Propagation et Satisfaction de Contraintes

EFREI 215/2016 – L3 – Aide à la Décision

Hervé BARBOT, 2016
www.momirandum.com

Exemple introductif : crypt-arithmétique

« DONALD + GERALD = ROBERT »

Par quel chiffre faut-il remplacer chaque lettre

- à chaque lettre correspond un et un seul chiffre
- à chaque chiffre correspond une et une seule lettre

pour que l'expression mathématique soit correcte ?

Reformulation du problème :

« DONALD + GERALD = ROBERT »

D + G = R
O + E = O
N + R = B
A + A = E
L + L = R
D + D = T

D	**O**	**N**	**A**	**L**	**D**
G	**E**	**R**	**A**	**L**	**D**
---	---	---	---	---	---
R	**0**	**B**	**E**	**R**	**T**

Reformulation du problème :

C_5	**C_4**	**C_3**	**C_2**	**C_1**	
D	**O**	**N**	**A**	**L**	**D**
G	**E**	**R**	**A**	**L**	**D**
---	---	---	---	---	---
R	**0**	**B**	**E**	**R**	**T**

Formalisation du problème :

$D + D = T + 10 \times C_1$
$C_1 + L + L = R + 10 \times C_2$
…
$C_5 + D + G = R$

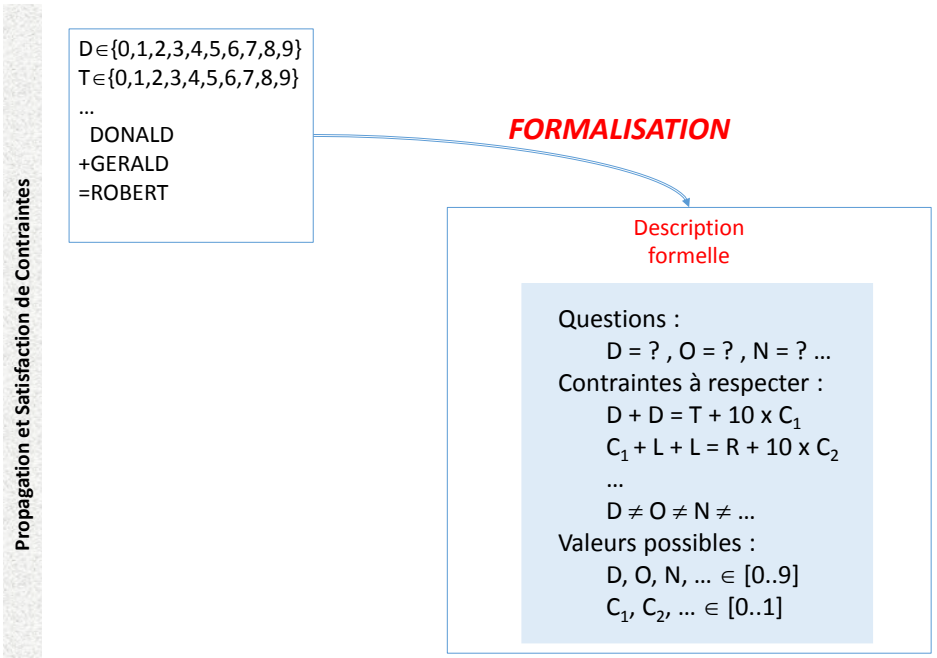
C_5	C_4	C_3	C_2	C_1	
D	O	N	A	L	D
G	E	R	A	L	D
R	0	B	E	R	T

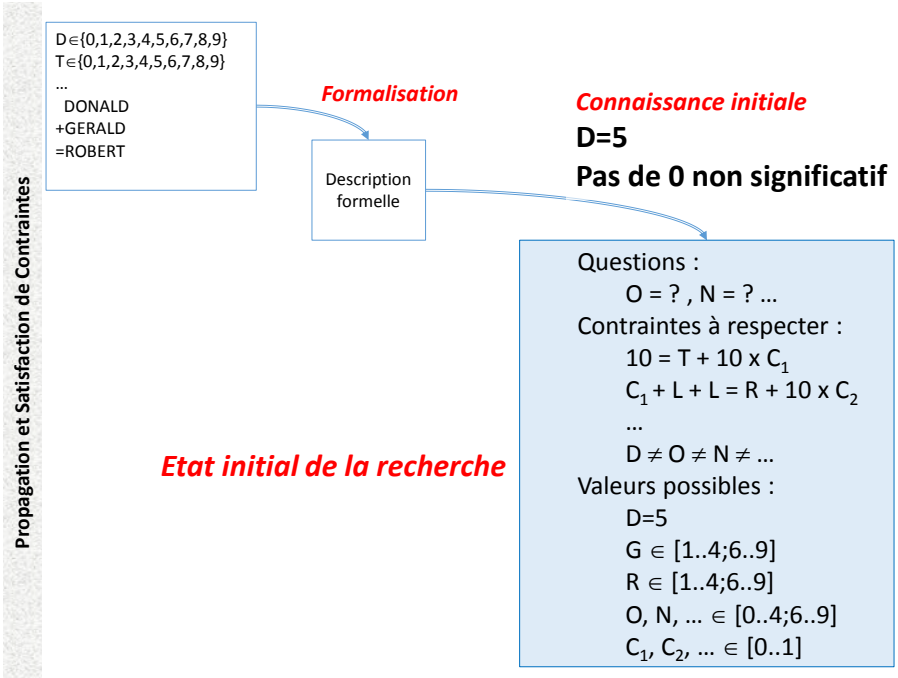
$D \neq O \neq N \neq \ldots$ *lettres 2 à 2 distinctes*
$D, O, N, \ldots \in [0..9]$ *valeurs possibles de 0 à 9*
$C_1, C_2, \ldots \in [0..1]$ *retenues possibles pour 2 opérandes*

Information initiale « privilégiée » pour résoudre le problème plus rapidement

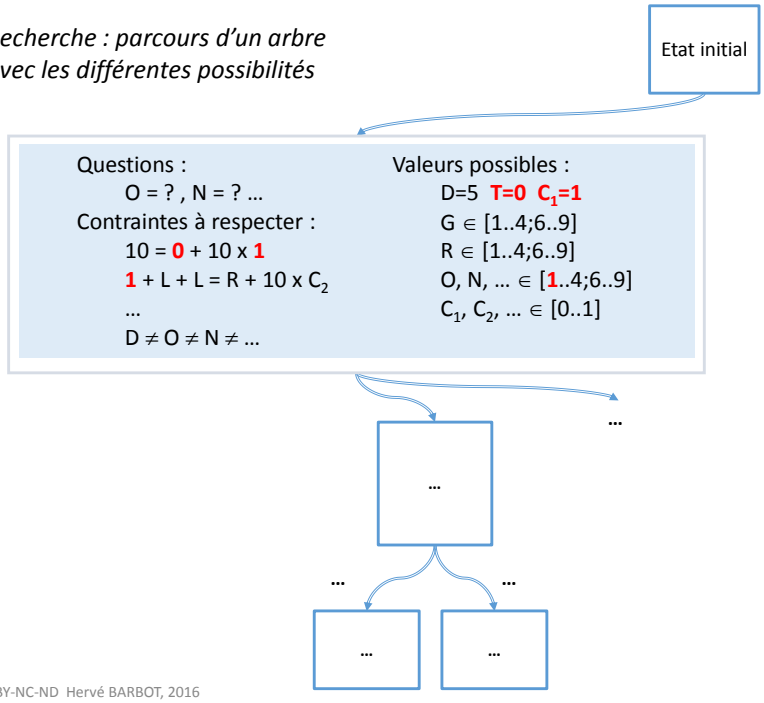
- Par de '0' non significatif
  $D \neq 0$; $G \neq 0$; $R \neq 0$
- D = 5
  - Donc les autres lettres ne peuvent pas être égales à 5 !

