

Problèmes de satisfaction de contraintes

- + Les chapitres 3 (« Algorithmes de recherche en IA ») et 4 (« Algorithmes et recherches heuristiques ») ont montré que les problèmes pouvaient être résolus par l'exploration d'un espace d'états.
- + Ces états peuvent être évalués par des heuristiques spécifiques d'un domaine et testés afin de déterminer s'il s'agit d'états buts.
- + Toutefois, du point de vue des algorithmes d'exploration, chaque état est atomique, c'est-à-dire indivisible - c'est une boîte noire sans structure interne.

Ce chapitre décrit un moyen de résoudre plus efficacement toute une gamme de problèmes.

- + Nous utilisons une représentation factorisée pour chaque état : un ensemble de variables, dont chacune a une valeur.
- + Un problème est résolu quand chaque variable a une valeur qui satisfait à toutes ses contraintes.
- + Un problème ainsi décrit est appelé un problème à satisfaction de contraintes, ou CSP (*Constraint Satisfaction Problem*).
- + Les algorithmes d'exploration CSP tirent parti de la structure des états et utilisent des heuristiques générales plutôt que spécifiques pour résoudre des problèmes complexes.
- + La principale idée ici est d'éliminer d'un coup de grandes parties de l'espace de recherche en trouvant les combinaisons variable / valeur qui violent les contraintes.

Problèmes de satisfaction de contraintes (CSP)

- Problèmes de recherche “classiques” :
 - Un état est une “boîte noire”
 - N’importe quelle structure de données qui contient un test pour le but, une fonction d’évaluation, une fonction successeur
- CSP :
 - Un état est défini par un ensemble de variables X_i , dont les valeurs appartiennent au domaine D_i
 - Le test pour le but est un ensemble de contraintes qui spécifient les combinaisons autorisées pour les valeurs sur des sous-ensembles de variables
- Exemple simple d’un langage formel de représentation
- Permet d’utiliser des algorithmes généraux plus efficaces que les algorithmes de recherche standards

Définition des problèmes à satisfaction de contraintes

Un problème à satisfaction de contraintes à trois composantes X , D , et C :

- + X est un ensemble de variables, $\{X_1, \dots, X_n\}$.
 - + D est un ensemble de domaines, $\{D_1, \dots, D_n\}$, un pour chaque variable.
 - + C est un ensemble de contraintes qui spécifient les combinaisons admissibles de valeurs.
-
- + Chaque domaine D_i est constitué d'un ensemble de valeurs admissibles $\{v_1, \dots, v_k\}$ pour la variable X_i
 - + Chaque contrainte C_i est constituée d'une paire de $\langle \text{portée}, \text{rel} \rangle$, où **portée** est un n-uplet de variables qui participent à la contrainte et **rel** est une relation qui définit les valeurs que ces variables peuvent prendre.
 - + Une relation peut être représentée comme une liste explicite de tous les n-uplets de valeurs qui satisfont la contrainte,
 - + ou comme une relation abstraite sur laquelle s'appliquent deux opérations : tester si un n-uplet est un membre de la relation, et énumérer les membres de la relation.

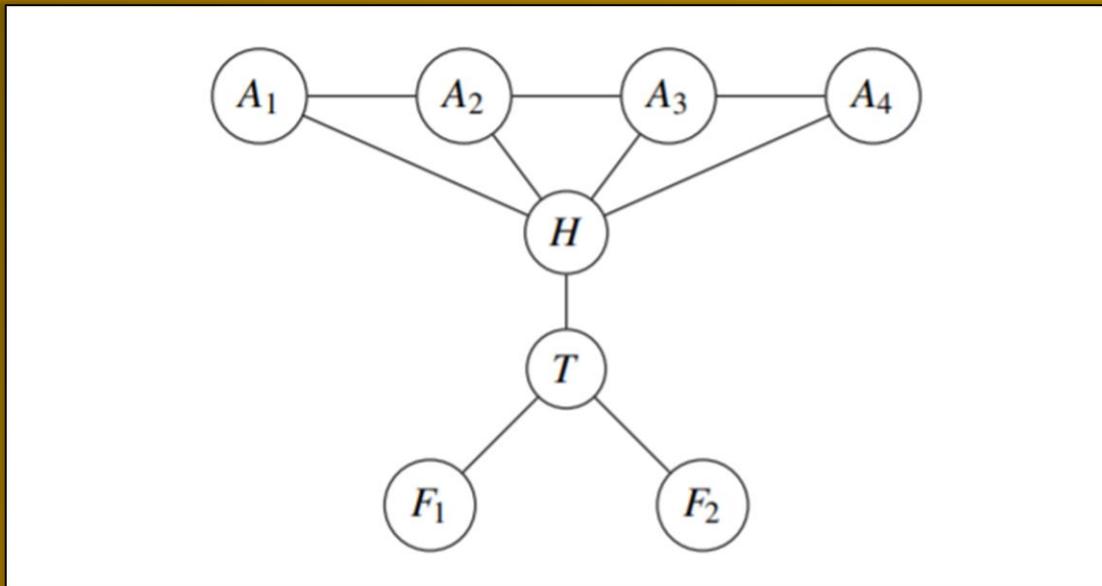
+ Par exemple, si X_1 et X_2 ont tous les deux le domaine $\{A, B\}$, la contrainte qui dicte que les deux variables doivent avoir des valeurs différentes peut s'écrire $\langle(X_1, X_2), [(A, B), (B, A)] \rangle$ ou $\langle(X_1, X_2), X_1 \neq X_2 \rangle$

- + Pour résoudre un CSP on doit définir un espace d'états et ce qu'est précisément une « solution ».
- + Chaque état du CSP est défini par une **assignation** (*attribution*) de valeurs à certaines variables ou à toutes les variables, $\{X_i = v_i, X_j = v_j, \dots\}$.
- + Une assignation qui ne viole aucune contrainte est dite **cohérente** ou légale.
- + Une **assignation complète** est une assignation dans laquelle chaque variable a une assignation, et une **solution** d'un CSP est une assignation cohérente et complète.
- + Une **assignation partielle** est une assignation qui n'attribue des valeurs qu'à certaines variables seulement.

Exercice 1

Trouvez un coloriage à 3 couleurs de ce graphe en appliquant la recherche par backtrack avec recherche en avant, heuristique MRV et heuristique du degré.

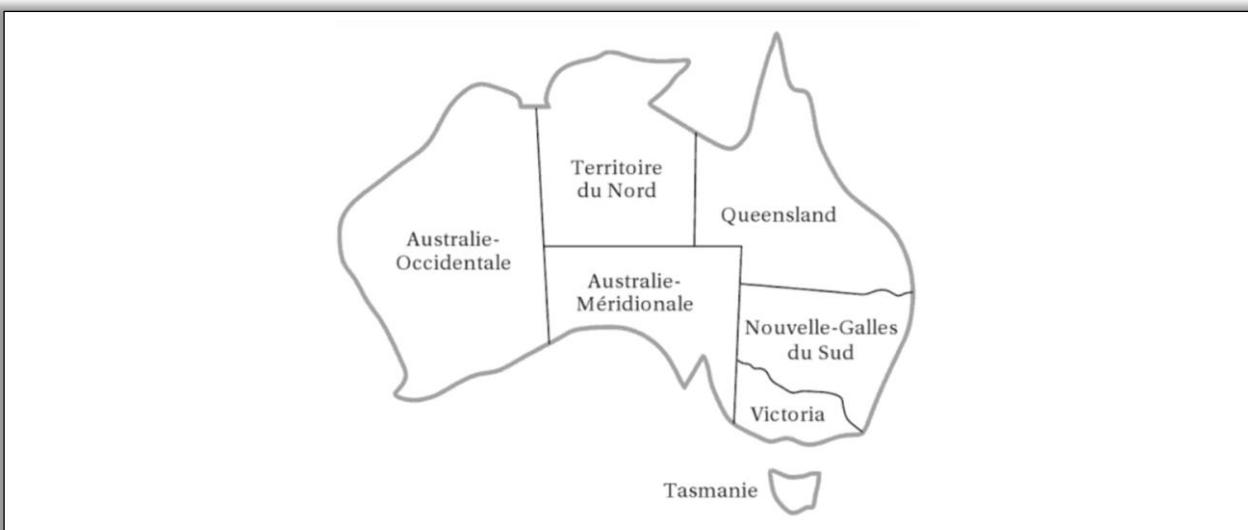
Si vous avez le choix entre plusieurs variables, vous choisirez en suivant l'ordre alphanumérique. Vous appliquerez les couleurs en respectant l'ordre {R, V, B}.



