**Intelligence artificielle (IF06M100) • TD 5 •** Livre : chapitre 6

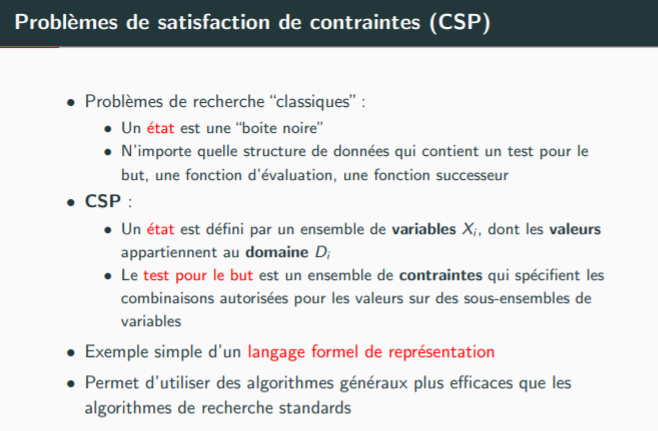
**Sources :**Artificial Intelligence: A Modern Approach by S. Russell and P. Norvig (French Version)  
Cours UE Intelligence artificielle / E. Bonzon

**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 boite 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.



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, .**
* **D est un ensemble de domaines, , un pour chaque variable.**
* **C est un ensemble de contraintes qui spécifient les combinaisons admissibles de valeurs.**
* Chaque domaine est constitué   
  d’un ensemble de valeurs admissibles pour la variable
* Chaque contrainte est constituée   
  d'une paire de ,

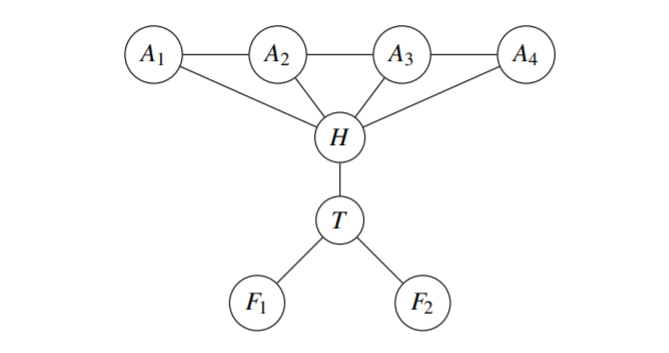
où **portée** estun n-uplet de variables qui participent à la contrainte   
et **rel** est une relation qui définit les valeurs que ces variables peuvent prendre.

* 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 et ont tous les deux le domaine ,   
  la contrainte qui dicte que les deux variables doivent avoir des valeurs différentes peut s’écrire
* 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, .
* 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.

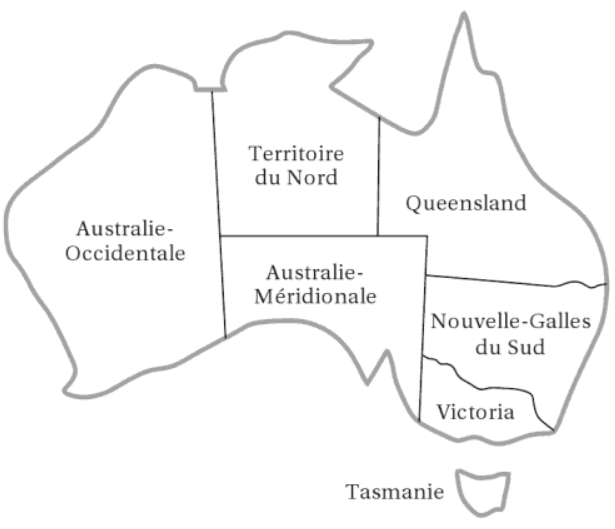
Si vous avez le choix entre plusieurs variables, vous choisirez en suivant l’ordre alphanumérique. Vous appliquerez les couleurs en respectant l’ordre {R, V, B}.



le graphe de contraintes

Coloration d'un plan

**Nous nous intéressions à une carte administrative de l'Australie représentant les États et les territoires de ce pays et que nous souhaitions colorer chaque région en rouge, vert ou bleu, de telle manière qu'aucune région n'ait la mème 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, .**