C_5	C_4	C_3	C_2	C_1	
5	O	N	A	L	5
G	E	R	A	L	5
R	0	B	E	R	T





Recherche : parcours d'un arbre avec les différentes possibilités



Approche « basique »

Arbre de recherche :

Racine : connaissance initiale

Une branche = un ensemble d'affectation *variable=valeur*

Relation nœud / fils :

- Prise de décision (certitude) :
  création d'une seule branche sous un nœud
  certitude compte-tenu des affectations déjà effectuées
- Hypothèse (incertitude) :
  création de plusieurs branches sous un nœud

Solution : une branche telle qu'en sa feuille, toutes les variables sont affectées

- Idée triviale et peu performante… mais qui donne le(s) résultat(s)
  - On construit un arbre de toutes les possibilités
  - Chaque branche correspond à un ensemble complet d'affectations <variable,valeur>
  - On vérifie aux feuilles si les contraintes sont respectées.
- Complexité ?

  b = plus les domaines de valeurs sont grands, plus il y a de choix à chaque noeud

  m = plus il y a de variables, plus il faudra faire d'affectation avant une réponse complète

  d = m

Exemple des 8 reines

- Placer 8 reines sur un échiquier avec :
  -2 reines ne se mettent pas en échec
  -toute case libre mise en échec
- Arbre de recherche :
  - A chaque niveau on place une reine donc 8 niveaux (profondeur de l'arbre)
  - A chaque niveau on peut placer une nouvelle reine sur l'une des 64 cases donc 64 fils pour chaque nœud de l'arbre
  - Total : 64^8 feuilles dans l'arbre

- Stratégie de largeur ou profondeur d'abord ?

  …il faut de toute façon aller jusqu'en bas de l'arbre

  On cherche une « feuille » ;
  toutes les branches « gagnantes » auront la même profondeur ;
  donc a priori parcours en profondeur d'abord

  Complétude : oui
  Optimalité : oui
  Complexité en temps : $O(b^m)$
  Complexité en espace : $O(b*m)$ / $O(b^m)$?

Pour faire mieux / plus performant

- La démarche est correcte mais :
  - La complexité de l'arbre croit avec la nombre de variables et le nombre de valeurs possibles

- Idée :
  - Propager les informations liées aux placements précédents
    - C'est-à-dire liées à la partie supérieure d'une branche
  - pour éviter les situations d'échec au plus tôt
    - Donc éliminer des branches de l'arbre de recherche

- Représentation explicite des contraintes
  Algorithmes de manipulation de contraintes

PSC
Formalisation

- Un ensemble de **variables**
  - incluant les variables « réponses » au problème posé,
  - mais pas [nécessairement] uniquement

D	**C_1**
O	**C_2**
N	**…**
…	

- Un ensemble **domaines de valeurs**

D, O, N, … ∈ [0..9]
C_1, C_2, … ∈ [0..1]

Chaque variable a un **domaine de valeurs** possibles D_i discret et fini

- Un ensemble de **variables**
  - incluant les variables « réponses » au problème posé,
  - mais pas [nécessairement] uniquement
- Un ensemble **domaines de valeurs**
  - chaque variable a un **domaine de valeurs** possibles D_i discret et fini
- Un ensemble de **contraintes**
  - mise en relation des variables

$$D + D = T + 10 \times C_1$$
$$C_1 + L + L = R + 10 \times C_2$$
$$\dots$$
$$D \neq O \neq N \neq \dots$$

Résolution d'un problème par PSC

- Un ensemble de variables $\{ x_1, x_2, \dots, x_n \}$
  - Chaque variable x_i a un domaine de valeurs possibles $D_i = \{ v_{i,k} \}$ discrets et finis
- Un ensemble de contraintes $\{c_1,c_2,\dots,c_p\}$
  - Chaque contrainte concerne un sous-ensemble de variables
  - Chaque contrainte spécifie les combinaisons autorisées pour les valeurs des variables concernées
- Objectif : **assignation complète** de valeurs aux 'n' variables tout en satisfaisant les p contraintes.

Variables &
Domaines de valeur

VARIABLES	Domaines de valeur
D	{1,2,3,4,5,6,7,8,9}
O	{0,1,2,3,4,5,6,7,8,9}
N	{0,1,2,3,4,5,6,7,8,9}
A	{0,1,2,3,4,5,6,7,8,9}
L	{0,1,2,3,4,5,6,7,8,9}
G	{1,2,3,4,5,6,7,8,9}
E	{0,1,2,3,4,5,6,7,8,9}
R	{1,2,3,4,5,6,7,8,9}
B	{0,1,2,3,4,5,6,7,8,9}
T	{0,1,2,3,4,5,6,7,8,9}
C_1	{0,1}
C_2	{0,1}
C_3	{0,1}
C_4	{0,1}
C_5	{0,1}

Variables &
Domaines de valeur
+
1ère valeur donnée

VARIABLES	Domaines de valeur
D	{5}
O	{0,1,2,3,4,6,7,8,9}
N	{0,1,2,3,4,6,7,8,9}
A	{0,1,2,3,4,6,7,8,9}
L	{0,1,2,3,4,6,7,8,9}
G	{1,2,3,4,6,7,8,9}
E	{0,1,2,3,4,6,7,8,9}
R	{1,2,3,4,6,7,8,9}
B	{0,1,2,3,4,6,7,8,9}
T	{0,1,2,3,4,6,7,8,9}
C_1	{0,1}
C_2	{0,1}
C_3	{0,1}
C_4	{0,1}
C_5	{0,1}

Contraintes

CONTRAINTES
$D + D = T + 10*C_1$
$C_1 + L + L = R + 10*C_2$
…
$C_5 + D + G = R$
$D \neq O$
$D \neq N$
…
…
$B \neq T$

Exemple de crypt-arithmétique

Variables + Domaines de valeurs + Contraintes
+ connaissance initiale

VARIABLES	Domaines de valeur
D	D=5
O, N, A, L, E, B, T	{0,1,2,3,4,6,7,8,9}
G, R	{1,2,3,4,6,7,8,9}
C_1, …,C_5	{0,1}

CONTRAINTES
$10= T + 10*C_1$
$C_1 + L + L = R + 10*C_2$
…
$C_5 + D + G = R$
$D \neq O$
$D \neq N$
…
…
$B \neq T$