le graphe de contraintes

Coloration d'un plan

Nous nous intéressons à une carte administrative de l'Australie représentant les États et les territoires de ce pays et que nous souhaitons colorer chaque région en rouge, vert ou bleu, de telle manière qu'aucune région n'ait la même couleur qu'une de ses voisines.



Un problème à satisfaction de contraintes à trois composantes X, D, et C :

⊕ X est un ensemble de variables, $\{X_1, \dots, X_n\}$.

Variables: AO, TN, Q, NGS, V, AM, T

⊕ D est un ensemble de domaines, $\{D_1, \dots, D_n\}$, un pour chaque variable.

Domaines : $D_i = \{\text{rouge, vert, bleu}\}$

⊕ C est un ensemble de contraintes qui spécifient les combinaisons admissibles de valeurs.

Étant donné qu'il y a neuf endroits où des régions se touchent, il y a neuf contraintes :

$$C = \{AM \neq AO, AM \neq TN, AM \neq Q, AM \neq NGS, AM \neq V, AO \neq TN, TN \neq Q, Q \neq NGS, NGS \neq V\}.$$

Ici, nous utilisons des abréviations : $AM \neq AO$ est une abréviation de $((AM, AO), AM \neq AO)$, où $AM \neq AO$ peut être explicité en :

$\{(rouge, vert), (rouge, bleu), (vert, rouge), (vert, bleu), (bleu, rouge), (bleu, vert)\}$.

Il existe de nombreuses solutions à ce problème, dont la suivante :

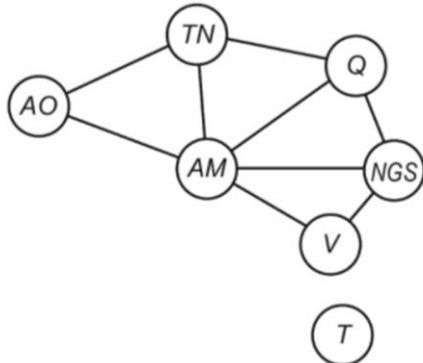
Les solutions sont des affectations qui satisfont toutes les contraintes.

$\{AO = \text{rouge}, TN = \text{vert}, Q = \text{rouge}, NGS = \text{vert}, V = \text{rouge}, AM = \text{bleu}, T = \text{rouge}\}$



Graphe de contraintes

Il peut être utile de visualiser un CSP sous la forme d'un **graphe de contraintes**, ce qui apporte une aide appréciable.



Les nœuds du graphe correspondent à des variables du problème et les arcs relient des paires de variables qui participent à une contrainte.

Pourquoi formuler un problème sous forme de CSP ?

- ⊕ Une raison est que les CSP donnent une représentation naturelle d'une grande variété de problèmes;
- ⊕ Si on dispose déjà d'un système de résolution de CSP il est souvent plus facile de s'en servir pour résoudre un problème plutôt que de concevoir une solution spécifique avec une autre technique d'exploration.
- ⊕ De plus, les résolveurs de CSP peuvent être plus rapides que les systèmes d'exploration d'espaces d'états parce qu'ils peuvent éliminer de larges pans de l'espace d'exploration.
- ⊕ **Par exemple**, si nous avons choisi {AM = bleu} dans le problème de la carte de l'Australie, nous pouvons conclure qu'aucune des cinq variables voisines ne peut prendre la valeur bleu.
- ⊕ En profitant de la propagation de contraintes, une procédure d'exploration aurait à examiner $3^5 = 243$ assignations pour les cinq variables voisines;
- ⊕ Alors qu'avec la propagation de contraintes nous n'avons jamais à examiner bleu comme valeur, et donc nous n'avons à tester que $2^5 = 32$ assignations, soit une réduction de 87 %.

Graphe de contraintes

- **CSP binaires** : chaque contrainte lie au maximum deux variables
- **Graphe de contraintes** : les nœuds sont des variables, les arcs représentent les contraintes
- Les algorithmes CSP utilisent les graphes de contraintes
- Permet d'accélérer la recherche : par exemple, colorier la Tasmanie est un sous-problème indépendant

- ⊕ Dans l'exploration standard des espaces d'états, nous pouvons seulement nous demander : Est-ce que cet état est un but ? Non ? Et celui-là ?
- ⊕ Avec les CSP une fois que nous savons qu'une assignation partielle enfreint certaines contraintes, nous pouvons immédiatement rejeter en bloc toute assignation subordonnée.
- ⊕ De plus, nous comprenons pourquoi l'assignation n'est pas une solution - nous voyons les variables qui violent une contrainte - et nous pouvons nous concentrer sur les variables importantes.
- ⊕ Ainsi, beaucoup de problèmes impossibles à traiter par une exploration standard des espaces d'états peuvent être résolus rapidement s'ils sont formulés sous forme de CSP.

Exploration avec backtracking pour les CSP

Recherche en arrière pour les CSPs

Le terme d'exploration avec backtracking désigne une exploration en profondeur d'abord (DFS) qui choisit des valeurs pour une variable à la fois et remonte dans l'arbre (backtracks) lorsqu'on ne peut plus assigner de valeur légale à une variable.

Comme la représentation des CSP est normalisée, il n'y a pas besoin de fournir à EXPLORATION-BACKTRACKING un état initial, une fonction d'action, un modèle de transition ni un test de but qui soient propres au domaine.

EXPLORATION-BACKTRACKING ne garde qu'une seule représentation d'un état et la modifie plutôt que d'en créer de nouvelles.

fonction EXPLORATION-BACKTRACKING(*csp*) **retourne** une solution, ou échec
retourner BACKTRACK({}, *csp*)

fonction BACKTRACK(*assignation, csp*) **retourne** une solution, ou échec

si *assignation* est complète **alors retourner** *assignation*

var \leftarrow SÉLECTIONNER-VARIABLE-NON-ASSIGNÉE(*csp*)

pour chaque *valeur* **dans** ORDONNER-VALEURS-DOMAINE(*var, assignation, csp*) **faire**

si *valeur* est cohérente avec *assignation* **alors**

ajouter {*var = valeur*} à *assignation*

inférences \leftarrow INFÉRENCE(*csp, var, valeur*)

si *inférences* \neq échec **alors**

ajouter *inférences* à *assignation*

résultat \leftarrow BACKTRACK(*assignation, csp*)

si *résultat* \neq échec **alors**

retourner *résultat*

enlever {*var = valeur*} et *inférences* de *assignation*

retourner échec

Il choisit à répétition une variable non assignée, puis essaie successivement toutes les valeurs du domaine de cette variable pour trouver une solution.

