retourner FAUX
finsi
si toutes les variables de X sont affectées **alors**
  retourner VRAI
sinon
  choisir une variable X_i de X qui n'est pas encore affectée
  pour toute valeur V_i appartenant à D_i **faire**
    si SRA(A $\cup$ $\{(X_i, V_i)\}$, (X,D,C)) = VRAI **alors**
      retourner VRAI
    finsi
  fin pour
  retourner FAUX
finsi
fin

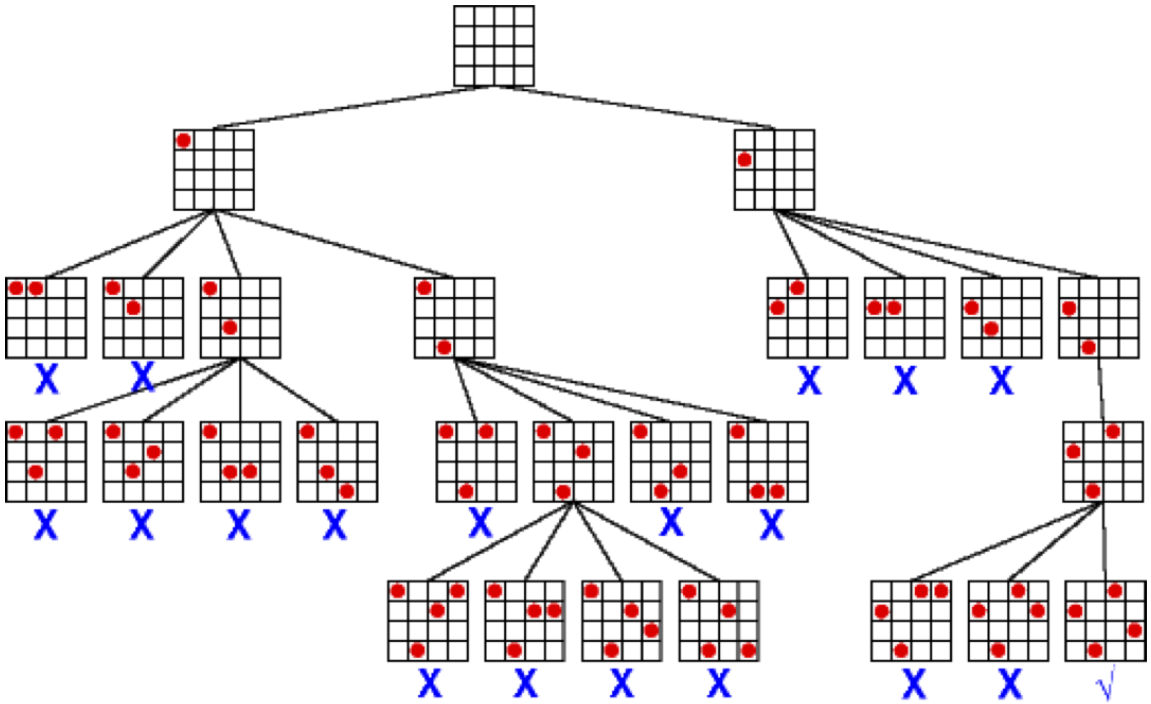
RSA



retour arrière

Algorithme Simple_Retour_Arrière : SRA

exemple du problème des 4 reines



Algorithme Algorithme Simple_Retour_Arrière : SRA

avantages

- réduction de l'espace de recherche
- améliore GET en espace et en temps d'exécution

inconvenients

- pas d'identification des cause de conflit
- redondance
- détection tardive des conflits

alternatives : backjumping, backmarking

Techniques de filtrage

principe

- filtrage des domaines
  - dans la construction d'une affectation partielle consistante : filtrage à priori
  - filtrer le domaine des variables en enlevant les valeurs localement inconsistantes
  - filtrage à différents niveaux : noeud, arc, chemin, $\cdots$

Techniques de filtrage

consistance de noeud

Un CSP (X, D, C, R) est consistant de noeud si pour toute variable X_i de X, et pour toute valeur v de D_i, l'affectation partielle $\{(X_i, v)\}$ satisfait toutes les contraintes unaires de C.

principe

- filtrage des domaines
- pour chaque variable X_i, on enlève de D_i toute valeur v telle que l'affectation partielle $\{(X_i, v)\}$ viole les contraintes unaires.