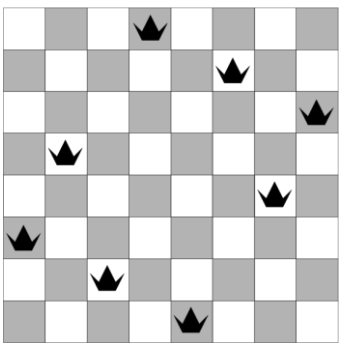
Exercices de modélisation

Définir les PSC, c'est-à-dire :
(1) identifier les variables,
(2) donner leurs domaines de valeurs,
(3) établir les contraintes reliant les variables

8 reines sur un échiquier

Comment placer 8 reines sur un échiquier de façon à ce que :

- Toutes les cases laissées libres soient mises en échec par au moins une reine
- Aucune reine ne met une autre reine en échec



Distributeur – Rendu de monnaie

- Fonctionnement :
  - Le distributeur dispose d'une « caisse » contenant C_2 pièces de 2€, C_1 pièces de 1€, C_{50} pièces de 50c, C_{20} pièces de 20c et C_{10} pièces de 10c.
  - L'utilisateur insère des pièces de monnaies pour un total T.
    - T est un multiple de 10.
  - Il sélectionne une boisson dont le prix est P.
    - P est un multiple de 10.
  - Il détermine s'il peut rendre la monnaie, et sert (ou non) la boisson.

Variante :
L'utilisateur insère des pièces de monnaies pour un total T, au moyen de T_2 pièces de 2€, T_1 pièces de 1€, T_{50} pièces de 50c, T_{20} pièces de 20c, T_{10} pièces de 10c.
Le distributeur peut utiliser ces pièces pour rendre la monnaie.

Crypt-arithmétique

On considère l'addition suivante :

```
    S E N D
+   M O R E
-----------
= M O N E Y
```

où chaque lettre représente un chiffre différent (compris entre 0 et 9). On souhaite connaitre la valeur de chaque lettre, sachant que la première lettre de chaque mot représente un chiffre différent de 0.

Coloriage de carte

- Colorier une carte de sorte que 2 régions voisines n'aient pas la même couleur

Graphe de contraintes

Une représentation des contraintes entre variables

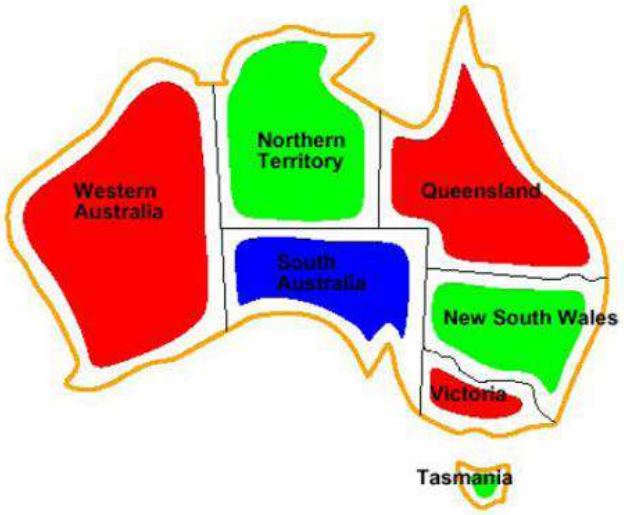
Graphe de contraintes

- Nœuds = variables
  Arcs / arêtes = contraintes
- Hypothèse : contraintes binaires :
  chaque contrainte concerne au plus 2 variables

Plus généralement, on peut avoir:

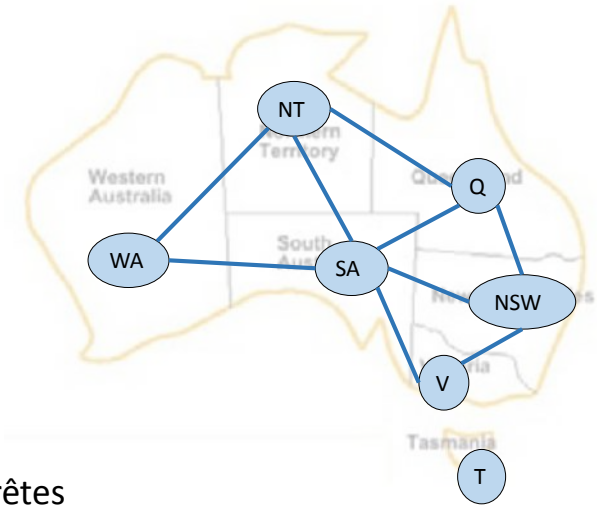
Contraintes unaires
Contraintes binaires
Contraintes n-aires (n>2)
Préférences (« contraintes molles »)
•*Coût / poids associé à chaque affectation possible*

Exemple – Coloriage de carte



Sommets :

- WA
- NT
- SA
- Q
- NSW
- V
- T



Contraintes / arêtes

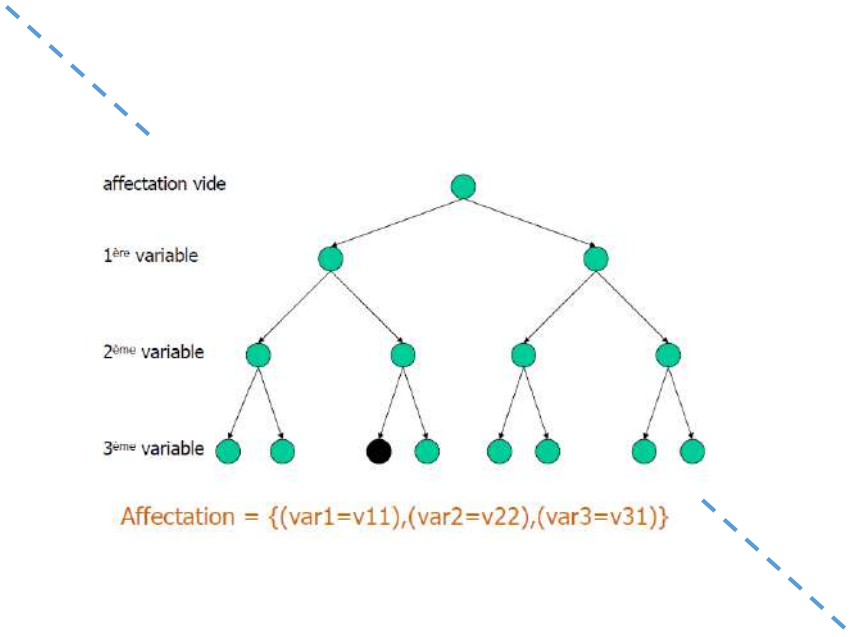
Couleur de « x » ≠ couleur de « y »

Ici, graphe non orienté car relation symétrique

Arbre de recherche
Principes généraux

Exploration complète de l'arbre (méthode naïve)

- Etat initial = aucune affectation
  - *Sauf si affectations initiales connues*
    par ex. crypt-arithmétique
- Fonction « successeur » =
  affectation d'une valeur à une variable
- Test solution =
  affectation complète tout en respectant l'ensemble des contraintes



- Base de connaissance :