La fonction INFÉRENCE peut optionnellement être utilisée pour imposer une cohérence d'arc, une cohérence de chemin ou une k-cohérence par exemple.

Si une valeur mène à un échec (détecté par INFÉRENCE ou par BACKTRACK), les assignations de valeur (dont celles effectuées par INFÉRENCE) sont retirées de l'assignation actuelle et un nouvel essai est tenté avec une nouvelle valeur.

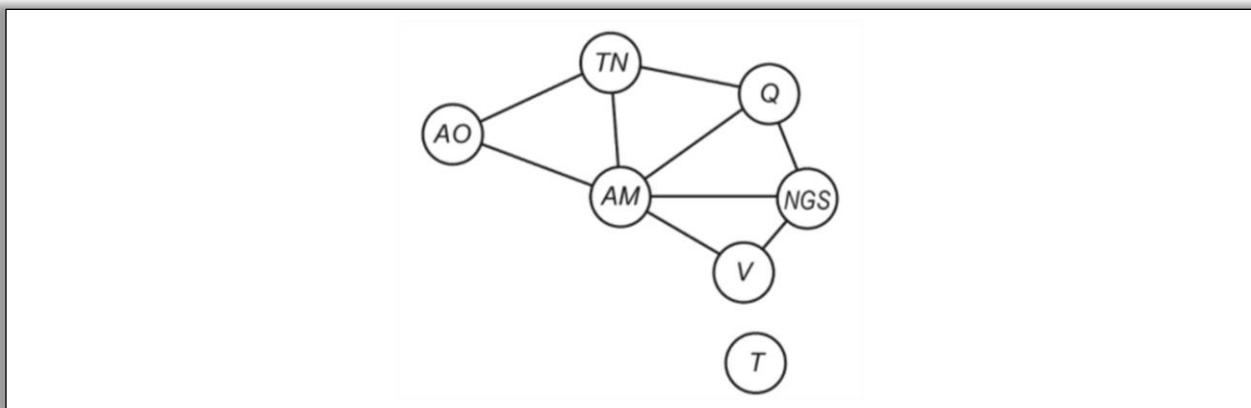
En faisant varier les fonctions SÉLECTIONNER-VARIABLE-NON-ASSIGNÉE et ORDONNER-VALEURS-DOMAINE, on peut implémenter les heuristiques génériques présentées dans le texte.

Si une incohérence est détectée, BACKTRACK renvoie un échec, le précédent appel essaie une autre valeur.

- Cette section : algorithmes d'exploration avec backtracking qui fonctionnent sur des **assignments partielles**.
- Un état serait **une assignation partielle** et une action consisterait en l'addition de var = valeur à l'assignation.

- **Une propriété cruciale commune à tous les CSP :**
La commutativité. Un problème est commutatif si l'ordre d'application d'un ensemble d'actions donné n'affecte pas le résultat.
- C'est le cas des CSP : en effet, quand on assigne des valeurs à des variables, on obtient la même assignation partielle, quel que soit l'ordre.
- **En conséquence**, on n'a à considérer que les assignations possibles pour une seule variable de chaque nœud dans l'arbre de recherche.

- **Par exemple**, au nœud racine d'un arbre de recherche construit en vue de colorer la carte de l'Australie,



- On peut choisir entre AM = rouge, AM = vert et AM = bleu, mais jamais entre AM = rouge et AO = bleu.
- Compte tenu de cette restriction, le nombre de feuilles est d^n , comme on l'espérait.

Backtracking search

- L'affectation des variables est **commutative**
 - L'ordre dans lequel on affecte les variables n'a pas d'importance
 - WA = rouge puis NT = vert est la même chose que NT = vert puis WA = rouge
- Il n'y a donc besoin de ne considérer **qu'une seule variable** par profondeur de l'arbre de recherche
 $\rightarrow b = d$, et donc d^n feuilles
- Recherche en profondeur d'abord avec l'affectation d'une variable à la fois est appelée **recherche par retour arrière** (**backtracking search**)
- Algorithme de recherche basique pour les CSPs
- Permet de résoudre le problème des n reines pour $n \sim 25$

Améliorer l'efficacité du la recherche par backtrack (1)

- ⊕ Au chapitre 3 (« Algorithmes de recherche en IA ») , nous avons amélioré les médiocres performances des algorithmes d'exploration non informée en leur fournissant des fonctions heuristiques spécifiques d'un domaine et issues de notre connaissance du problème.
- ⊕ Il s'avère que l'on peut résoudre efficacement des CSP sans connaissances spécialisées.
- ⊕ À la place, nous pouvons améliorer les fonctions non spécifiées à la figure 6.5 :

```
fonction EXPLORATION-BACKTRACKING(csp) retourne une solution, ou échec
    retourner BACKTRACK({}, csp)

fonction BACKTRACK(assignation, csp) retourne une solution, ou échec
    si assignation est complète alors retourner assignation
    var ← SÉLECTIONNER-VARIABLE-NON-ASSIGNÉE(csp)
    pour chaque valeur dans ORDONNER-VALEURS-DOMAINE(var, assignation, csp) faire
        si valeur est cohérente avec assignation alors
            ajouter {var = valeur} à assignation
            inférences ← INFÉRENCE(csp, var, valeur)
            si inférences ≠ échec alors
                ajouter inférences à assignation
                résultat ← BACKTRACK(assignation, csp)
                si résultat ≠ échec alors
                    retourner résultat
                enlever {var = valeur} et inférences de assignation
            retourner échec
```

Comment choisir la variable à affecter ensuite ? (Select-Unassigned-Variable)

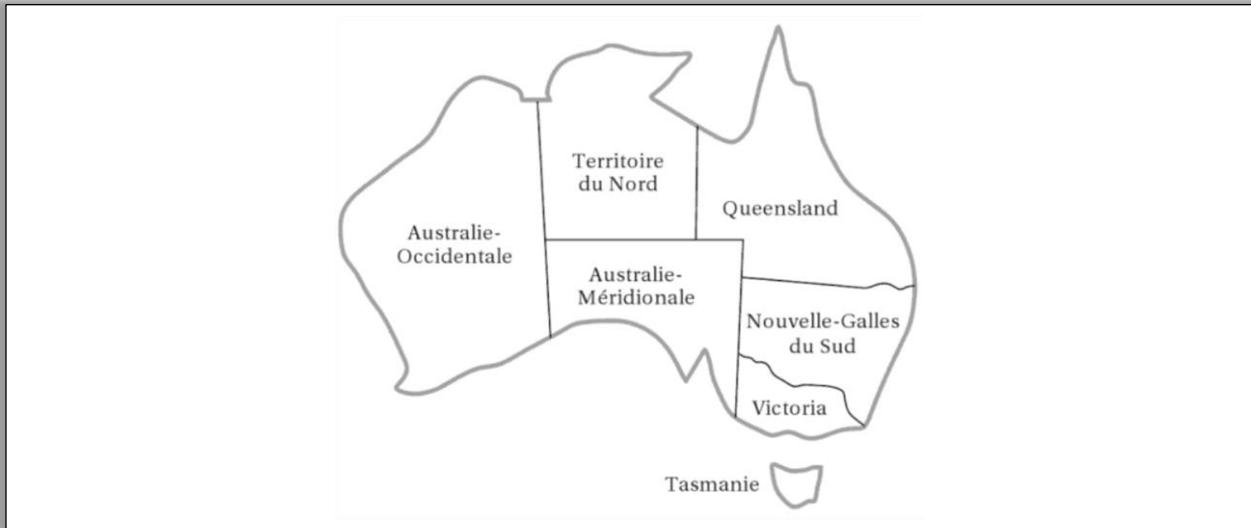
L'algorithme de backtracking contient la ligne

