

Problèmes de satisfaction de contraintes

Exemple : colorier une carte

On possède une carte d'Australie montrant ses territoires et états. On a trois couleurs à notre disposition Rouge, Bleu et Vert. Notre problème est de colorier chaque état de sorte que deux états voisins n'aient jamais la même couleur.

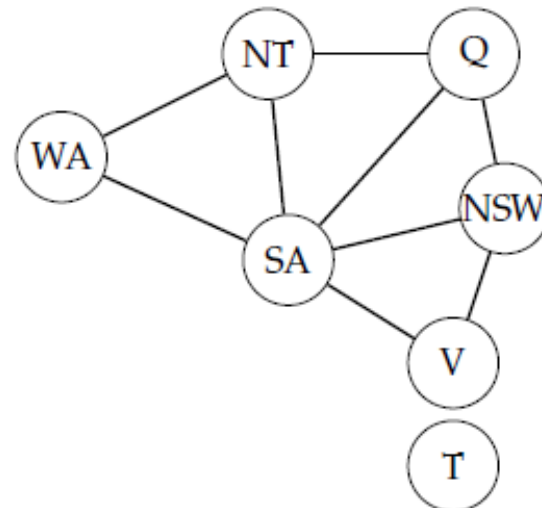


CSP

- Un CSP est défini par:
 - Un ensemble de variables: X_1, X_2, \dots, X_n . Chaque variable X_i a un domaine non vide de valeurs possibles.
 - Un ensemble de contraintes: C_1, C_2, \dots, C_m
- Un état est une affectation de valeurs pour quelques unes ou toutes les variables.
- Affectation consistante: aucune violation de contraintes.
- Affectation complète: toutes les variables ont une valeur.
- Solution: Affectation complète et consistante.
- Si solution optimale, il convient d'ajouter une fonction objective

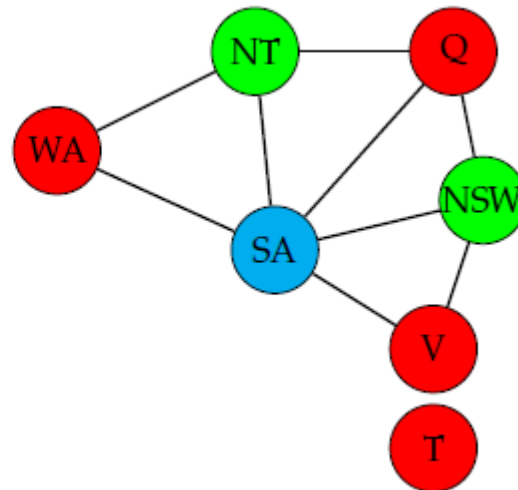
Exemple CSP

- Ensemble des variables : $X = \{WA, NT, SA, Q, NSW, V, T\}$
- Domaine de chaque variable $D_i = \{\text{rouge, bleu, vert}\}$
- Ensemble des contraintes $C = \{WA \neq NT, WA \neq SA, NT \neq SA, SA \neq Q, SA \neq NSW, SA \neq V, NT \neq Q, Q \neq NSW, NSW \neq V\}$
- Une solution possible : $\{WA = \text{rouge}, NT = \text{vert}, Q = \text{rouge}, NSW = \text{vert}, V = \text{rouge}, SA = \text{bleu}, T = \text{rouge}\}$



Exemple CSP

- Ensemble des variables : $X = \{WA, NT, SA, Q, NSW, V, T\}$
- Domaine de chaque variable $D_i = \{\text{rouge, bleu, vert}\}$
- Ensemble des contraintes $C = \{WA \neq NT, WA \neq SA, NT \neq SA, SA \neq Q, SA \neq NSW, SA \neq V, NT \neq Q, Q \neq NSW, NSW \neq V\}$
- Une solution possible : $\{WA = \text{rouge}, NT = \text{vert}, Q = \text{rouge}, NSW = \text{vert}, V = \text{rouge}, SA = \text{bleu}, T = \text{rouge}\}$



Intérêt d'un CSP

- Utilisation des contraintes pour éliminer un bon nombre d'états
- Par exemple dès que SA = bleu, aucun nœud voisin ne peut prendre la couleur bleu.
- Au lieu de devoir considérer $3^5 = 243$ allocations possibles une fois que SA = bleu, il ne reste plus que $2^5 = 32$ allocations possibles !
- Dans les recherches dans les espaces d'états, on peut simplement demander "cet état est-il oui ou non solution"
- Avec un CSP, si une solution partielle n'est pas solution, on n'a pas besoin de considérer ses complétions

Domaine

Le domaine d'une variable peut être:

- discret et fini: on pourrait énumérer toutes les possibilités pour les contraintes
- infini : un langage de contraintes est nécessaire pour représenter toutes les contraintes sans énumération
- continu : par exemple en programmation linéaire, les contraintes sont des égalités ou inégalités de combinaisons linéaires des variables

Les contraintes

Les contraintes peuvent être:

- unaires : les contraintes ne concernent qu'une variable
- binaires : met en relations deux variables. Si toutes les relations sont binaires, on peut représenter le problème avec un graphe.
- complexes : la relation est entre trois variables ou plus
- globales : met en relation toutes les variables (par exemple, une contrainte peut être que toutes les variables prennent des valeurs différentes)
- pour des domaines discrets et finis, on peut toujours se rapporter à un problème avec des relations binaires (en introduisant des variables auxiliaires).
- certaines contraintes indiquent des préférences, ex : dans un problème d'emploi du temps, professeur A préfère enseigner le matin et professeur B l'après midi