Données globales au problème à résoudre :

- Ensemble des variables

$$V = \{ x_i , 1 \leq i \leq n \}$$

- Domaine de valeur pour chaque variable x_i

$$D_i = \{ v_i^k , 1 \leq k \leq |D_i| \}$$

- Ensemble des domaines de valeurs

$$D = \{ D_i , 1 \leq i \leq n \}$$

- Ensemble des contraintes

$$C = \{ c_j , 1 \leq j \leq p \}$$

Données relatives à un nœud 's' particulier :

- Ensemble des affectations faites à un nœud

$$V_A(s) = \{ (x_i, v_i^k) , x_i \in V, v_i^k \in D_i \}$$

- Ensemble des variables sans valeur à un nœud

$$V_{NA}(s) = \{ x_i , x_i \in V \}$$

Une variable est bien évidemment référencée soit dans V_A, soit dans V_{NA}

Base de connaissance au nœud 'n' :

- $V = \{ x_i \}$
- $D_i = \{ v_i^k \}$
- $D = \{ D_i \}$
- $C = \{ c_j \}$

Connaissance globale au PSC

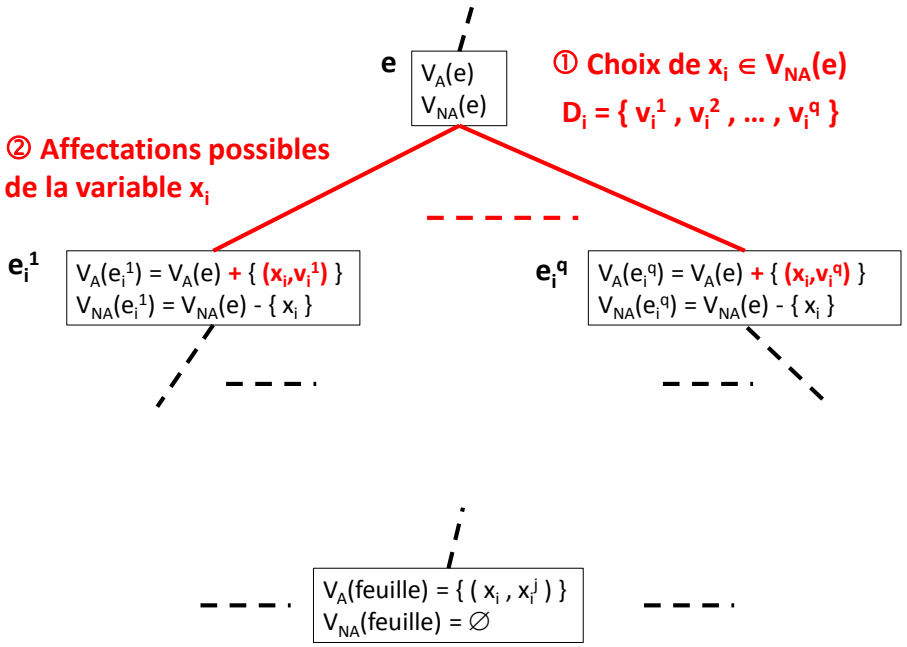
- $V_A(n) = \{ (x_i, v_i^k) \}$
- $V_{NA}(n) = \{ x_i \}$

Connaissance spécifique d'un nœud 'n'

$V \cong V_A + V_{NA}$

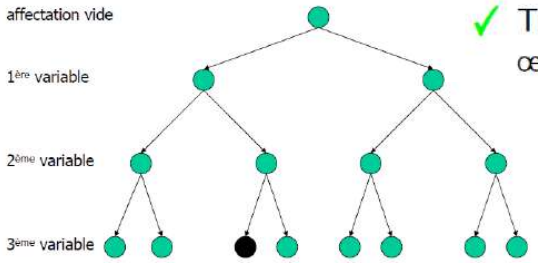
Exploration complète – Définition de l'arbre

- Racine de l'arbre 'r':
  - $V_A(r)$ = { affectations initialement connues }
    - Peut être l'ensemble vide
  - $V_{NA}(r) = V - V_A(r)$
- Descendants 'e_i^j' d'un nœud 'e'
  Choix d'affectation de la variable x_i :
  - $V_A(e_i^j) = V_A(e) + \{ (x_i , v_i^j) \}$ $\quad v_i^j \in D_i$
    Autant de descendants de e que d'éléments dans D_i
  - $V_{NA}(e_i^j) = V_A(e) - \{ x_i \}$
- Feuille de l'arbre
  - $V_{NA}(\text{feuille}) = \varnothing$
  - Solution au problème si $V_A(\text{feuille})$ ne contredit aucune contrainte de C

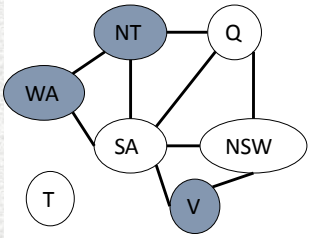


Méthode naïve

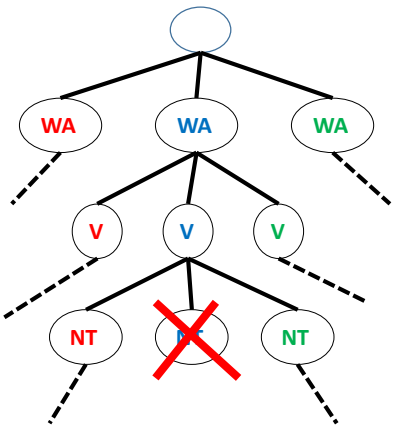
- ✗ Toutes les affectations sont considérées
- ✓ On ne teste les contraintes que sur les affectations totales
- ✓ Très facile à mettre en œuvre



Affectation = {(var1=v11),(var2=v22),(var3=v31)}



Mais :



Conflit avec WA=bleu,
donc ne pas développer !

Vérification de conflit

Pour éviter de développer tout l'arbre des combinaisons

Exploration complète de l'arbre (méthode naïve)

- Etat initial = aucune affectation
  - *Sauf si affectations initiales connues*
    par ex. crypt-arithmétique
- Fonction « successeur » =
  affectation d'une valeur à une variable

Sans entrer en conflit avec les affectations faites sur la branche

SI (x_i , v_i^j) n'entre pas en conflit avec $V_A(e)$, alors :

- $V_A(e_i^j) = V_A(e) + \{ (x_i , v_i^j) \}$ $\quad v_i^j \in D_i$
- $V_{NA}(e_i^j) = V_A(e) - \{ x_i \}$

- Test solution =
  affectation complète tout en respectant l'ensemble des contraintes

Si aucune affectation n'est possible à partir d'un nœud de l'arbre, pour une variable donnée
c'est-à-dire :

Si il n'existe pas (x_i , v_i^j) n'entrant pas en conflit avec V_A(e)

Alors abandonner la branche et remonter d'un niveau

- Une branche ne sera donc développée que si aucune affectation n'est incohérente avec celles faites précédemment
- Donc une feuille est nécessairement une solution du problème

Recherche PSC en « profondeur d'abord »

affectation PSC (affectation <V_A,V_{NA}>) :