***var* ← SÉLECTIONNER-VARIABLE-NON-ASSIGNÉE(*CSD*) .**

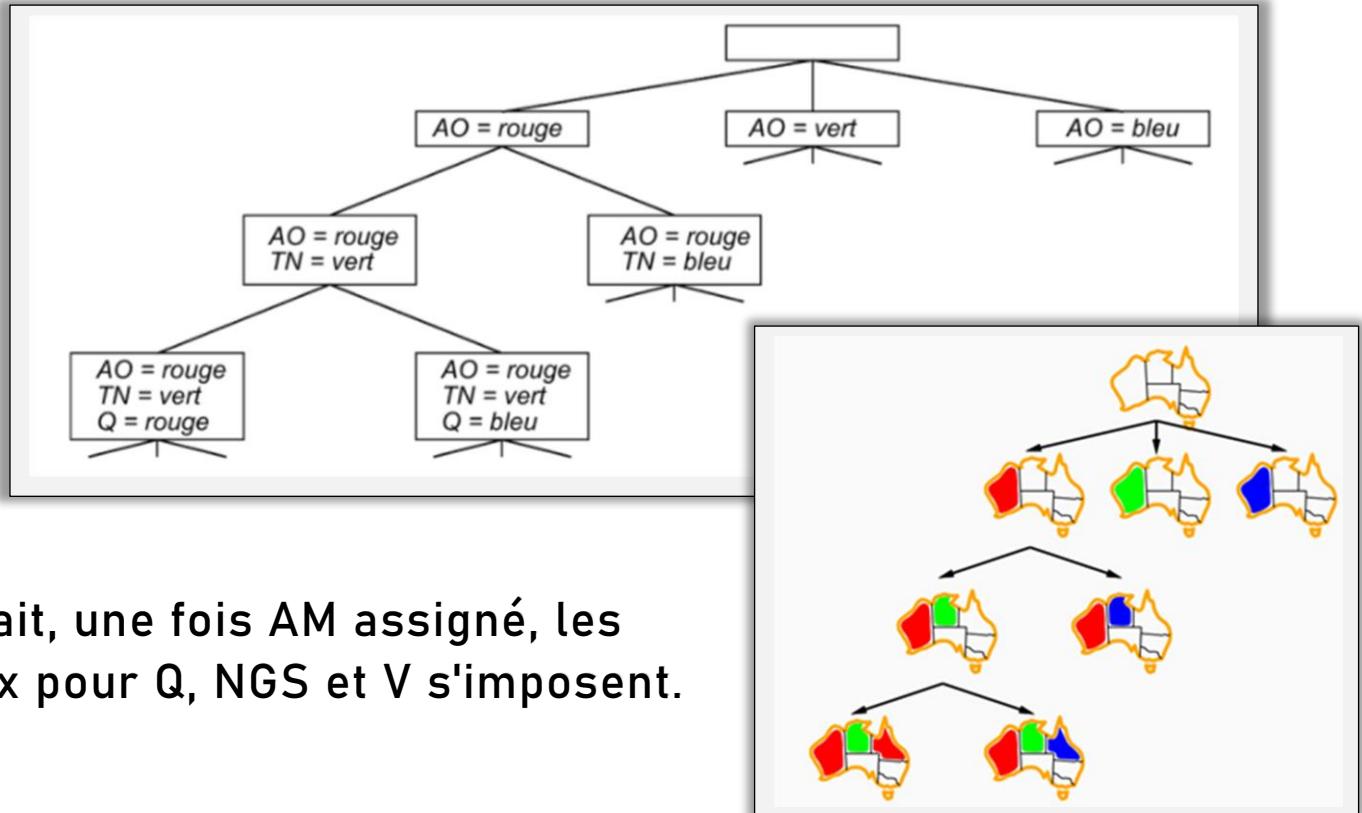
La stratégie la plus simple pour SÉLECTIONNER-VARIABLE-NON-ASSIGNÉE se contente de sélectionner la variable non assignée suivante dans l'ordre {X1, X2,...}.

Cet ordre statique des variables débouche rarement sur l'exploration la plus efficace.

Par exemple,



après les assignations $AO = \text{rouge}$ et $TN = \text{vert}$, il ne reste plus qu'une valeur possible pour AM . Il est donc raisonnable d'assigner ensuite $AM = \text{bleu}$, plutôt que d'assigner Q .



Heuristique des valeurs minimum restantes (MRV)

- ⊕ Cette idée intuitive (le choix de la variable dont les valeurs «légales» sont les moins nombreuses) est appelée **l'heuristique du minimum de valeurs restantes**, ou **MRV (Minimum Remaining Values)**.
- ⊕ On la nomme également «**heuristique de la variable la plus contrainte**» ou heuristique fail-first (maillon faible), car on sélectionne la variable qui risque le plus de provoquer un échec, ce qui contribue à élagger l'arbre de recherche.
- ⊕ Si pour une variable X, il ne reste plus de valeurs légales, l'heuristique MRV sélectionnera W et l'échec sera immédiatement détecté, ce qui évitera des recherches inutiles à partir des autres variables.
- ⊕ L'heuristique MRV est en général plus efficace qu'un classement aléatoire ou statique, parfois d'un facteur de 1000 ou plus, mais le résultat varie beaucoup en fonction du problème.
- ⊕ L'heuristique MRV n'est d'aucun secours quant au choix de la première région à colorer dans la carte de l'Australie : au début, les trois couleurs sont légales pour chaque région.

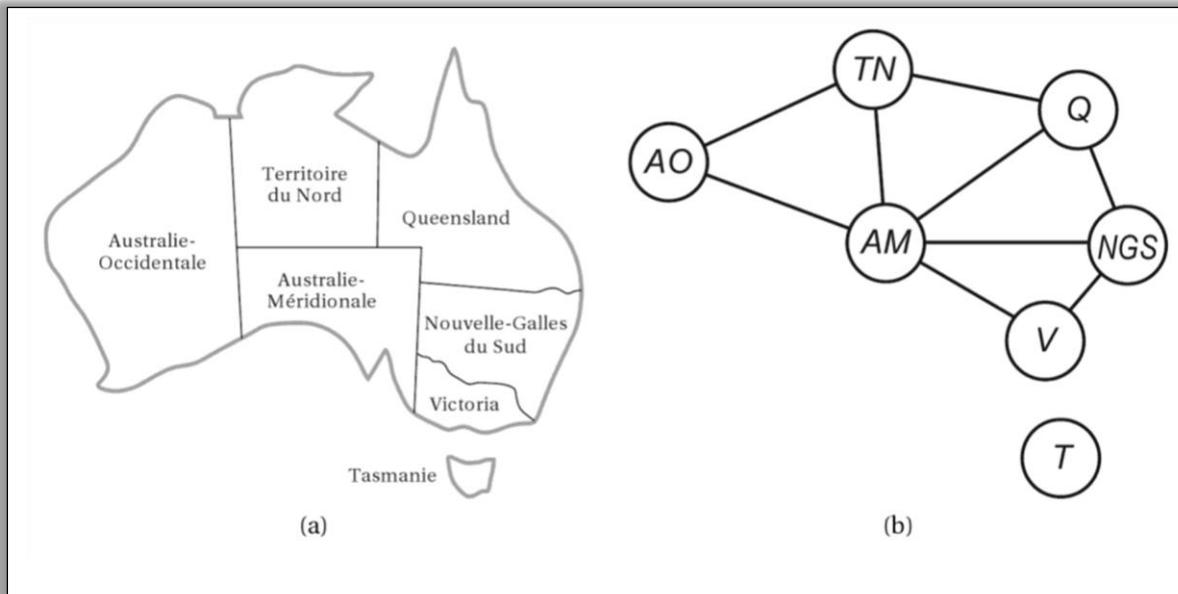
• Heuristique des valeurs minimum restantes (MRV)