Applications des CSP

- Problèmes d'assignation
Ex: Qui enseigne quel cours?
- Problèmes d'horaire
Ex: Quelle classe, quand et où?
- Planification pour le transport
- Planification à l'intérieur d'une usine

Exemple: Problème des n-reines

- On veut disposer n reines sur un échiquier de taille $n \times n$ sans qu'aucune reine ne s'attaque :
 - ensemble de variables est l'ensemble des reines : $X = \{R_1, \dots, R_n\}$
 - le domaine de chaque reine est l'ensemble des positions sur l'échiquier
 - les contraintes indiquent que les reines ne doivent pas s'attaquer entre elles
- On peut aussi réduire le nombre d'états en ajoutant que la reine i est placée sur la colonne i

Exemple: Sudoku

- 81 variables
- cases vides : domaine $\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$
- cases remplies : domaine est un singleton contenant le chiffre
- Une contrainte allDiff par ligne, par colonne, et par carré

	1	2	3	4	5	6	7	8	9
A			3		2		6		
B	9			3		5			1
C			1	8		6	4		
D			8	1		2	9		
E	7								8
F			6	7		8	2		
G			2	6		9	5		
H	8			2		3			9
I			5		1		3		

(a)

	1	2	3	4	5	6	7	8	9
A	4	8	3	9	2	1	6	5	7
B	9	6	7	3	4	5	8	2	1
C	2	5	1	8	7	6	4	9	3
D	5	4	8	1	3	2	9	7	6
E	7	2	9	5	6	4	1	3	8
F	1	3	6	7	9	8	2	4	5
G	3	7	2	6	8	9	5	1	4
H	8	1	4	2	5	3	7	6	9
I	6	9	5	4	1	7	3	8	2

(b)

Exemple: problème d'ordonnancement

- On a un ensemble de n tâches \rightarrow variables $X = \{T_1, \dots, T_n\}$
- Chaque tâche T_i a une durée d_i
- Certaines tâches doivent être effectuées avant d'autres \rightarrow contraintes
- Objectif : trouver un temps (par exemple en minute) où chaque tâche doit commencer \rightarrow domaine

Exemple1 : tâche T_1 dure $d_1 = 10\text{min}$ et doit être effectuée avant tâche T_2 :
 $T_1 + d_1 \leq T_2$

Exemple2 : on veut dire que deux tâches T_3 et T_4 ne peuvent pas avoir lieu en même temps (par exemple si les deux tâches utilisent un outil en commun).
Ici, on veut exprimer que: soit on fait T_3 avant T_4 ou l'inverse, mais on ne veut satisfaire qu'une seule des deux contraintes

$$(T_3 + d_3 \leq T_4) \vee (T_4 + d_4 \leq T_3)$$

Principe de résolution d'un CSP

- assigner une valeur à une variable va avoir un impact sur les valeurs possibles pour les autres variables: **c'est la propagation**
- essayer des affectations de variables: **c'est la recherche**