SI feuille ALORS retourner V_A (« succès »)

Soit :

$X \in V_{NA}$ *une variable non assignée*
D_X *son domaine de valeurs*

$\forall$ v $\in$ D_X, v consistante avec V_A et C,
$V_{A'} \leftarrow$ PSC (V_A + { X,v })
$V_{A'} \neq$ échec $\Rightarrow$ retourner $V_{A'}$

Retourner « échec »

var V , D , C
type *affectation* $= < V_A , V_{NA} >$

affectation PSC (affectation A)
{
  $A.V_{NA} = \varnothing \Rightarrow$ retourner A

  Soit $x_i \in A.V_{NA}$
  $\forall v_i^j \in D_i$,
    v_i^j consistent avec $A.V_A$ et C
      $\Rightarrow A' \leftarrow$ PSC ($< A.V_A + (x_i , v_i^j)$, $A.V_{NA} - x_i >$)
        $A' \neq$ échec $\Rightarrow$ retourner A'

  retourner échec
}

Appel initial : PSV (< {} , V >)

Backtrack

- ✓ Moins d'affectations considérées
- ✗ Toutes les contraintes sont testées à chaque affectation même partielle
- ✓ Facile à mettre en œuvre

Méthode naïve

- ✗ Toutes les affectations sont considérées
- ✓ On ne teste les contraintes que sur les affectations totales
- ✓ Très facile à mettre en œuvre

- Importance de l'ordre des variables

Impacts de la formalisation

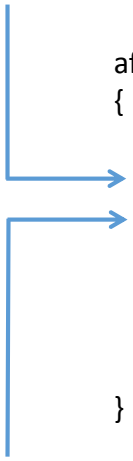
Sur l'efficacité, la performance de la recherche

Taille du problème

- Identifier / définir les variables et les contraintes
  - La profondeur de l'arbre de recherche augmente avec le nombre de variables
  - La largeur de l'arbre de recherche augmente avec la taille des domaines de valeurs des variables
  - La complexité et le temps de calcul nécessaire augmente avec le nombre (et la complexité) des contraintes

Réalités sans doute opposées…

- Ordre de prise en compte des variables



affectation PSC (affectation A)
{
  $A.V_{NA} = \varnothing \Rightarrow$ retourner A
  Soit $x_i \in A.V_{NA}$
  $\forall v_i^j \in D_i$,
    v_i^j consistent avec $A.V_A$ et C
      $\Rightarrow A' \leftarrow$ PSC ($< A.V_A + (x_i , v_i^j)$, $A.V_{NA} - x_i >$)
        $A' \neq$ échec $\Rightarrow$ retourner A'
  retourner échec
}

- Ordre de prise en compte des valeurs dans le domaine possible pour la variable

Ordre de choix des variables

- L'ordre n'est pas nécessairement le même dans toutes les branches

- Variables la plus contrainte
  « ayant le plus petit nombre de valeurs possibles »
  - Réduction du nombre de sous-arbres à évaluer
- Variable la plus contraignante
  « celle impliquée dans le plus de contraintes portant sur des variables non assignées »
  - Diminution du nombre de valeurs restantes possibles pour les autres variables
    - La réduction de complexité se fait au nœud en cours de traitement ou sur les nœuds descendants
- Variable la moins contraignante
  « celle qui laisse le plus de valeurs possibles pour les variables non assignées »
  - On évite [peut-être] les culs-de-sac

Réduction des domaines de valeurs

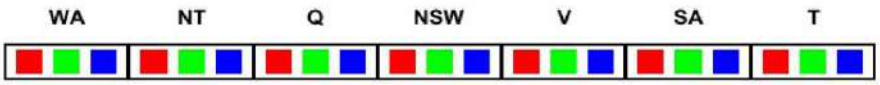
Une façon de vérifier que toute affectation est cohérente avec les choix précédents

Principe

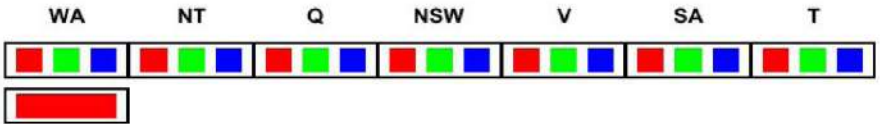
- L'affectation d'une valeur à une variable 'x' réduit le domaine de valeurs possibles pour les autres
  - Celles qui sont sous contrainte avec 'x'
- A chaque affectation (x_i , v_i^j) :
  - Examiner chaque variable x_j non assignée et ayant au moins une contrainte avec x_i
  - Supprimer de D_j les valeurs incompatibles avec $x_i = v_i^j$
  - Si un domaine de valeur D_j devient vide, abandon de la branche de l'arbre de recherche
- Conséquence
  - Lors du choix de x_i pour un nœud donné, toutes les valeurs de D_i sont cohérentes avec les affectations déjà effectuées (dans la branche)

var V , D , C
type *affectation* $= < V_A , V_{NA} >$
affectation PSC (affectation A , domaines D)
{
  $A.V_{NA} = \varnothing \Rightarrow$ retourner A
  Soit $x_i \in A.V_{NA}$
  $\forall\ v_i^j \in D_i$
    $A' \leftarrow < A.V_A + (x_i , v_i^j) , A.V_{NA} - x_i >$
    $D'_i \leftarrow \{ v_i^j \}$
    $\forall\ k \neq i, x_k \in A.V_{NA}$
      $D'_k \leftarrow$ réduction $(D_k , (x_i , v_i^j))$
    $D' \leftarrow \{ D_i, x_i \in A'.VA \} \cup \{ D'_k, x_k \in A.V_{NA} \}$
    $A'' \leftarrow$ PSC (A' , D')
    $A'' \neq$ échec $\Rightarrow$ retourner A''
  retourner échec
}