⇒ choisir une des variables ayant le moins de valeur "légale" possible



Heuristique du degré

- ⊕ Dans ce cas, on peut recourir à **l'heuristique des degrés**, qui tente de réduire le facteur de branchement des choix futurs en sélectionnant la variable impliquée dans le plus grand nombre de contraintes sur les autres variables non assignées
- ⊕ À la figure 6.1, AM est la variable de degré le plus élevé : 5. Les autres variables ont les degrés 2 ou 3, à l'exception de T dont le degré est 0. En fait, dès lors que AM est choisi, l'application de l'heuristique des degrés résout le problème sans faux pas : on peut choisir n'importe quelle couleur cohérente à chaque point de décision et parvenir à une solution, sans jamais recourir au backtracking.



- Si plusieurs variables ne peuvent pas être départagées par l'heuristique MRV
- **Heuristique du degré**
⇒ choisir la variable qui a le plus de contraintes à respecter parmi les variables restantes



- ⊕ L'heuristique MRV est habituellement un guide plus sûr, mais l'heuristique des degrés peut être précieuse pour départager des options que ne distingue pas MRV.

Améliorer l'efficacité du la recherche par backtrack (2)

Est-il possible de détecter un échec inévitable plus tôt ?

Association de l'exploration et de l'inférence

- ✚ L'**inférence** peut être puissante au cours d'une exploration : chaque fois que nous choisissons une valeur pour une variable, nous avons une occasion d'inférer de nouvelles réductions de domaine sur les variables voisines.
- ✚ Une des formes les plus simples d'**inférence** est appelée **forward checking**, ou « vérification en avant ».
- ✚ Chaque fois qu'une variable X est assignée, elle détermine sa cohérence d'arc : pour chaque variable non assignée Y qui est reliée à X par une contrainte, elle supprime du domaine de Y toutes les valeurs non cohérentes avec la valeur choisie pour X.
- ✚ Comme le forward checking n'effectue que des inférences sur la cohérence d'arc, il n'y a aucune raison de l'appliquer si l'on a déjà examiné la cohérence d'arc lors d'un prétraitement.
- ✚ La figure 6.7 montre la progression en backtracking de l'exploration entreprise pour la coloration de la carte de l'Australie avec activation du forward checking,

	AO	NT	Q	NGS	V	AM	T
Domaines initiaux	R V B	R V B	R V B	R V B	R V B	R V B	R V B
Après AO = rouge	(R)	V B	R V B	R V B	R V B	V B	R V B
Après Q = vert	(R)	B	(V)	R B	R V B	B	R V B
Après V = bleu	(R)	B	(V)	R	(B)		R V B

Figure 6.7 : Progression d'une recherche de coloration de la carte avec forward checking. AO = rouge est assigné en premier. Ensuite, le forward checking supprime rouge des domaines des variables voisines TN et AM. Après Q = vert, vert est supprimé des domaines de TN, AM et NGS. Après l'assignation de V = bleu, bleu est supprimé des domaines de NGS et AM, ce qui laisse AM sans valeur légale.

⊕ Tout d'abord, remarquez ceci: après qu'on a assigné $AO = \text{rouge}$ et $Q = \text{vert}$, les domaines de TN et de AM sont réduits à une seule valeur.

⊕ Pour de nombreux problèmes, l'exploration sera plus efficace si nous combinons l'heuristique MRV avec le forward checking.

⊕ Considérons la figure 6.7 après l'attribution $\{AO = \text{rouge}\}$.

Intuitivement, il semble que l'assignation contraine ses voisins, TN et AM , donc nous devrions traiter ces variables ; toutes les autres variables trouveront leur place.

⊕ C'est exactement ce qui se passe avec MRV : TN et AM ont deux valeurs : on choisit une en premier, puis l'autre, puis Q , NGS et V , dans cet ordre.

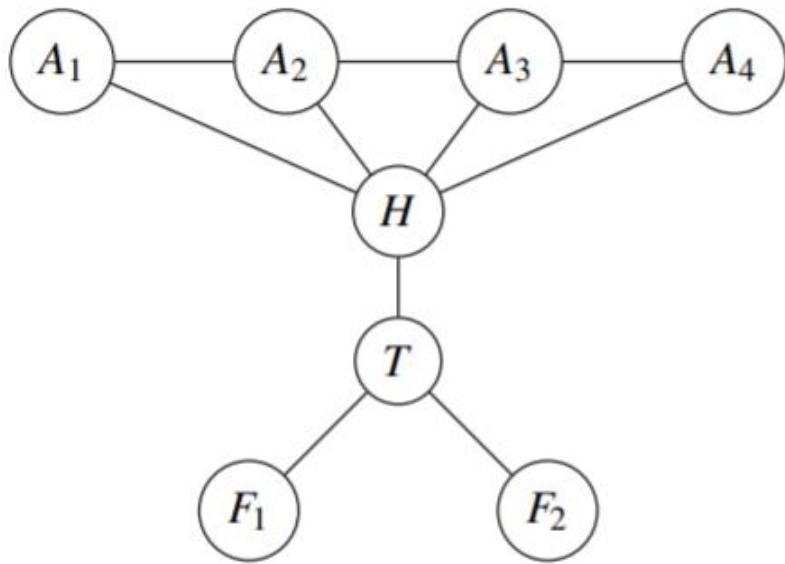
⊕ Enfin T a encore trois valeurs, toutes recevables.

⊕ On peut considérer le forward checking comme une manière efficace de calculer incrémentalement l'information dont a besoin l'heuristique MRV pour fonctionner.

⊕ Le forward checking détecte de nombreuses incohérences, mais pas toutes. Le problème réside en ce qu'il rend la variable courante arc-cohérente mais qu'il n'essaie pas de faire pareil avec les autres.

⊕ Examinez par exemple la troisième ligne de la figure 6.7 :

lorsque AO est en rouge et que Q est en vert, TN et AM doivent être en bleu. Forward checking ne va pas assez loin pour remarquer qu'il y a un problème : TN et AM sont côté à côté et ne peuvent pas avoir la même valeur.



Premier chose à faire : bien définir le problème :

Un problème à satisfaction de contraintes à trois composantes X, D, et C :

- X est un ensemble de variables, $\{X_1, \dots, X_n\}$.

Variables: AO, TN, Q, NGS, V, AM, T

- D est un ensemble de domaines, $\{D_1, \dots, D_n\}$, un pour chaque variable.

Domaines : $D_i = \{\text{rouge, vert, bleu}\}$

- C est un ensemble de contraintes qui spécifient les combinaisons admissibles de valeurs.

Etant donné qu'il y a neuf endroits où des régions se touchent, il y a neuf contraintes :

$$C = \{AM \neq AO, AM \neq TN, AM \neq Q, AM \neq NGS, AM \neq V, AO \neq TN, TN \neq Q, Q \neq NGS, NGS \neq V\}.$$

Ici, nous utilisons des abréviations : $AM \neq AO$ est une abréviation de $((AM, AO), AM \neq AO)$, où $AM \neq AO$ peut être explicité en :

$\{(\text{rouge, vert}), (\text{rouge, bleu}), (\text{vert, rouge}), (\text{vert, bleu}), (\text{bleu, rouge}), (\text{bleu, vert})\}$.