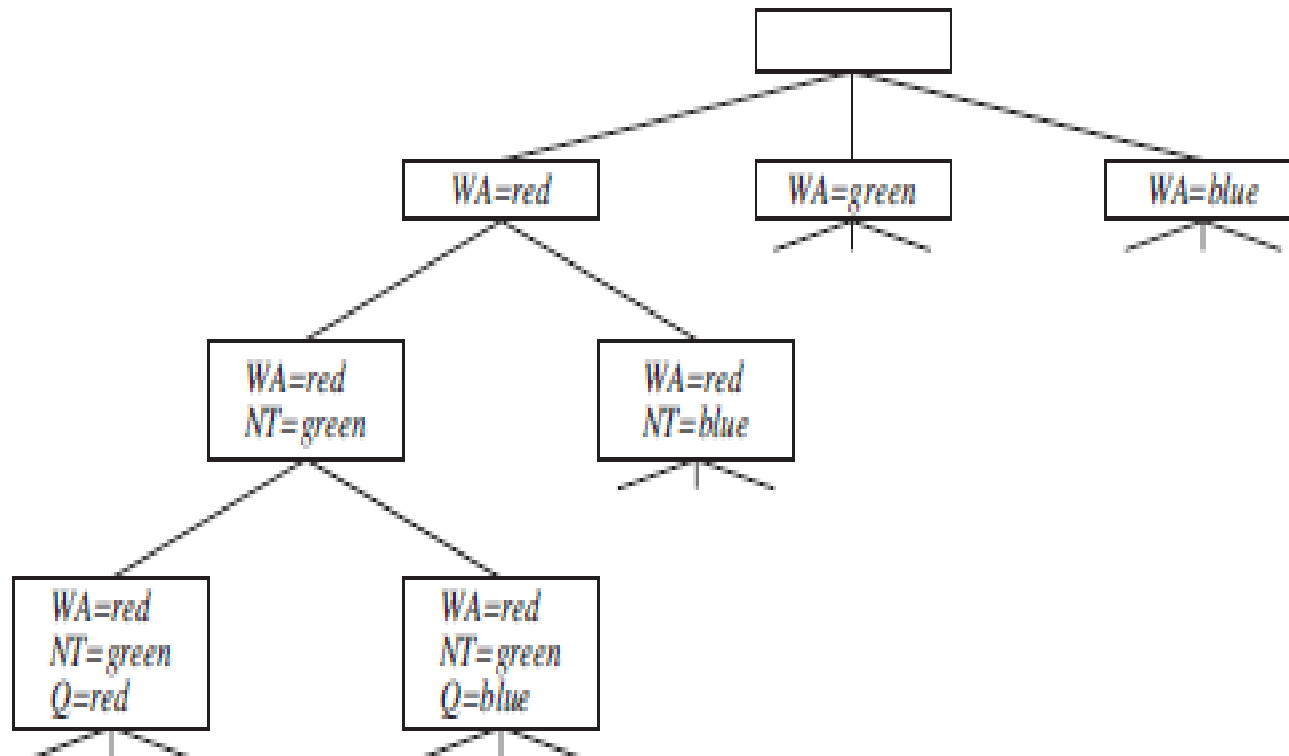


combiner recherche et propagation

Recherche avec retours-arrière

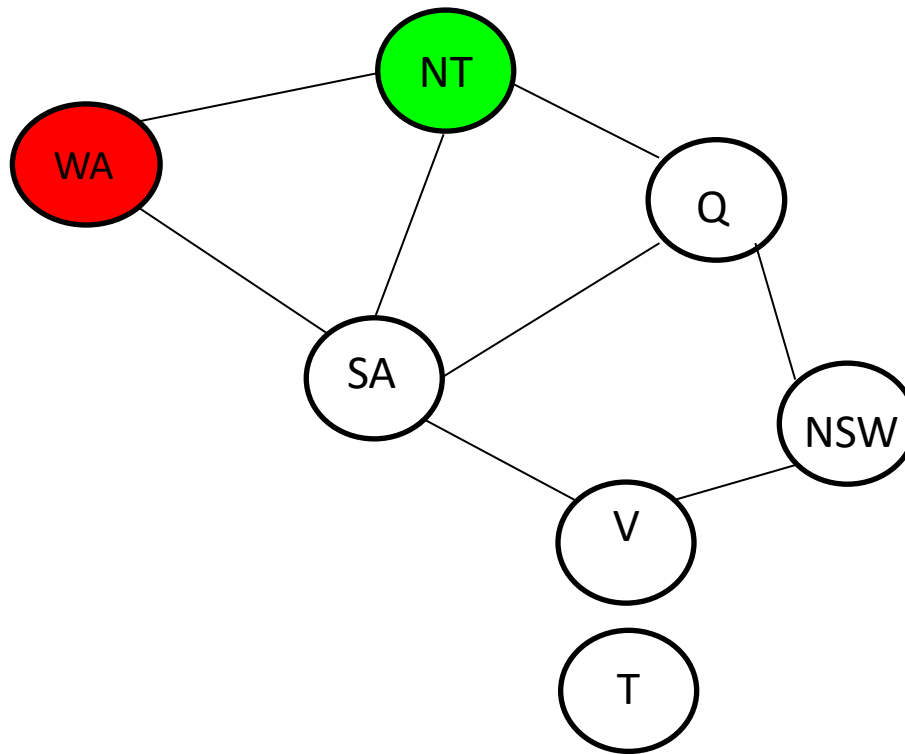
- L'assignation des variables est commutative
Ex: $[WA = \text{rouge} \text{ suivi de } NT = \text{vert}]$ est la même chose que $[NT = \text{vert} \text{ suivi de } WA = \text{rouge}]$
- A chaque noeud, on a seulement besoin de considérer l'affectation d'une seule variable.
Ex: si on a n variables pouvant chacune prendre b valeurs, il y a b^n feuilles
- on va effectuer une recherche en **profondeur d'abord** où à chaque niveau, on cherche à affecter une variable, on fait un retour-arrière(**backtrack**) à chaque fois qu'il n'y a plus de valeurs disponibles pour affecter une variable, on retourne à la variable précédente et on essaye une autre valeur.

Exemple de recherche avec retours-arrière



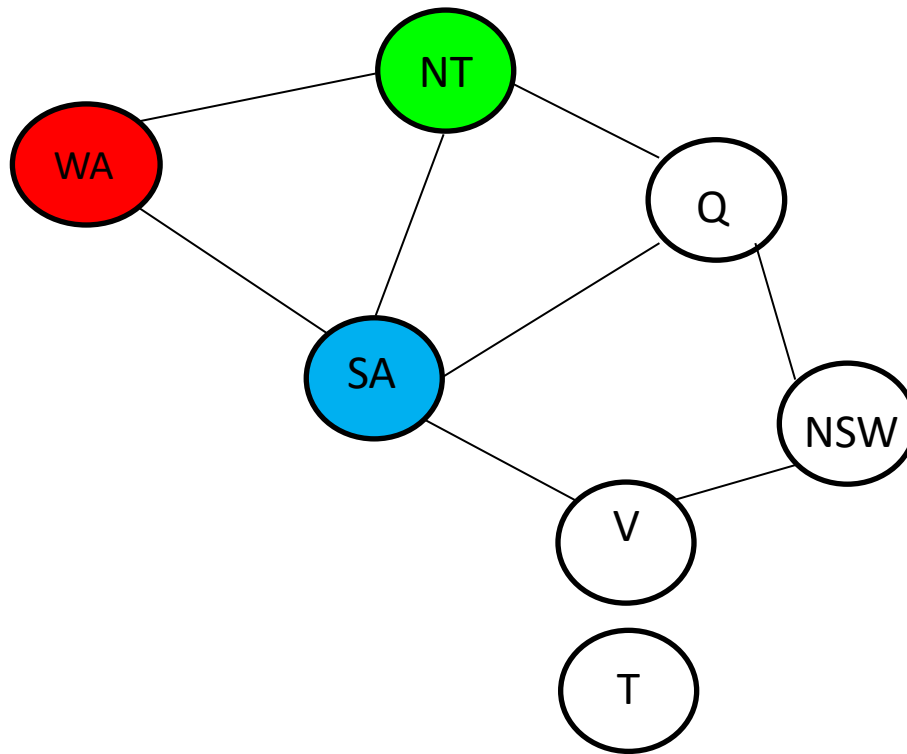
Ordre des variables et des valeurs

- Choix des variables:
 - Heuristique du nombre de valeurs restantes minimum (minimum remaining values (MRV)): Choisir la variable avec le moins de valeurs possibles.