Exemple

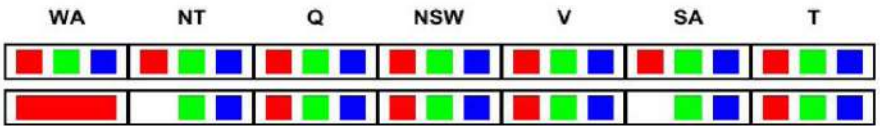


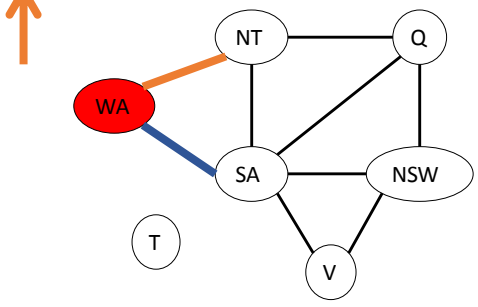
Exemple



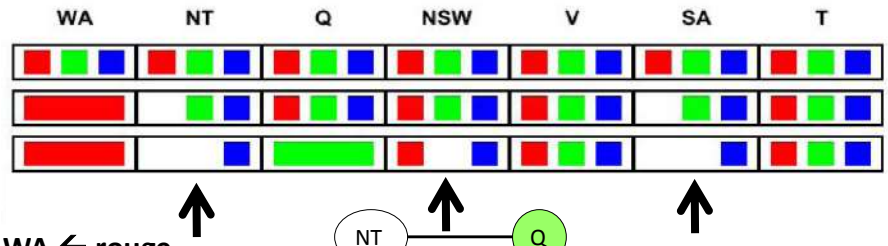
WA ← rouge

Exemple

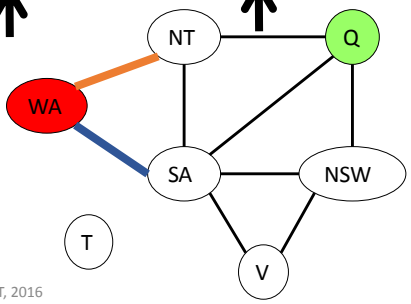




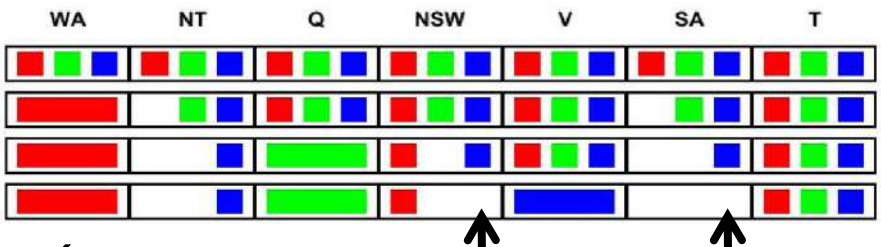
Exemple



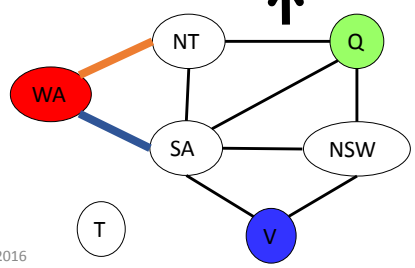
WA ← rouge
Q ← vert



Exemple



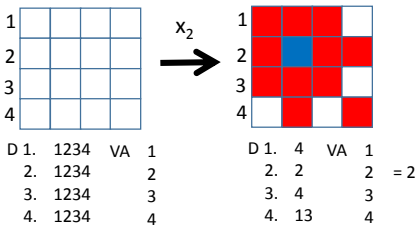
WA ← rouge
Q ← vert
V ← bleu

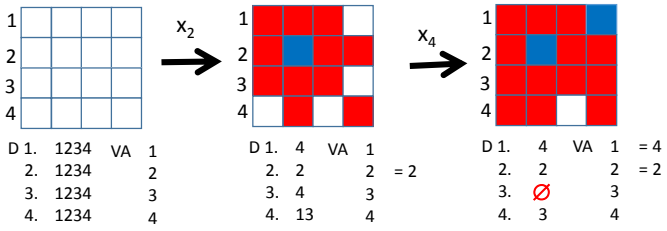


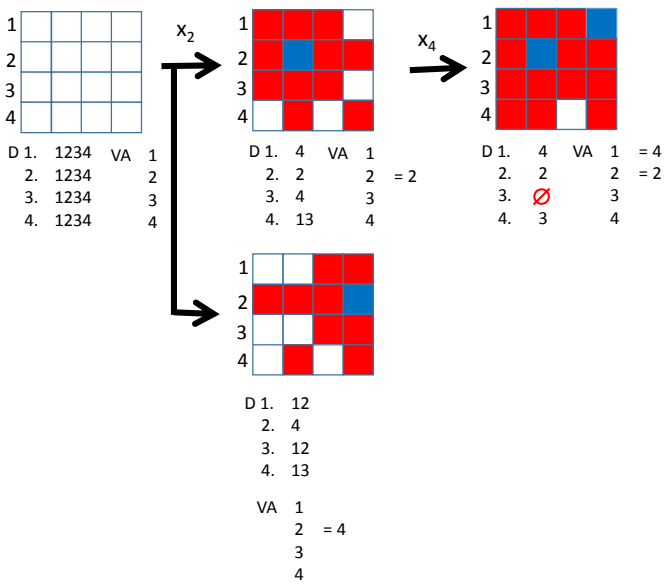
$D_{SA} = \varnothing$

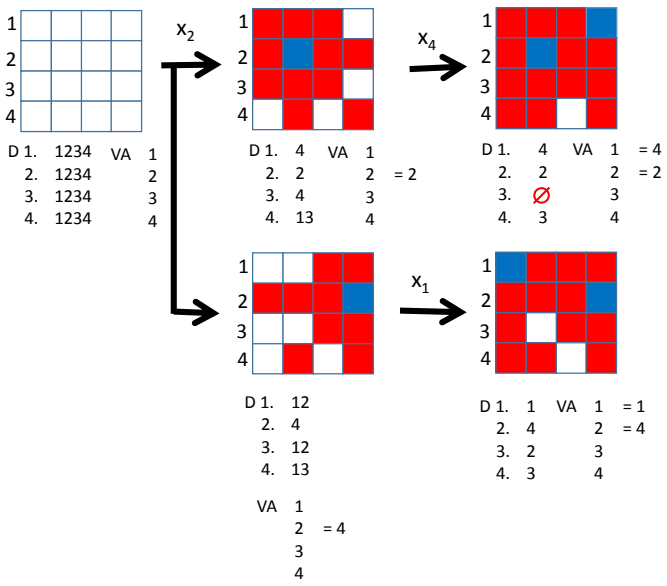
var V , D , C
type $affectation = < V_A , V_{NA} >$
affectation PSC (affectation A , domaines D)
{
  $A.V_{NA} = \varnothing \Rightarrow$ retourner A
  Soit $x_i \in A.V_{NA}$
  $\forall\ v_i^j \in D_i$
    $A' \leftarrow < A.V_A + (x_i , v_i^j) , A.V_{NA} - x_i >$
    $D'_i \leftarrow \{ v_i^j \}$
    $\forall\ k \neq i, x_k \in A.V_{NA}$
      $D'_k \leftarrow$ réduction (D_k , (x_i , v_i^j))
    $\neg$ **($\exists\ k \neq i, x_k \in A.V_{NA} , D'_k = \varnothing$)** $\Rightarrow$
      $D' \leftarrow \{ D_i, x_i \in A'.VA \} \cup \{ D'_k, x_k \in A.V_{NA} \}$
      $A'' \leftarrow$ PSC (A' , D')
      $A'' \neq$ échec $\Rightarrow$ retourner A''
  retourner échec
}