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

* **D est un ensemble de domaines, , un pour chaque variable.**

**Domaines : Di = {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 :**

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

**{(rouge, vert), (rouge, bleu), (vert, rouge), (vert, bleu), (bleu, rouge), (bleu, vert)}.**

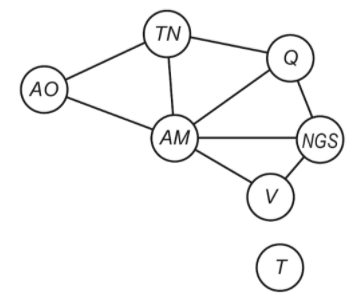
**Il existe de nombreuses solutions à ce problème, dont la suivante :**

Les solutions sont des affectations qui satisfont toutes les contraintes.



**Graphe de contraintes**

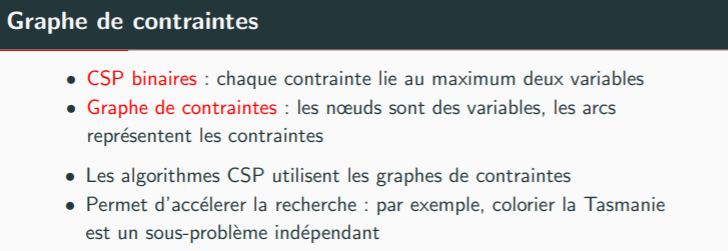
**Il peut être utile de visualiser un CSP sous la forme d'un graphe de contraintes, ce qui apporte une aide appréciable.**



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 35 = 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 25 = 32 assignations, soit une réduction de 87 %.



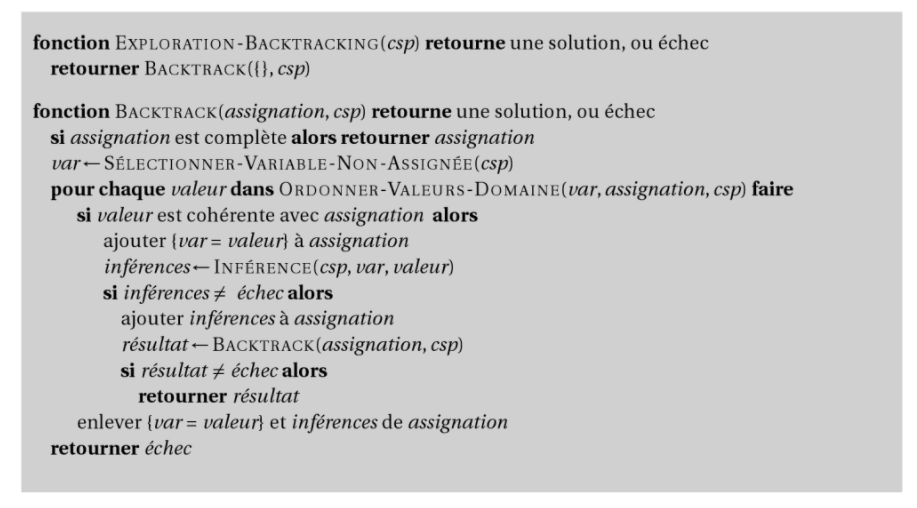
* Dans l'exploration standard des espaces d'états, nous pouvons seulement nous demander : Est-ce que cet état est un but ? Non? Et celui-là ?
* Avec les CSP une fois que nous savons qu’une assignation partielle enfreint certaines contraintes, nous pouvons immédiatement rejeter en bloc toute assignation 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.