Exemple

Même SI LE DOMAINE est pareil il est important de l'écrire plusieurs fois pour la suite.

Variables	Domaines des variables	Degréss
A1	R, V, G	➤ $A_1 = 2$ (Combien de contraintes a la variables A1 ?)
A2	R, V, G	➤ $A_2 = 3$
A3	R, V, G	➤ $A_3 = 3$
A4	R, V, G	➤ $A_4 = 2$
H	R, V, G	➤ $H = 5$
T	R, V, G	➤ $T = 3$
F1	R, V, G	➤ $F_1 = 1$
F2	R, V, G	➤ $F_2 = 1$

Maintenant on a tous ce qui faut pour appliquer l'algorithme.

- On commence par une affectation qui contient seulement l'ensemble vide...
- Ensuite quelle va être la première variable à affecter ?
- MRV (heuristique qui cherche la variable qui a le moins de valeurs restantes, c'est ce qu'on fait tjr en premier : y a une variable qui a moins de valeurs dispo ?)
- Quand on a plusieurs choix possibles avec MRV on applique l'heuristique de degré (la variable qui a le plus de contraintes à respecter)

1/ heuristique MRV : variable qui a le moins de valeur restante
 2/ heuristique Degré : variable qui a le plus de contrainte à satisfaire
 3/ si encore le choix entre plusieurs variable : ordre alphanumérique

MRV : ne discrimine pas. Degré : H

{}

Dapres MRV , on a le choix entre quoi et quoi ?

MRV: A1 et T
Degré : A2, A3, T
alpha : A2

H = R

H = V

H = B

MRV : A1 et A3
degré : A3

A₂ = V

A₂ = B

A₃ = B

MRV : A1, A4
DEGRE : A1, A4
ALPHA (ALPHANUMERIQUE): A1

MRV : A4

MRV : T

MRV : F1, F2
degre : F1, F2
Alpha : F1

MRV : F2

T = V

T = B

F₁ = R

F₂ = B

Exercice 2

Résolvez le puzzle cryptarithmétique suivant en utilisant la recherche par backtrack avec recherche en avant, propagation des contraintes, heuristique MRV et heuristique du degré.

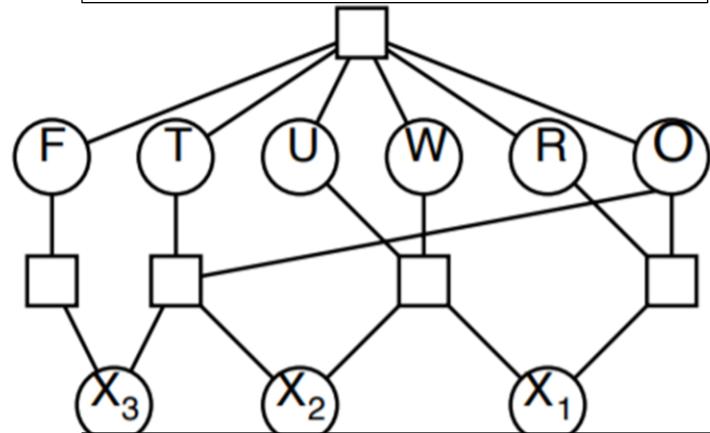
Si vous avez le choix entre plusieurs variables, vous choisirez en suivant l'ordre alphanumérique.
Si vous avez le choix entre plusieurs valeurs, vous choisirez la plus petite.

Objectif : résoudre cette addition

$$\begin{array}{r} \text{T} \ \text{W} \ \text{O} \\ + \ \text{T} \ \text{W} \ \text{O} \\ \hline \text{F} \ \text{O} \ \text{U} \ \text{R} \end{array}$$

T et F peuvent pas être égal à 0 !

Le graphe de contraintes
(contraintes pas forcément binaire)



X1,X2,X3 : LES retenue DE LADDITION

Variables etape1	Domaines des variables etape2	Degrés des variables etape 4	Contraintes entre les variables etape 3
X1	{0,1}	2	Dans le schéma y a 5 carrée - → 5 contraintes à définir : AllDiff(F,T,U,W,R,O) F ≠ O ≠ U ≠ R ≠ T ≠ W
X2	{0,1}	2	O + O = R + 10 * x1
X3	{0,1}	2	Quand on fait une addition si O est égal à 7 on a 7+7 = 14 = 4 + 10* 1 donc on a une retenue de 1
F	{1,2,3,4,5,6,7,8,9}	2	W + W + X1 = U + 10 * X2
T	{1,2,3,4,5,6,7,8,9}	2	T + T + X2 = O + 10 * X3
U	{0,1,2,3,4,5,6,7,8,9}	2	F = X3
W	{0,1,2,3,4,5,6,7,8,9}	2	chaque contrainte correspond à un carré sur le graph des contraintes
R	{0,1,2,3,4,5,6,7,8,9}	2	
O	{0,1,2,3,4,5,6,7,8,9}	3	

Ce qu'on a en plus par rapport à l'exo de tt à l'heure c'est la propagation des contrariantes :

propagation des contraintes : pré-processing

contrainte 2 : $O + O = R + 10 * x_1$

On peut réduire le domaine d'une variable juste avec la contrainte ?

Quel que soit la valeur de O , R peut avoir n'importe quelle valeur ?

R est forcément un nombre paire vu qu'on additionne 2 chiffres identiques

Donc : contrainte 2 $\rightarrow R$ paire

La contrainte 2 me permet de déduire autre chose ?

Quelque il se passe si O est égal à 0 ?

d'après la contrainte 1, O ne peut pas être égal à 0 .

R peut être égal à 0 ? oui..

contrainte 3 : rien à déduire

Contrainte 5 me permet de déduire :

puisque $F = X_3$ et F est différent de O , $X_3 = 1 = F$

On peut déduire autre chose ?

on peut déduire que les autres ne peuvent pas être 1 (contrainte 1) (Sauf les X)

Dernière déduction à partir de la **contrainte 4** : si $X_3 = 1$ Quelque on sait ?

T supérieur ou égale à 5. Comme O est différent de 0 , T est même strictement supérieur à 5.

Contraintes :

- 1/ AllDiff(F, T, U, W, R, O)
- 2/ $O+O = R + 10*x_1$
- 3/ $W+W+x_1 = U + 10*x_2$
- 4/ $T+T+x_2 = O+10*x_3$
- 5/ $F = X_3$

Propagation des contraintes : pré-processing

- Contrainte 2 : R pair
- Contraintes 1 et 2 : si $O=0$, $R=0$
- Contrainte 5 : $F=X_3$ et $F \neq 0$, $X_3 = 1 = F$
- Contrainte 1 : tous différents de 1 (sauf les X)
- Contrainte 4 : $X_3 = 1$ et $O \neq 0$, donc $T > 5$

On commence l'algorithme de recherche

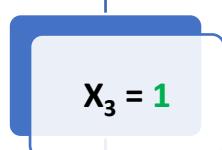
On commence l'algorithme de recherche avec une affectation vide