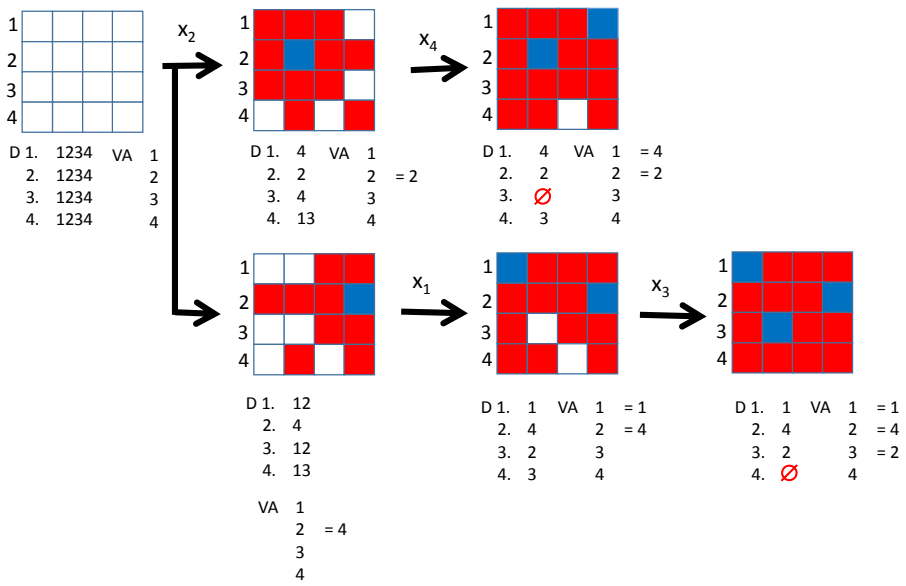
Exemple

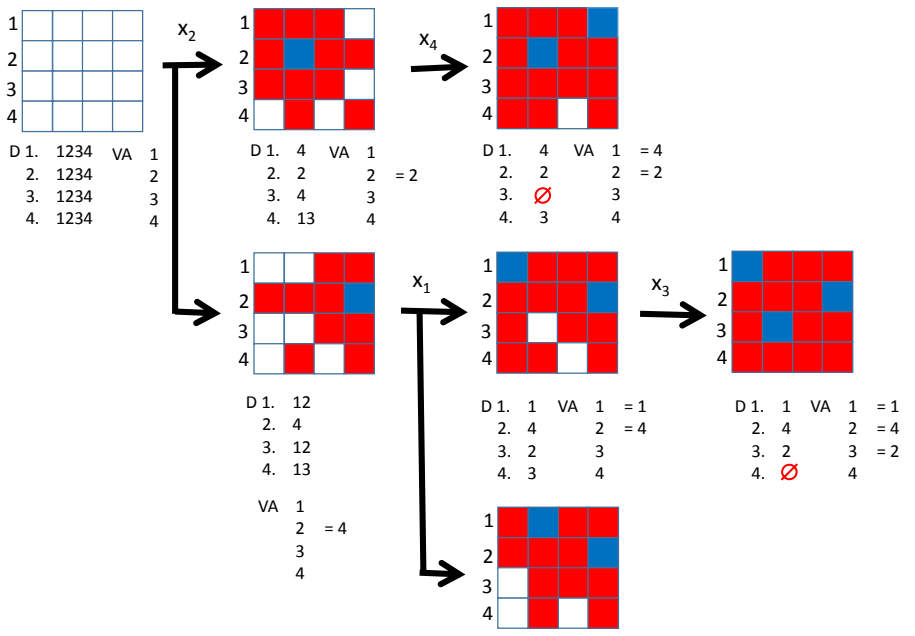






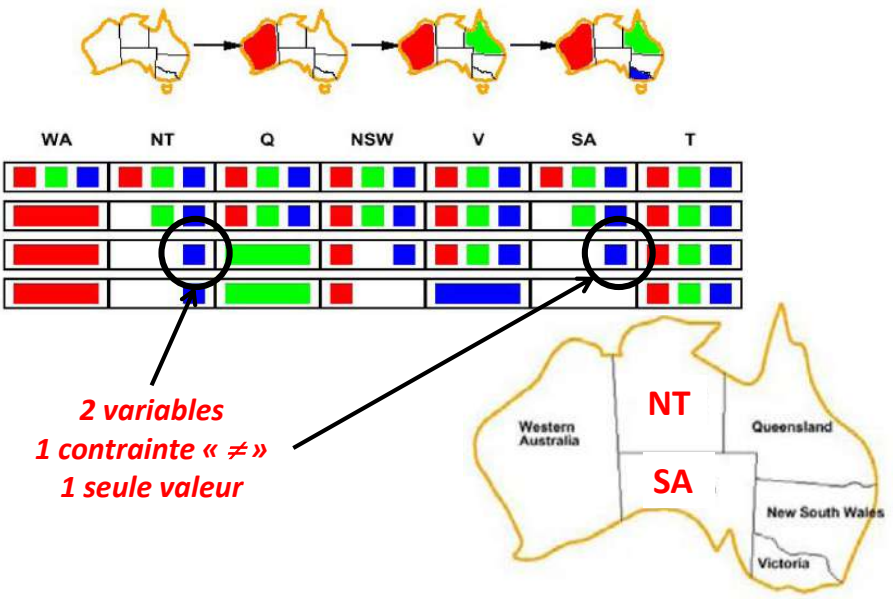






Cohérence d'arête

Amélioration supplémentaire
par anticipation des situations de blocage



Cohérence d'arête - Définition

- Soit une arête (c-à-d une contrainte) entre 2 variables X et Y

- Cohérence si :

  pour toute valeur possible pour X, il reste au moins une valeur possible pour Y

  et vice-versa

Utilisation

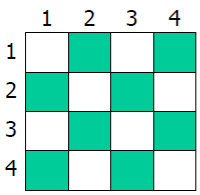
- Vérification de cohérence
  ⇒ réduction des domaines de valeurs
  Elimination des valeurs n'appartenant à aucune solution

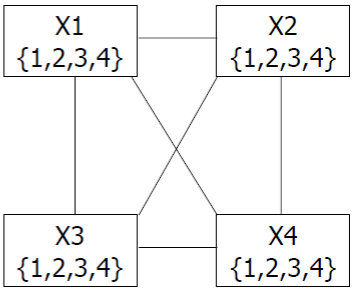
- Exécution :
  - au début de la résolution
  - après chaque affectation '*variable* ← *valeur*'

- Intéressant ssi :
  la vérification peur être faite très rapidement
  - Ex. mise en œuvre particulière du principe pour un problème particulier, plutôt que mise en œuvre générique

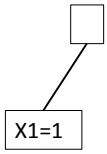
Exemple

1 variable par colonne
Valeurs possibles : n° ligne

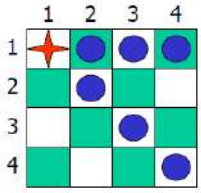


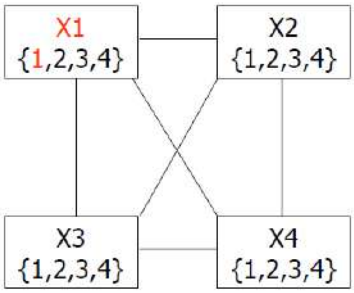


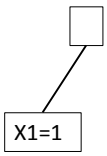
AC3 ne réduit pas les domaines de valeur



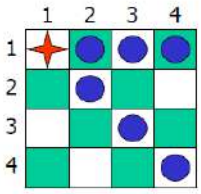
Affectation de la première valeur à la variable X1