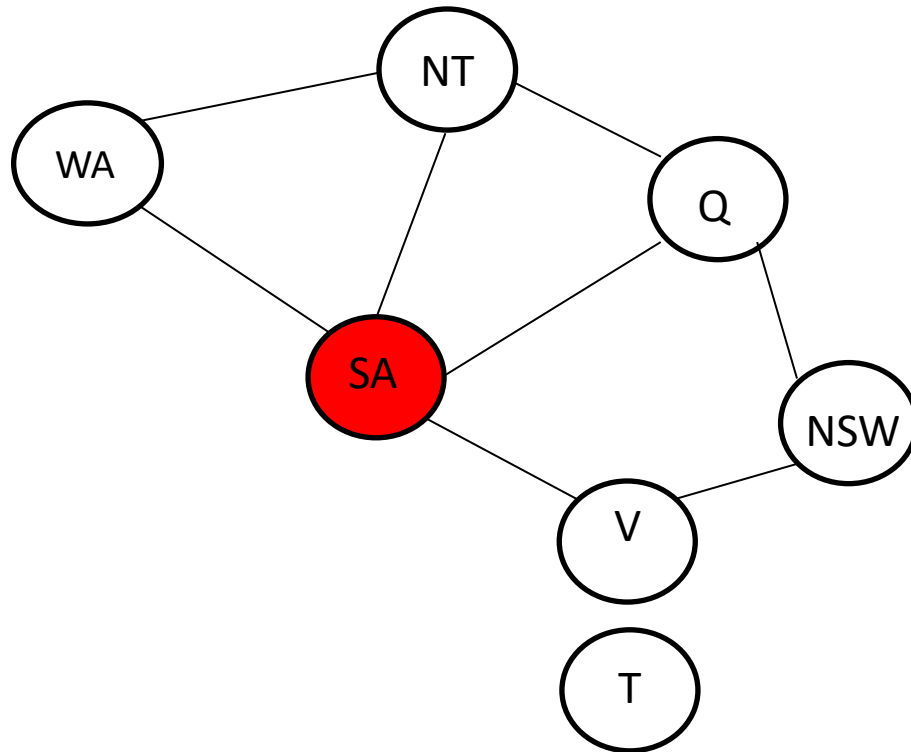
Ordre des variables et des valeurs

- Choix des variables:
 - Heuristique du nombre de valeurs restantes minimum (minimum remaining values (MRV)): Choisir la variable avec le moins de valeurs possibles.



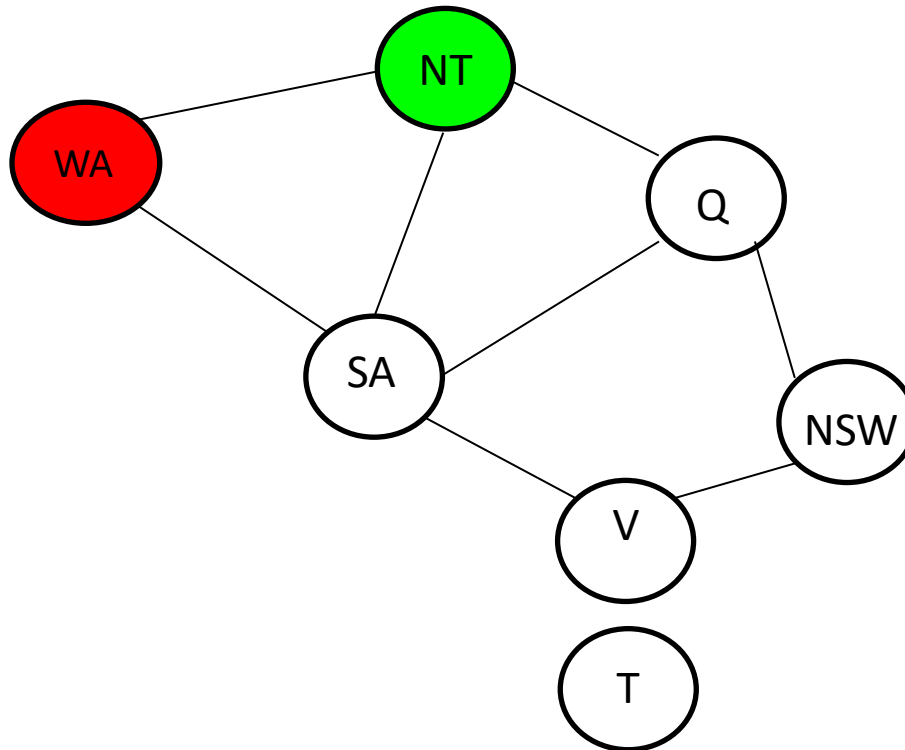
Ordre des variables et des valeurs

- Choix des variables:
 - Heuristique du degré: Choisir la variable présente dans le plus de contraintes.