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

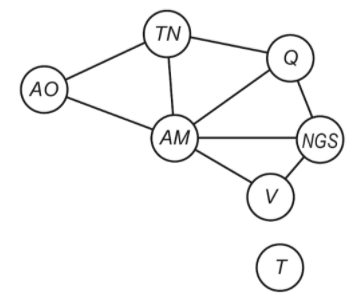
**La fonction INFÉRENCE peut optionnellement être utilisée pour imposer une cohérence d’arc, une cohérence de chemin ou une k-cohérence par exemple.**

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

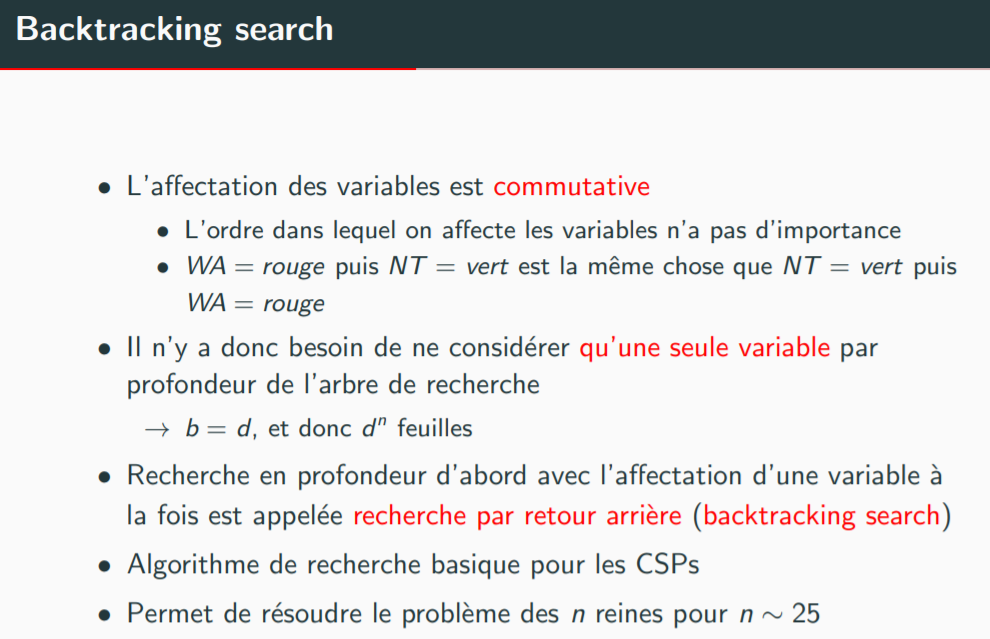
**Si une incohérence est détectée, BACKTRACX renvoie un échec, le précédent appel essaie une autre valeur.**

**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.**

* Cette section : algorithmes d'exploration avec backtracking qui fonctionnent sur des **assignations 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,

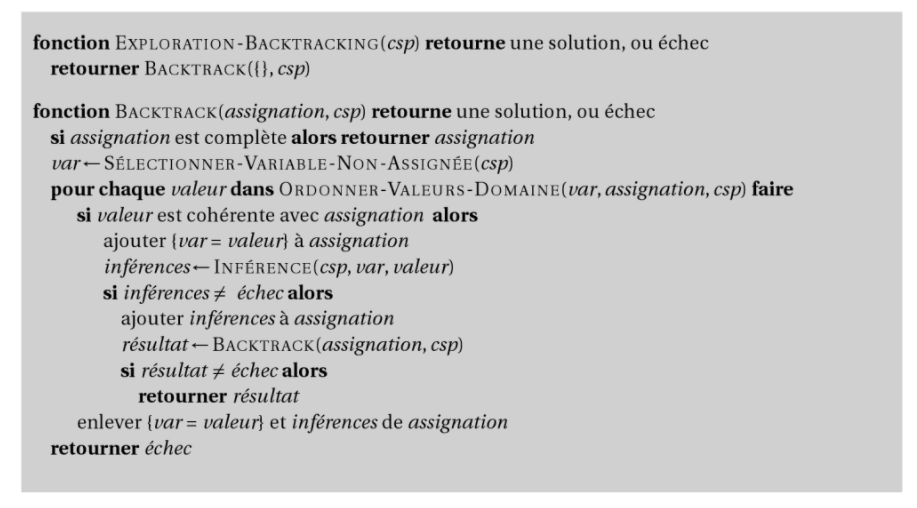


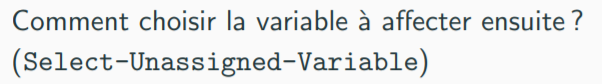
* + On peut choisir entre AM = rouge, AM = vert et AM = bleu, mais jamais entre AM = rouge et AO = bleu.
  + Compte tenu de cette restriction, le nombre de feuilles est dn, comme on l'espérait.