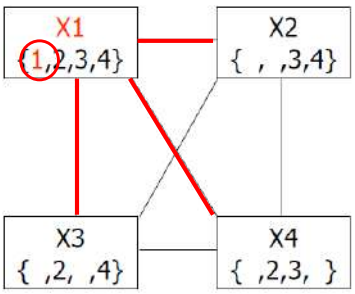




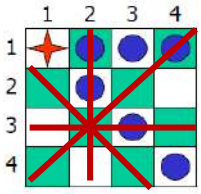


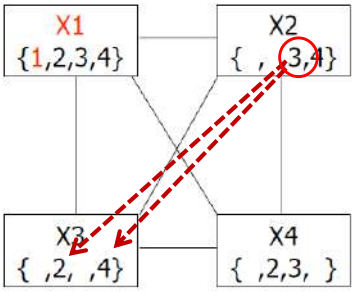
Application du « Forward checking »
Elimination des valeurs contradictoires dans les Domaines de valeurs des autres variables





« Consistence d'arc » : si on affecte X2 = 3 …



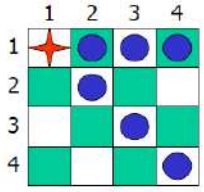


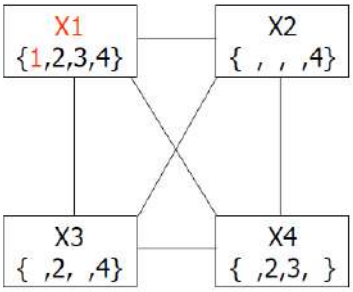
alors il n'y aura plus de valeurs possibles pour X3 !

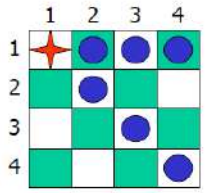
Car une contrainte dit que les diagonales passant par (C2,L3) seront 'fermées' pour les autres reines…

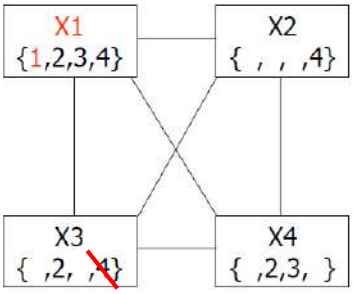
Donc X2 = 3 est impossible

Donc $D_2 = \{ 4 \}$ …

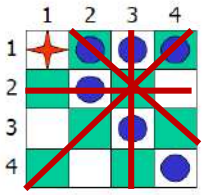


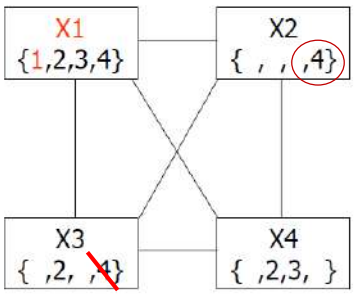






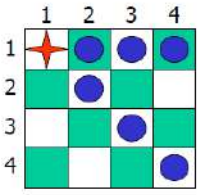
Mais alors $D_3 = \{ 2 \}$ …

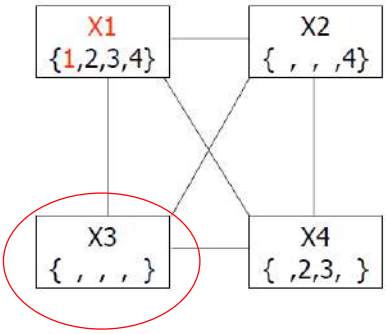




mais X3=2 n'est pas possible car cela enlève les valeurs restantes pour X4…

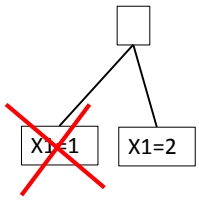
Ligne 2 et diagonale passant par (C3,L2)

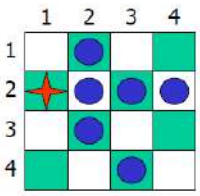


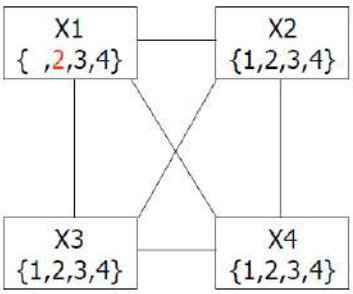


IMPOSSIBLE !

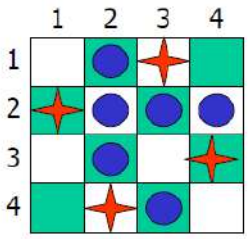
contradiction ➔ backtracking

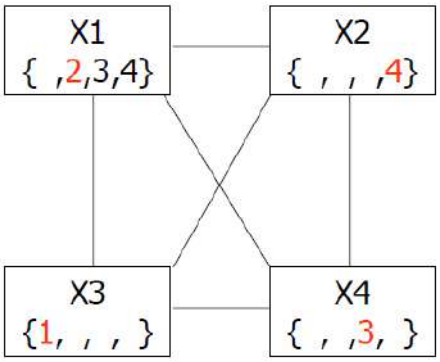






… et ainsi de suite, jusqu'à :





Algorithme AC3

function AC-3(*csp*) **return** the CSP, possibly with reduced domains
  inputs: *csp*, a binary csp with components ($X_{variables}$, $D_{domaines}$, $C_{contraintes}$)
  local variables: *queue,* a queue of arcs initially the arcs in *csp*
  while queue is not empty **do**
    (*X_i, X_j***)** ← REMOVE-FIRST(*queue*)
    if REVISE(*csp*, X_i, X_j) **then**
      if size of D_i = 0 **then return** *false*
      for each X_k **in** X_i.NEIGHBORS – {X_j} **do**
        add **(***X_k, X_i***)** to queue
  return *true*

function REVISE(*csp*, X_i, X_j) **return** *true* iff we revise the domain of X_i
  revised ← *false*
  for each *x* **in** D_i **do**
    if no value *y* in D_i allows *(x,y)* to satisfy the constraints between X_i and X_j **then**
      delete *x* from D_i;
      removed ← *true*
  return *revised*

Initialisation

Itération 1

$C_1 = (x_i , x_j , op)$

$\forall x_i^k \in D_i$,

$(\neg \exists x_j^l \in D_j , C_1 \text{ satisfaite}) \Rightarrow D_i \leftarrow D_i - \{ x_i^k \}$

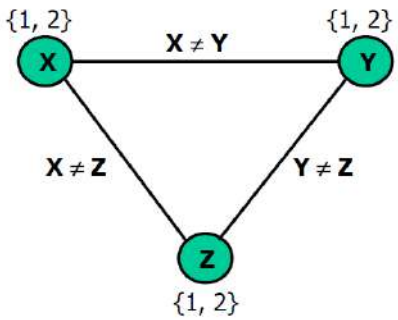
$(D_i = \varnothing) \Rightarrow$ retourner échec

$\forall C_\alpha = (x_i , x_\beta , op_\alpha)$, $x_\beta \neq x_j$, ajouter C_α dans la file

Itération 2

…

Mais est-ce suffisant ?



Il n'y a pas d' « incohérence d'arête » !

Alors qu'il n'y a pas de solution…