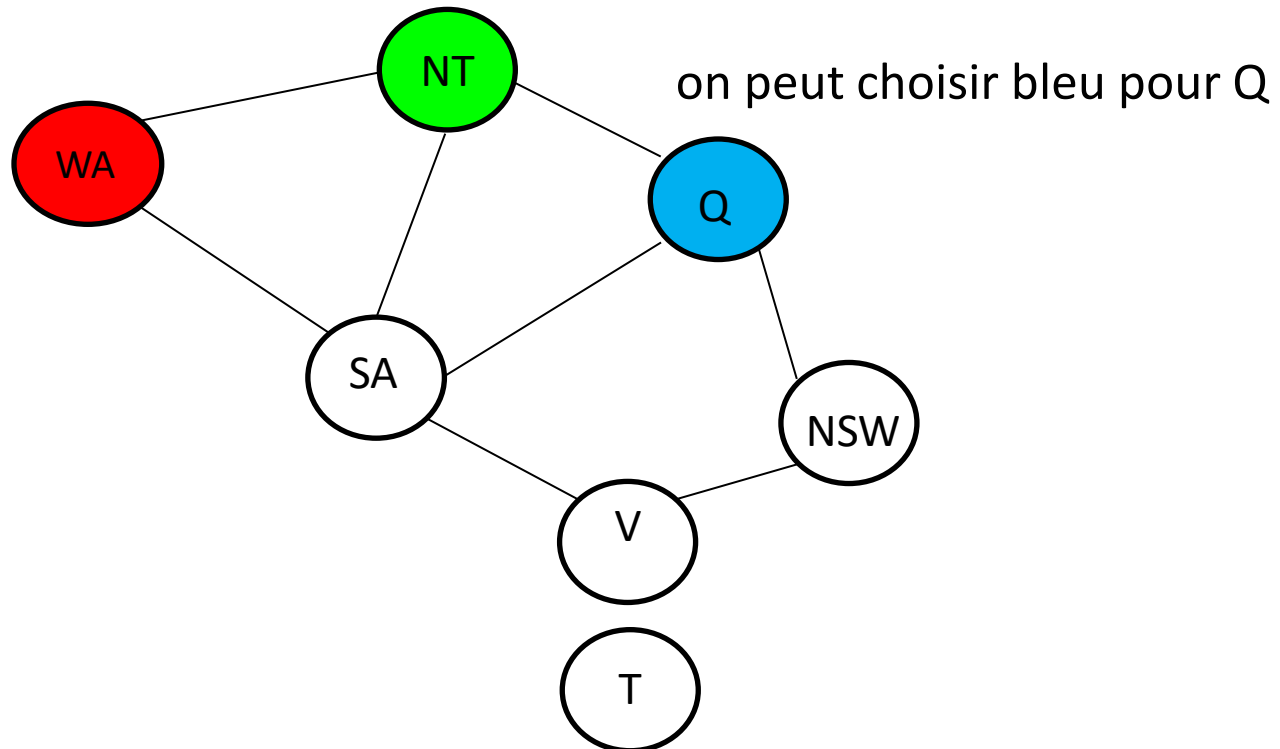
Ordre des variables et valeurs

- Choix des valeurs:
 - Heuristique des valeurs les moins contraignantes (least constraining-value): Choisir la valeur qui va enlever le moins de choix pour les variables voisines.