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 :





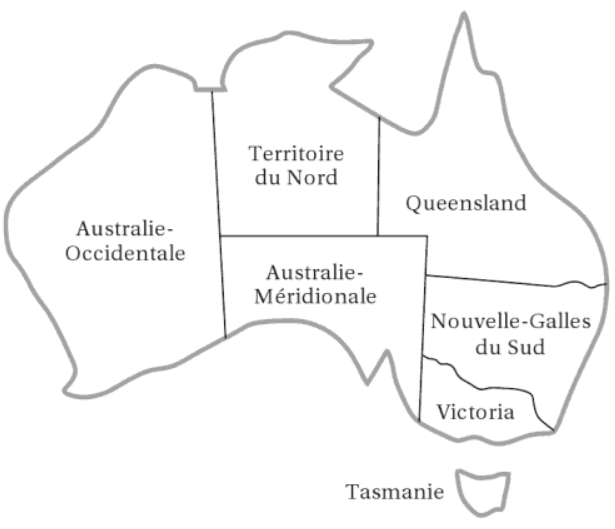
L'algorithme de backtracking contient la ligne

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

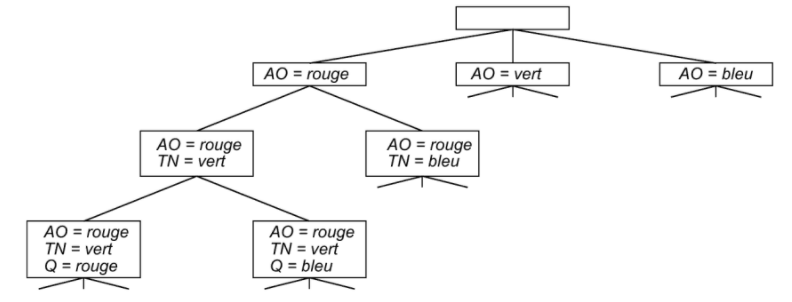
La stratégie la plus simple pour **SÉLECTIONNER-VARIABLE-NON-ASSIGNÉE** se contente de sélectionner la variable non assignée suivante dans l'ordre {X1, X2,...}.

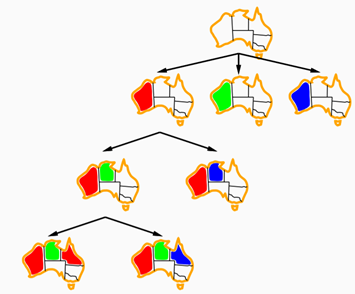
Cet ordre statique des variables débouche rarement sur l'exploration la plus efficace.

Par exemple,



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

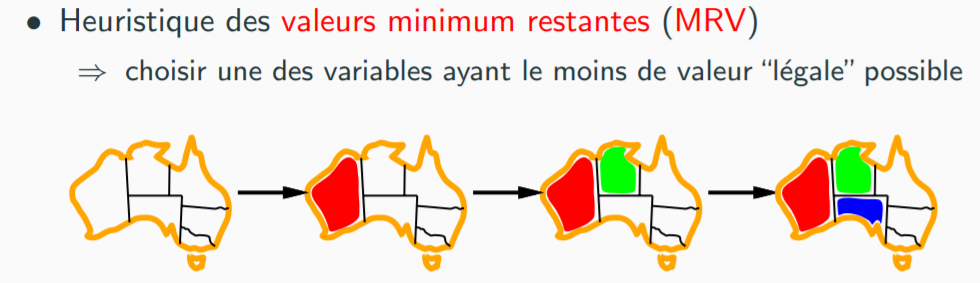




En fait, une fois AM assigné, les choix pour Q, NGS et V s'imposent.