MRV : F, X3
Degré = F, x3
ALPHA : f



MRV : X3



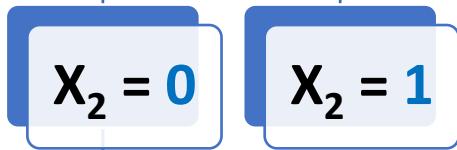
MRV : X1, X2
Degré = X1, X2
ALPHA : X1

C1: O<5; C2 : U pair
MRV : X2



X1 = 1

C4 : O est pair
W<5
O:{2,4} donc R={4, 8}
MRV : R,O | Degre : 0



R=4, T=6
Tous différent de 2
MRV : R,T | Degre : R,T
ALPHA : R



Tous diff de 4

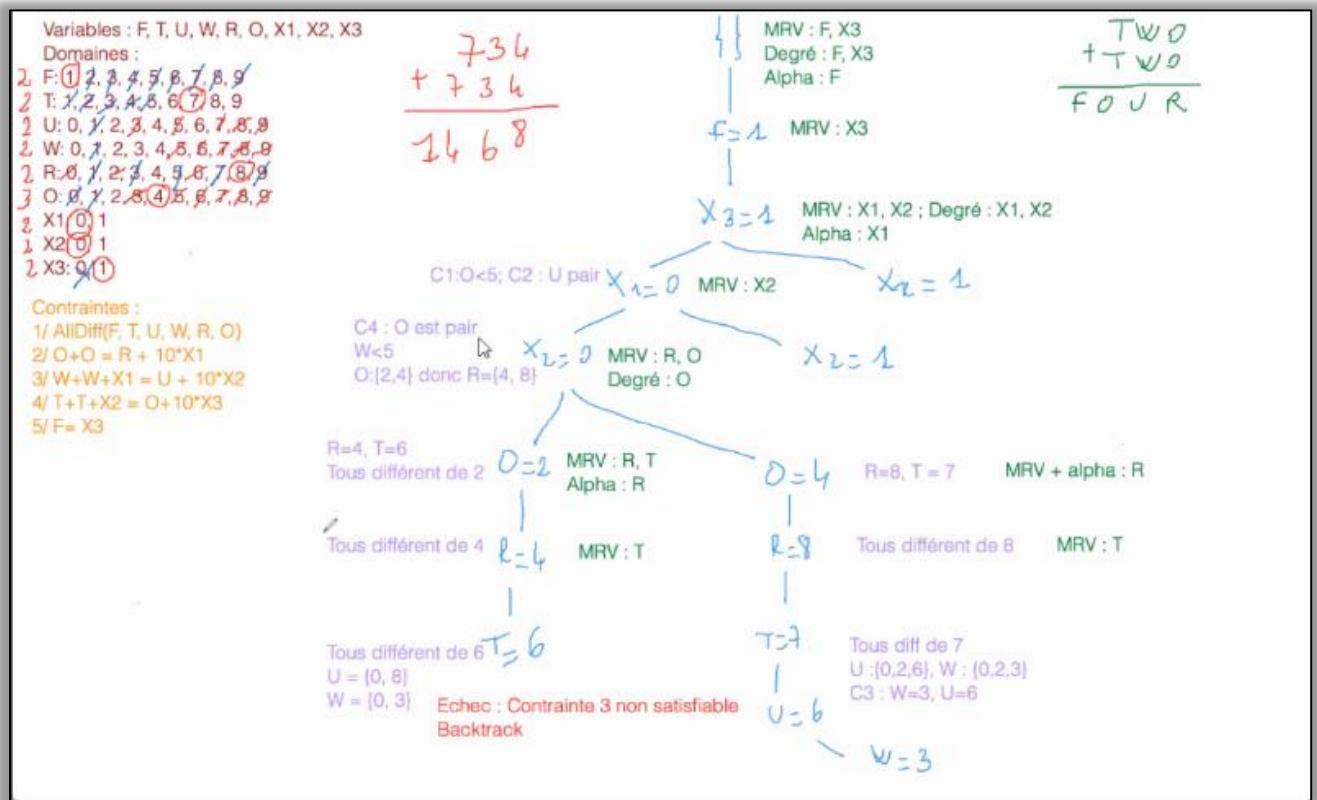
R = 4

NRV : T

T = 6

Tous différent de 6 $T \geq 6$
U = {0, 8}
W = {0, 3} Echec : Contrainte 3 non satisfiable
Backtrack

Echec -> donc on fait un backtrack



Exercice 3

Un carré magique est une matrice 3×3
 dont chaque case contient un nombre différent compris entre 1 et 9,
 de façon à ce que la somme de chaque ligne,
 chaque colonne et chaque diagonale ait la même valeur (soit 15).

A ₁	A ₂	A ₃
B ₁	B ₂	B ₃
C ₁	C ₂	C ₃

Question 1

Modélisez ce problème sous la forme d'un CSP : établissez la liste des variables nécessaires, ainsi que les contraintes existant entre ces variables

Variables etape1	Domaines des variables etape2	Degrés des variables etape 4	Contraintes entre les variables etape 3
A1	{1,2,3,4,5,6,7,8,9}	4	
A2	{1,2,3,4,5,6,7,8,9}	3	
A3	{1,2,3,4,5,6,7,8,9}	4	
B1	{1,2,3,4,5,6,7,8,9}	3	
B2	{1,2,3,4,5,6,7,8,9}	5	
B3	{1,2,3,4,5,6,7,8,9}	3	
C1	{1,2,3,4,5,6,7,8,9}	4	
C2	{1,2,3,4,5,6,7,8,9}	3	
C3	{1,2,3,4,5,6,7,8,9}	4	$A_1 + A_2 + A_3 = 15$ $B_1 + B_2 + B_3 = 15$ $C_1 + C_2 + C_3 = 15$ Somme des colonnes = 15 $A_1 + B_1 + C_1 = 15$ $A_2 + B_2 + C_2 = 15$ $A_3 + B_3 + C_3 = 15$ Les 2 diagonales $A_1 + B_2 + C_3 = 15$ $A_3 + B_2 + C_1 = 15$ AllDiff(A1,A2,...) $A_1 \neq A_2 \neq A_3 \neq B_1 \neq B_2 \neq B_3 \neq C_1$ $\neq C_2 \neq C_3$

Exercice 3

Un carré magique est une matrice 3×3 dont chaque case contient un nombre différent compris entre 1 et 9, de façon à ce que la somme de chaque ligne, chaque colonne et chaque diagonale ait la même valeur (soit 15).

A ₁	A ₂	A ₃
B ₁	B ₂	B ₃
C ₁	C ₂	C ₃

Question 2

Déterminez quelle variable est la plus contrainte. Etudiez ce qui se passe si la valeur donnée à cette variable est 1, puis 2. Déduisez en sa valeur.

Variables	Degrés des variables
A1	4
A2	3
A3	4
B1	3
B2	5
B3	3
C1	4
C2	3
C3	4

Variable la plus contrainte : B2.

Si B2 = 1 :

D'abord, les autres variables ne peuvent pas être égale à 1...

$$A2 + C2 = 14$$

$$B1 + B3 = 14$$

$$A1 + C3 = 14$$

$$A3 + C1 = 14$$

Comment je peux obtenir 14 en additionnant 2 chiffres différents ?

On a que 2 possibilités : 6 + 8 ou 5 + 9

donc Jai 4 valeurs possibles pour les 8 variables..

On va pouvoir trouver une solution ?

Cest insatisfaisable, impossible de trouver une solution qui satisfait les contraintes.

Et si $B_2 = 2$?

Le raisonnement est le même sauf que la somme vaut 14 et pas 13 :

$$B_1 + B_3 = 13$$

$$A_2 + C_2 = 13$$

$$A_1 + C_3 = 13$$

$$A_3 + C_1 = 13$$

POSSIBILITES POUR obtenir 13 en additionnant 2 chiffres différents (sauf le 2) :

$$7 + 6, 9 + 4, 8 + 5$$

=> 6 valeurs possibles pour 8 variables => donc impossible.