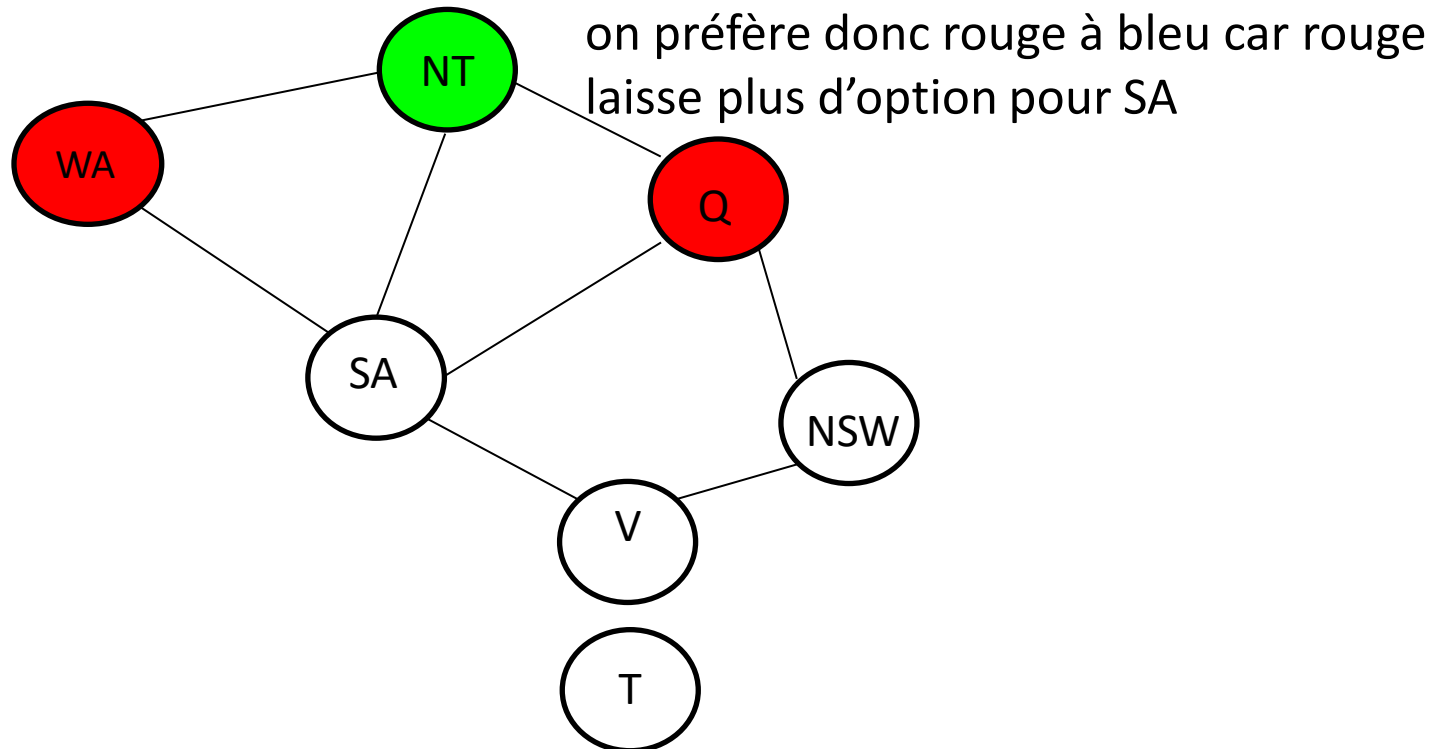
Ordre des variables et valeurs

- Choix des valeurs:
 - Heuristique des valeurs les moins contraignantes (least constraining-value): Choisir la valeur qui va enlever le moins de choix pour les variables voisines.



Ordre des variables et valeurs

- Choix des valeurs:
 - Heuristique des valeurs les moins contraignantes (least constraining-value): Choisir la valeur qui va enlever le moins de choix pour les variables voisines.



Propagation de contraintes

Comment mettre à jour les domaines des variables voisines lorsqu'on a choisi une valeur pour une variable ?

- **Noeud cohérence** : une variable est “noeud cohérente” si elle satisfait toutes les contraintes unaires.
Il suffit de boucler sur chaque variable pour garantir cette cohérence.
- **Arc cohérence** : Un arc entre X et Y est cohérent si pour toutes les valeurs x de X , il y a au moins une valeur possible y de Y . Ex:
 - contrainte $Y = X^2$
 - domaine de X et Y : ce sont des chiffres $DX = DY = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
 - pour que (X, Y) soit arc cohérent, réduction à $DX = \{0, 1, 2, 3\}$
 - pour que (Y, X) soit arc cohérent, réduction à $DY = \{0, 1, 4, 9\}$
- Si X perd une valeur, les voisins de X doivent être revérifiés
- le CSP est arc cohérent si tous ses arcs sont cohérents

function AC-3(*csp*) **returns** false if an inconsistency is found and true otherwise

inputs: *csp*, a binary CSP with components (X, D, C)

local variables: *queue*, a queue of arcs, initially all the arcs in *csp*

while *queue* is not empty **do**

$(X_i, X_j) \leftarrow \text{REMOVE-FIRST}(\text{queue})$

if REVISE(*csp*, X_i, X_j) **then**

if size of $D_i = 0$ **then return** *false*

for each X_k **in** $X_i.\text{NEIGHBORS} - \{X_j\}$ **do**

 add (X_k, X_i) to *queue*

return *true*

function REVISE(*csp*, X_i, X_j) **returns** true iff we revise the domain of X_i

revised \leftarrow *false*

for each x **in** D_i **do**

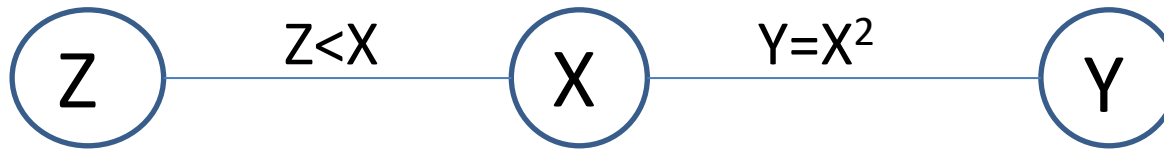
if no value y in D_j allows (x,y) to satisfy the constraint between X_i and X_j **then**

 delete x from D_i



revised \leftarrow *true*

return *revised*

Figure 6.3 The arc-consistency algorithm AC-3. After applying AC-3, either every arc is arc-consistent, or some variable has an empty domain, indicating that the CSP cannot be solved. The name “AC-3” was used by the algorithm’s inventor (Mackworth, 1977) because it’s the third version developed in the paper.



queue	DX	DY	DZ	Paire rajoutée
(X,Y) (Y,X) (X,Z) (Z,X)	{0,1,2,3,4,5,6,7,8,9}	{0,1,2,3,4,5,6,7,8,9}	{0,1,2,3,4,5,6,7,8,9}	
(Y,X) (X,Z) (Z,X)	{0, 1, 2,3}	{0,1,2,3,4,5,6,7,8,9}	{0,1,2,3,4,5,6,7,8,9}	(Z,X)
(X,Z) (Z,X)	{0, 1, 2,3}	{0, 1, 4,9}	{0,1,2,3,4,5,6,7,8,9}	
(Z,X)(Y,X)	{1,2,3}	{0,1,4,9}	{0,1,2,3,4,5,6,7,8,9}	(Y,X)
(Y,X)	{1,2,3}	{0,1,4,9}	{0,1,2}	
	{1,2,3}	{1,4,9}	{0,1,2}	