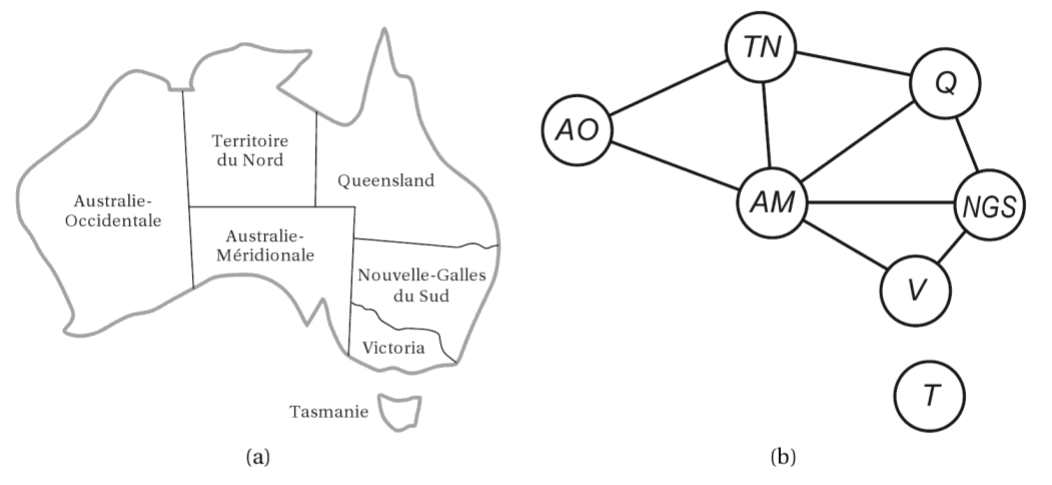
Heuristique des valeurs minimum restantes (MRV)

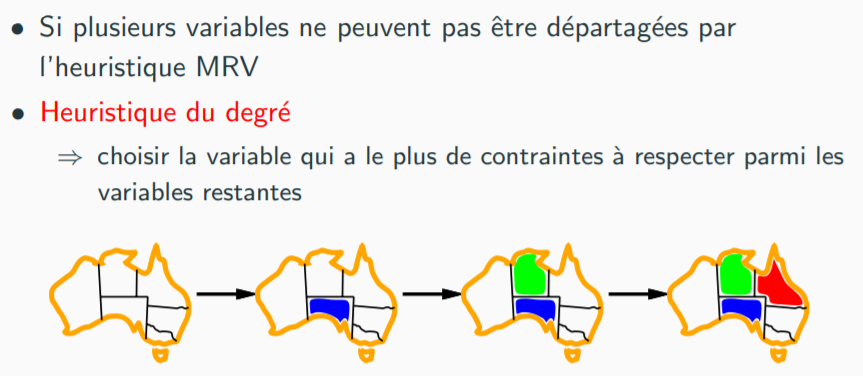
* Cette idée intuitive (le choix de la variable dont les valeurs «légales » sont les moins nombreuses) est appelée **l’heuristique du minimum de valeurs restantes**, ou **MRV (Minimum Remaining Values)**.
* **On la nomme également** **«heuristique de la variable la plus contrainte »** ou heuristique fail-first (maillon faible), car on sélectionne la variable qui risque le plus de provoquer un échec, ce qui contribue à élaguer 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 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.



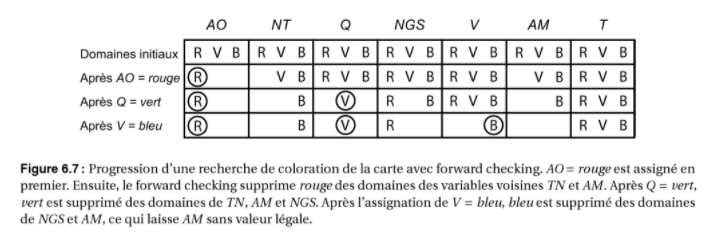


* L'heuristique MRV est habituellement un guide plus sûr, mais l'heuristique des degrés peut être précieuse pour départager des options que ne distingue pas MRV.