Techniques de filtrage

consistance d'arc

Un CSP (X, D, C, R) est consistant d'arc si tout couple de variables (X_i, X_j) de X, et pour toute valeur v_i de D_i, il existe une valeur v_j appartenant D_j telle que l'affectation partielle $\{(X_i, v_i), (X_j, v_j)\}$ satisfasse toutes les contraintes binaires de C.

principe

- filtrage des domaines
- pour chaque variable X_i, on enlève de D_i toute valeur v_i telle quil existe une variable X_j pour laquelle, pour toute valeur v_j de D_j, l'affectation partielle $\{(X_i, v_i), (X_j, v_j)\}$ viole les contraintes binaires.

Techniques de filtrage : consistance d'arc

plusieurs algorithmes

- AC ou REVISE : réduit la taille des domaines, supprime les valeurs qui violent les contraintes binaires
- AC1 : réapplique REVISE chaque fois qu'un domaine est changé
- AC3 : ne réapplique REVISE que le nombre de fois nécessaires

Algorithme Consistance_D'Arc : AC

fonction REVISE((X_i, X_j),(X,D,C)) : booléen
début
DELETE $\leftarrow$ FAUX
pour tous les V_i de D_i **faire**
  si il n'y a pas de V_j dans D_j qui satisfasse les contraintes binaires entre X_i et X_j **alors**
    supprimer V_i de D_i
    DELETE $\leftarrow$ VRAI
  finsi
fin pour
retourner DELETE
fin

Algorithme Consistance_D'Arc : AC1

fonction AC1((X,D,C))
début
Q $\leftarrow$ $\{(X_i, X_j)$ /il existe une contrainte entre X_i et X_j $\}$
rpter
    R $\leftarrow$ FALSE
    pour tous les (X_i, X_j) de Q **faire**
        R $\leftarrow$ (R ou REVISE((X_i, X_j),(X,D,C)))
    fin pour
jusqu' non R
retourner (X,D,C)
fin

Algorithme Consistance_D'Arc : AC3

fonction AC3((X,D,C))
début
Q $\leftarrow$ $\{(X_i, X_j)$ /il existe une contrainte entre X_i et X_j $\}$
tantque Q $\neq \emptyset$ **faire**
  Q $\leftarrow$ Q $\setminus (X_i, X_j)$
  si REVISE((X_i, X_j),(X,D,C)) **alors**
    Q $\leftarrow$ Q $\cup \{(X_k, X_i)$ /il existe une contrainte entre X_k et X_i et $X_k \neq X_i$ et $X_k \neq X_j\}$
  finsi
fin tantque
retourner (X,D,C)
fin

Techniques de filtrage : consistance locale

plusieurs notions

- 1- consistance : consistance de noeud
- 2- consistance : consistance d'arc
- 3-consistance : consistance de chemin
  un ensemble $\{X_i, X_j\}$ est consistant de chemin par rapport à la variable X_k de X, si pour toute affectation partielle $\{(X_i, v_i), (X_j, v_j)\}$, il existe une valeur v_k appartenant D_k telle que $\{(X_i, v_i), (X_k, v_k)\}$ est consistant et $\{(X_k, v_k), (X_j, v_j)\}$ est consistant.
- ...
- k-consistance : considérer les valeurs d'une variable par rapport à la combinaison de $k-1$ valeurs des autres variables.

Techniques de propagation de contraintes

principe

- Forward checking et Look-ahead
  - Combinaison du filtrage et du retour-arrière : filtrage au cours de la résolution
  - Forward checking : Après chaque affectation d'une varible X_i, filtrer le domaine de la variable X_j non encore affectèes dont les valeurs violent les contraintes contenant X_i et X_j.
  - Look ahead : Après chaque affectation d'une varible X_i, filtrer le domaine de toutes les variable non encore affectèes dont les valeurs violent les contraintes contenant X_i et ces variables.