- après l'exécution de l'algorithme, on a un CSP équivalent mais il peut y avoir moins de valeurs dans chaque domaine
 la recherche pour résoudre le CSP sera sûrement plus facile
- pour des contraintes binaires, la complexité de l'algorithme est $O(cd^3)$ où d est le nombre de valeurs maximum dans chaque domaine et c est le nombre de contraintes
- Remarque: Si on a seulement deux couleurs, le csp est arc-cohérent mais il n'y a pas de solution !: WA, NT et SA se touchant, il faut au moins 3 couleurs !
 l'arc-cohérence ne peut détecter le problème

Autres types de cohérence

- **cohérence de chemin** $\{X_i, X_j\}$ est “chemin-cohérent” par rapport à X_k si pour chaque affectation $X_i = a$ et $X_j = b$ cohérente avec les contraintes sur $\{X_i, X_j\}$, il existe toujours une affectation de X_k cohérente avec les contraintes sur $\{X_i, X_k\}$ et $\{X_j, X_k\}$.
ex : Coloration de la carte avec deux couleur rouge et bleu.
Question : Peut-on rendre $\{WA, SA\}$ chemin-cohérent par rapport à NT ?
affectation $WA = \text{bleu}$ et $SA = \text{rouge}$.
→ NT ne peut pas être affecté sans violer une contrainte ($WA \neq NT$ ou $SA \neq NT$).
→ on détecte l'impossibilité.
- **k-cohérence:** (Notion plus forte) un CSP est k-cohérent si pour chaque ensemble de k-1 variables et pour chaque affectation cohérente de ces variables, une valeur cohérente peut toujours être trouvée pour la kième variable.

Forward checking

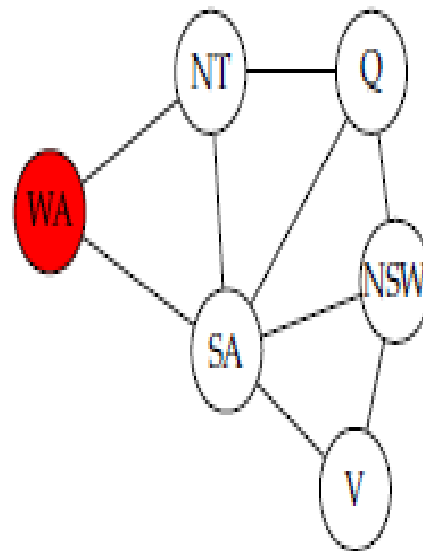
On allie la recherche et la propagation des contraintes selon le principe suivant :

- on effectue une recherche en profondeur d'abord
- un noeud de l'arbre de recherche correspond à l'affectation d'une variable. A chaque affectation d'une variable X :
 - on vérifie l'arc-cohérence entre chacun des voisins de X dans le graphe des contraintes et X
 - on réduit le domaine des voisins de X
- Remarque: on ne continue pas avec les voisins des voisins

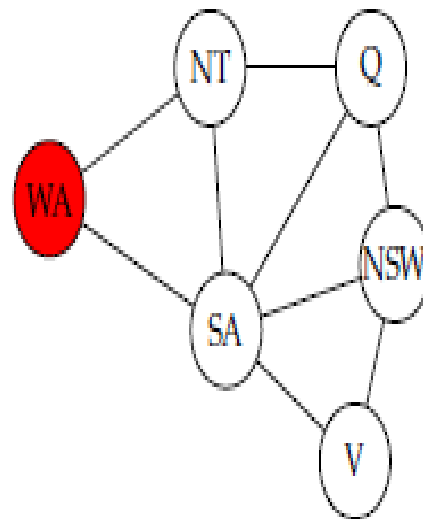
Algorithme

```
1 ForwardChecking (CSP net, Assignment a)
2   if the assignment a is complete then return a
3   var ← next non-assigned variable
4   for each value val ∈ D(var)
5       if {var=val} does not violate any constraint
6           a = a ∪ {var=val}
7           for each variable v connected to var
8               apply arc-consistency from v to var
9           result ← ForwardChecking(net,a)
10          if result ≠ failure
11              return result
12  return failure
```

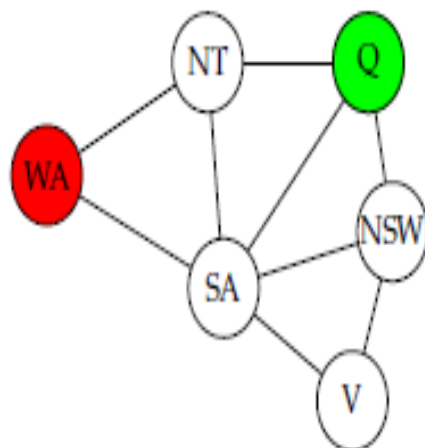
	WA	NT	Q	NSW	V	SA	T
Domaines initiaux	r v b	r v b	r v b	r v b	r v b	r v b	r v b



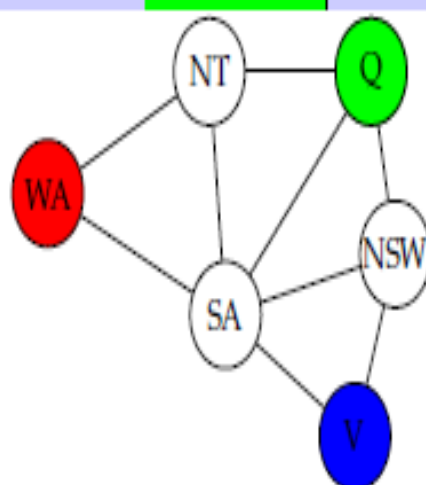
	WA	NT	Q	NSW	V	SA	T
Domaines initiaux	r v b	r v b	r v b	r v b	r v b	r v b	r v b
Affectation WA=r	r	v b	r v b	r v b	r v b	v b	r v b