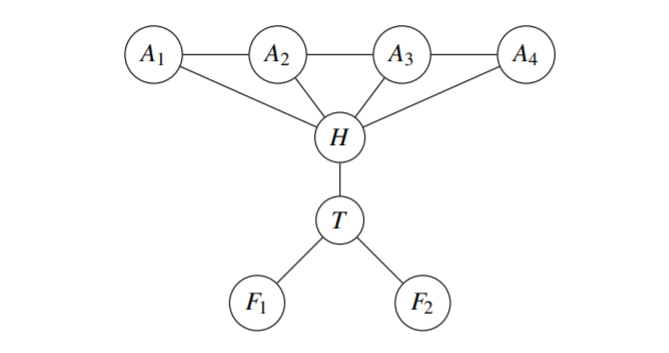
Améliorer l’efficacité du la recherche par backtrack (2)  


Association de l'exploration et de l'inférence

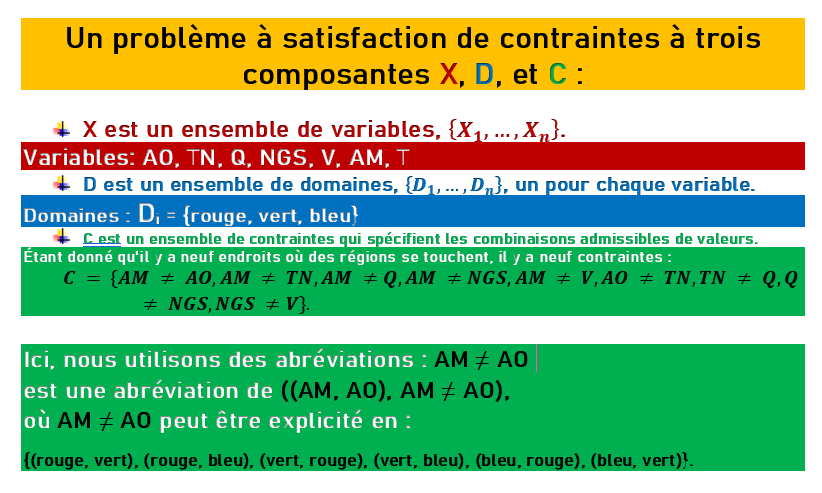
* **L'inférence** peut être puissante au cours d’une exploration : chaque fois que nous choisissons une valeur pour une variable, nous avons une occasion d’inférer de nouvelles réductions de domaine sur les variables voisines.
* **Une des formes les plus simples d’inférence** est appelée **forward checking**, 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,**



* Tout d’abord, remarquez ceci: après qu’on a assigné AO = rouge et Q = 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 .**Intuitivement, il semble que l’assignation contraigne ses voisins, et , donc nous devrions traiter ces variables ; toutes les autres variables trouveront leur place.
* C'est exactement ce qui se passe avec MRV : et ont deux valeurs : on choisit une en premier, puis l’autre, puis , et , dans cet ordre.
* **Enfin 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 est en rouge et que est en vert, et doivent être en bleu. Forward checking ne va pas assez loin pour remarquer qu'il y a un problème : et sont côte à côte et ne peuvent pas avoir la même valeur.



Premier chose à faire : bien définir le problème :

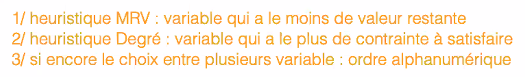


Exemple

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

|  |  |  |
| --- | --- | --- |
| Variables | Domaines des variables | Degréss |
| A1 A2 A3 A4 H T F1 F2 | R, V, G R, V, G  R, V, G R, V, G R, V, G R, V, G R, V, G R, V, G | * **(Combien de contraintes a la variables A1 …. ?)** |

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 mois de valeurs dispo ?
* Quand on a plusieurs chois possible avec MRV on applique l’heuristique de degré (la variable qui a le plus de contraintes à respecter)  
    
  

MRV : F2

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

MRV : T

MRV : A4

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

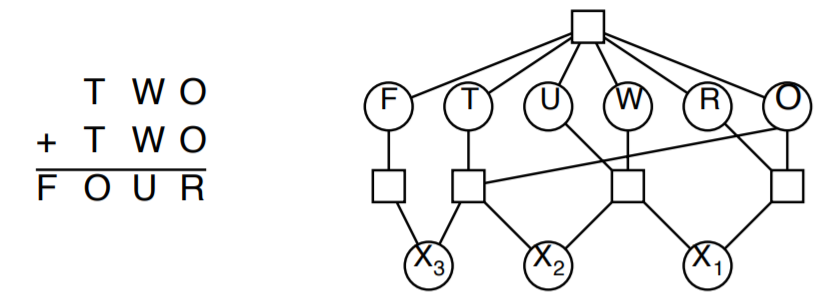


Dapres MRV , on a le choix entre quoi et quoi ?  