Techniques de propagation de contraintes

consistance de noeud

Un CSP (X, D, C, R) est consistant de noeud si pour toute variable X_i de X, et pour toute valeur v de D_i, l'affectation partielle $\{(X_i, v)\}$ satisfait toutes les contraintes unaires de C.

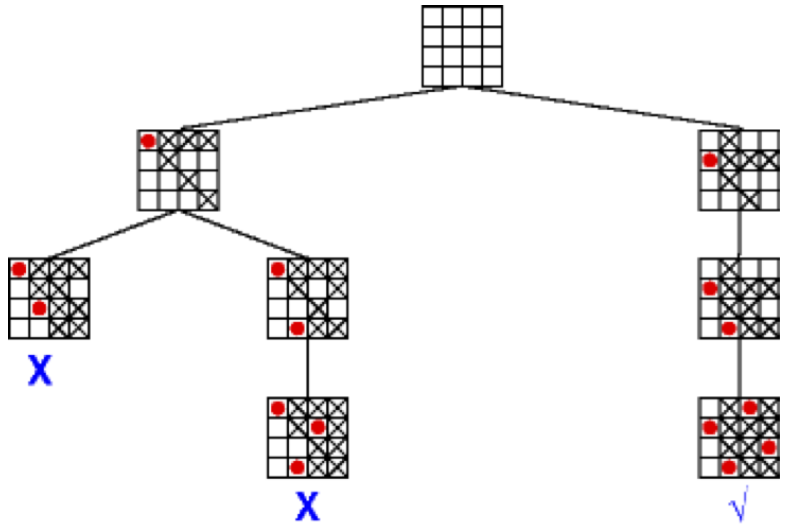
principe : anticipation + consistance de noeud

- anticipation d'une étape l'affectation
- pour chaque variable X_i non affectée dans A, on enlève de D_i toute valeur v telle que $A \cup \{(X_i, v)\}$ est inconsistante.

Algorithme anticipation : forward checking

fonction FC(A,(X,D,C)) : booléen
début
si toutes les variables de X sont affectées **alors**
retourner VRAI
sinon
choisir une variable X_i de X qui n'est pas encore affectée
pour toute valeur V_i appartenant à D_i **faire**
pour toute variable X_j de X qui n'est pas encore affectée **faire**
$D'_j \leftarrow \{V_j$ élément de $D_j \mid$ A $\cup\{(X_i, V_i), (X_j, V_j)\}$ est consistante $\}$
si D'_j est vide **alors**
retourner FAUX
finsi
fin pour
si FC(A $\cup\{(X_i, V_i)\}$,(X,D',C))= VRAI **alors**
retourner VRAI
finsi
fin pour
retourner FAUX
finsi

Algorithme Anticipation : FC



Techniques de propagation de contraintes

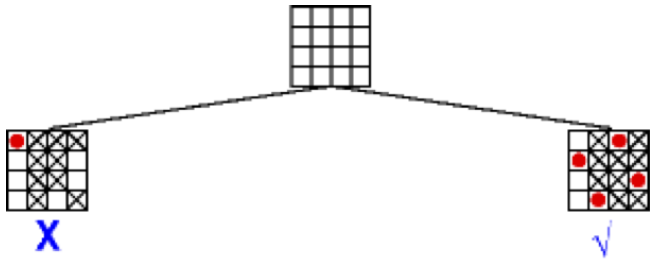
consistance d'arc

Un CSP (X, D, C, R) est consistant d'arc si tout couple de variables (X_i, X_j) de X, et pour toute valeur v_i de D_i, il existe une valeur v_j appartenant D_j telle que l'affectation partielle $\{(X_i, v_i), (X_j, v_j)\}$ satisfasse toutes les contraintes binaires de C.

principe : anticipation + consistance d'arc

- anticipation de deux étapes l'affectation
- pour chaque variable X_i non affectée dans A, on enlève de D_i toute valeur v_i telle quil existe une variable X_j non affectée pour laquelle, pour toute valeur v_j de D_j, l'affectation $A \cup \{(X_i, v_i), (X_j, v_j)\}$ est inconsistante.

Algorithme Look ahead : LH



Heuristiques

- ordre des variables
  - statique
  - dynamique
- ordre des valeurs