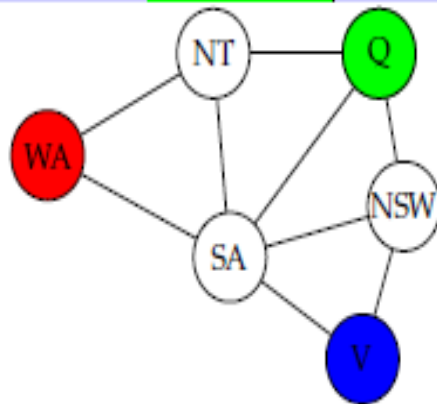
	WA	NT	Q	NSW	V	SA	T
Domaines initiaux	r v b	r v b	r v b	r v b	r v b	r v b	r v b
Affectation WA=r	r	v b	r v b	r v b	r v b	v b	r v b
Affectation Q=v	r	b	v	r b	r v b	b	r v b



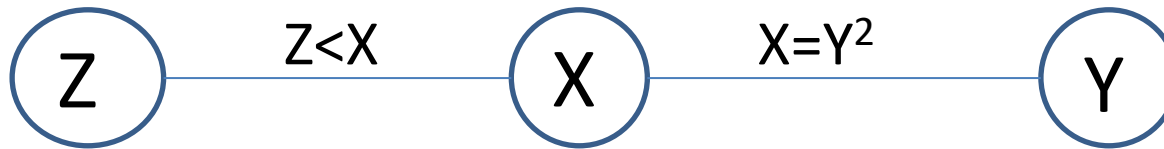
	WA	NT	Q	NSW	V	SA	T
Domaines initiaux	r v b	r v b	r v b	r v b	r v b	r v b	r v b
Affectation WA=r	r	v b	r v b	r v b	r v b	v b	r v b
Affectation Q=v	r	b	v	r b	r v b	b	r v b
Affectation V=b	r	b	v	r	b		r v b



	WA	NT	Q	NSW	V	SA	T
Domaines initiaux	r v b	r v b	r v b	r v b	r v b	r v b	r v b
Affectation WA=r	r	v b	r v b	r v b	r v b	v b	r v b
Affectation Q=v	r	b	v	r b	r v b	b	r v b
Affectation V=b	r	b	v	r	b		r v b



Certaines inconsistances ne sont pas détectées par forward checking
(ligne 2, NT et SA n'ont plus qu'une valeur disponible alors qu'ils sont voisins !)



Solution partielle	DX	DY	DZ	
{}	{0,1,2,3,4,5,6,7,8,9}	{0,1,2,3,4,5,6,7,8,9}	{0,1,2,3,4,5,6,7,8,9}	
{X=0}	{0}	{0}	{}	Retour en arrière
{X=1}	{1}	{1}	{0}	
{X=1,Y=1}	{1}	{1}	{0}	
{X=1,Y=1,Z=0}	{1}	{1}	{0}	

Gestion de contraintes spécifiques

Contrainte *toutes différentes*

- Toutes les variables doivent avoir des valeurs distinctes.
- S'il y a m variables, n valeurs possibles et $m > n$, alors les contraintes ne peuvent pas être satisfaites.
- Algorithme:
 - Enlever toutes les variables qui n'ont qu'une valeur dans leur domaine et enlever cette valeur de toutes les autres variables restantes.
 - Continuer tant qu'il y a des variables avec un domaine contenant qu'une seule valeur.
 - Si un domaine vide est produit, ou qu'il y a plus de variables que de valeurs disponibles, alors une incohérence a été détectée.

Exemple

	1	2	3	4	5	6	7	8	9
A			3		2		6		
B	9			3		5			1
C			1	8		6	4		
D			8	1		2	9		
E	7								8
F			6	7		8	2		
G			2	6		9	5		
H	8			2		3			9
I			5		1		3		

(a)

On va étudier les variables: E6, I6, A6, ...

Avec cet exemple, on peut résoudre entièrement le sudoku sans faire de recherche !

	1	2	3	4	5	6	7	8	9
A	4	8	3	9	2	1	6	5	7
B	9	6	7	3	4	5	8	2	1
C	2	5	1	8	7	6	4	9	3
D	5	4	8	1	3	2	9	7	6
E	7	2	9	5	6	4	1	3	8
F	1	3	6	7	9	8	2	4	5
G	3	7	2	6	8	9	5	1	4
H	8	1	4	2	5	3	7	6	9
I	6	9	5	4	1	7	3	8	2

(b)

Autres contraintes spécifiques

Il existe d'autres types de contraintes globales

- contraintes sur les ressources :
 - ex : au Plus on peut vouloir dire que la somme des valeurs des variables A, B, C et D soit 10. Si les domaines sont $\{2, 3, 4, 5, 6\}$ on peut enlever 5 et 6 de chaque domaine !
- contraintes avec des bornes:
 - ex : si $A+B = 420$ et $DA = [0, 165]$, $DB = [0, 385]$, on peut réduire le domaine à $DA = [35, 165]$, $DB = [255, 385]$

Une approche différente : la recherche locale

Au lieu de construire une solution en affectant une à une les variables, on peut partir d'une affectation complète (qui n'est pas cohérente) et essayer de modifier l'affectation.

- Quelle variable corriger ? Elle peut être choisie au hasard
- Quelle valeur choisir ? heuristique du minimum de conflit : choisir la valeur qui a le minimum de conflits avec les autres variables.