Variable la plus contrainte : B_2

Si $B_2 = 9$: c/ $B_1+B_3 = 6$ f/ $A_2+C_2 = 6$ h/ $A_1+C_3 = 6$ i/ $A_3+C_1 = 6$	Si $B_2 = 8$: c/ $B_1+B_3 = 7$ f/ $A_2+C_2 = 7$ h/ $A_1+C_3 = 7$ i/ $A_3+C_1 = 7$
--	--

Possibilités pour obtenir 6 en additionnant 2 chiffres différents :
1+5 ou 4+2
=> 4 valeurs possibles pour 8 variables
Impossible de trouver une solution qui satisfait les contraintes

Possibilités pour obtenir 7 en additionnant 2 chiffres différents (sauf le 2) :
6+1, 5+2, 4+3
=> 6 valeurs possibles pour 8 variables
Impossible

On a 9 valeurs pour 9 variables, donc chaque valeur (entre 1 et 9) doit être instanciée 1 et 1 seule fois.
Si B_2 est différent de 1, il faut que l'une des 8 autres variables soit égale à 1.

Si $B_2 < 5$, les 4 contraintes (c, f, h, i) donnent des sommes qui sont ≥ 11 pour les 8 autres variables. Dans ce cas là, on ne pourra pas utiliser la valeur 1 pour une variable.

On en déduit donc que $B_2 \geq 5$.

Si $B_2 > 5$, les 4 contraintes (c, f, h, i) donnent des sommes qui sont ≤ 9 pour les 8 autres variables. Dans ce cas là, on ne pourra pas utiliser la valeur 9 pour une variable.

On en déduit donc que la seule valeur possible pour $B_2 = 5$.

Donc on met à jours nos contraintes et nos domaines :

Variables : A1, A2, A3, B1, B2, B3, C1, C2, C3

Domaines :

4 A1 : 1, 2, 3, 4, 5, 6, 7, 8, 9

3 A2 : 1, 2, 3, 4, 5, 6, 7, 8, 9

4 A3 : 1, 2, 3, 4, 5, 6, 7, 8, 9

3 B1 : 1, 2, 3, 4, 5, 6, 7, 8, 9

5 B2 : 1, 2, 3, 4, 5, 6, 7, 8, 9

3 B3 : 1, 2, 3, 4, 5, 6, 7, 8, 9

4 C1 : 1, 2, 3, 4, 5, 6, 7, 8, 9

3 C2 : 1, 2, 3, 4, 5, 6, 7, 8, 9

4 C3 : 1, 2, 3, 4, 5, 6, 7, 8, 9

Contraintes

a/ AllDiff(A1, A2, A3, B1, B2, B3, C1, C2, C3)

b/ A1+A2+A3 = 15

c/ B1+B2+B3 = 15 \wedge 0

d/ C1+C2+C3 = 15

e/ A1+B1+C1 = 15

f/ A2+B2+C2 = 15 \wedge 0

g/ A3+B3+C3 = 15

h/ A1+B2+C3 = 15 \wedge 0

i/ A3+B2+C1 = 15 \wedge 0

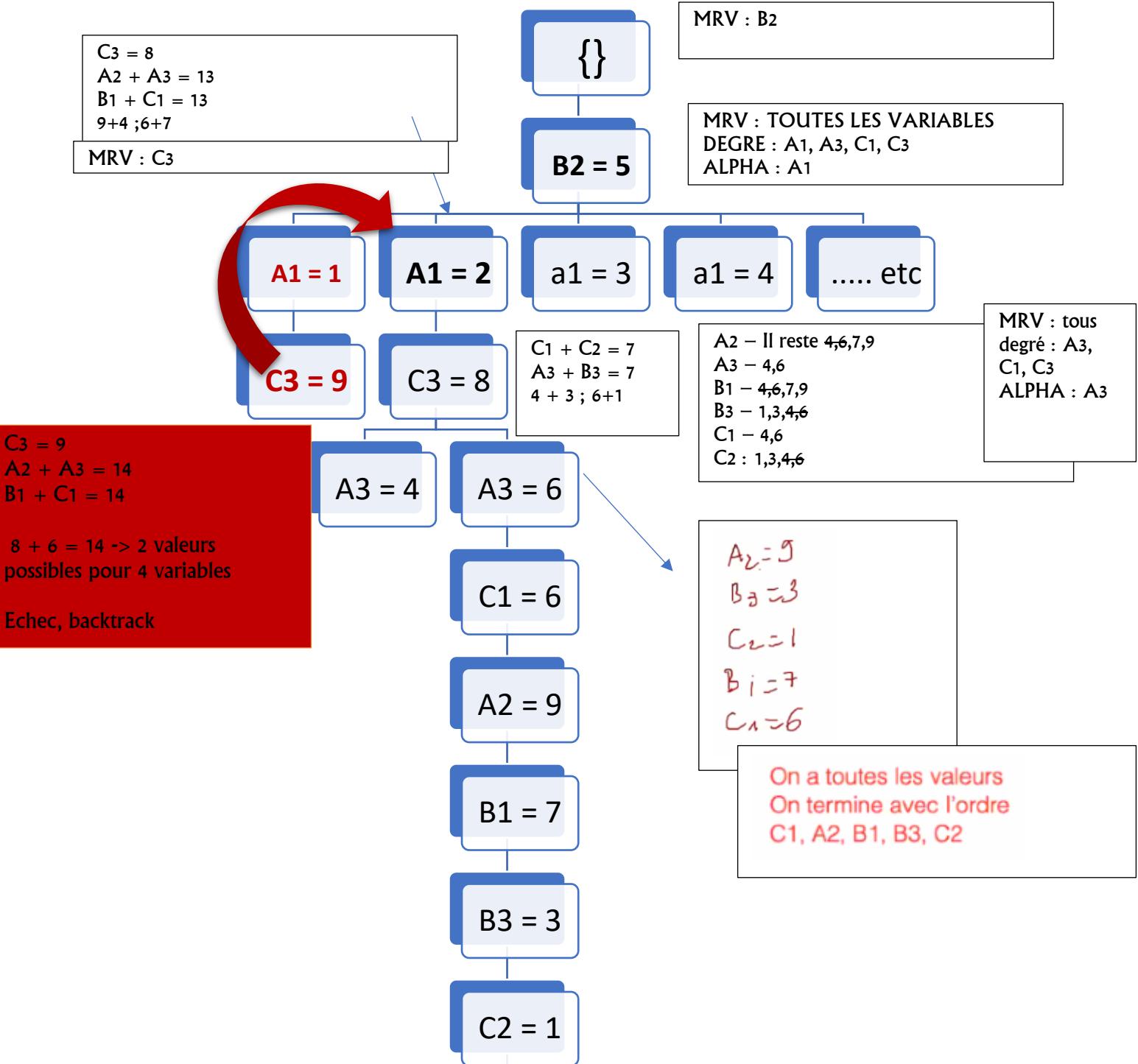
Une fois qu'on a ça on peut appliquer le même algo que tout a l'heure.

Exercice 3

Un carré magique est une matrice 3×3 dont chaque case contient un nombre différent compris entre 1 et 9, de façon à ce que la somme de chaque ligne, chaque colonne et chaque diagonale ait la même valeur (soit 15).

Question 3

. Résolvez ce CSP en utilisant la recherche par backtrack avec recherche en avant, heuristique MRV et heuristique du degré. Si vous avez le choix entre plusieurs variables, vous choisirez en suivant l'ordre alphanumérique. Si vous avez le choix entre plusieurs valeurs, vous choisirez la plus petite.



A_1	A_2	A_3
B_1	B_2	B_3
C_1	C_2	C_3

2	9	4
7	5	3
6	1	8