**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.



Le graphe de contraintes  
(contraintes pas forcement binaire)

Objectif : résoudre cette addition

T et F peuvent pas être égal a 0 !

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  O + O = R + 10 \* x1  Quand on fait une addition si O est égal a 7 on a 7+7 = 14 = 4 + 10\* 1 donc on a une retenue de 1  W + W + X1 = U + 10 \* X2  T + T + X2 = O + 10 \* X3  F = X3  chaque contrainte correspond à un carrée sur le graph des contraintes |
| X2 | {0,1} | 2 |
| X3 | {0,1} | 2 |
| F | {1,2,3,4,5,6,7,8,9} | 2 |
| T | {1,2,3,4,5,6,7,8,9} | 2 |
| U | {0,1,2,3,4,5,6,7,8,9} | 2 |
| W | {0,1,2,3,4,5,6,7,8,9} | 2 |
| 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 a l’heure c’est la propagation des contrariantes :propagation des contraintes : pré-proccessing

**contrainte 2 :** O + O = R + 10 \* x1

On peut réduire le domaine dune 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

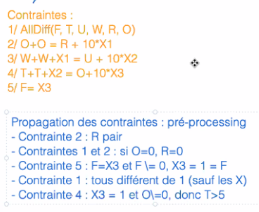
Donc : contrainte 2 -> R paire

La contrainte 2 me permet de déduire autre chose ?  
Quesque il se passe si O est égal a 0 ?  
d’après la contrainte 1, O ne peut pas être égal a 0.  
R peut être egal a 0 ? oui..  
  
contrainte 3 : rien a déduire

Contrainte 5 me permet de déduire :  
puisque F = X3 et F est diffèrent de O, X3 = 1 = F

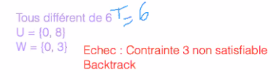
On peut déduire autre chose ?  
on peut déduire que les autres ne peuvent pas être 1 (contrainte 1) (Sauf les X)

Dernière déduction a partir de la contrainte 4 : si X3 = 1 Quesque on sait ?  
T supérieur ou égale a 5. Comme O est diffèrent de 0, T est même strictement supérieur a 5.



On commence l’algorithme de recherche

On commence l’algorithme de recherche avec une affectation vide



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

MRV : X3

MRV : F, X3  
Degré = F, x3   
ALPHA : f

NRV : T

Tous diff de 4

MRV : R,T | Degre : R,T  
ALPHA : R



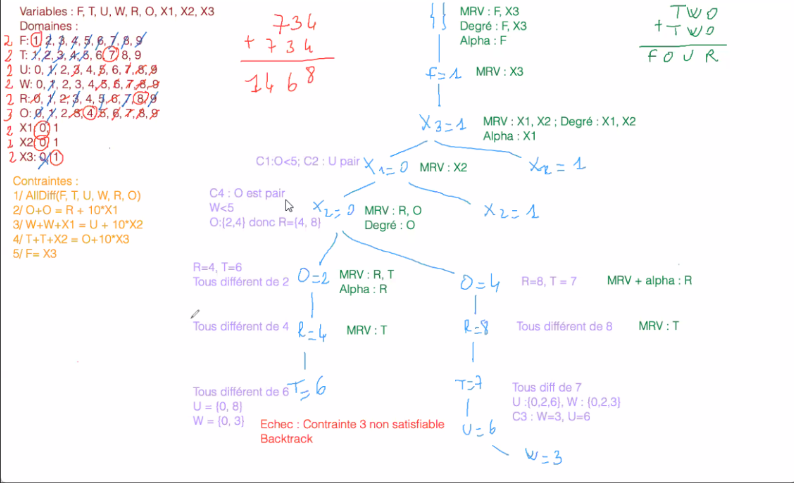
MRV : R,O | Degre : 0



MRV : X2



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).

|  |  |  |
| --- | --- | --- |
| **A1** | **A2** | **A3** |
| **B1** | **B2** | **B3** |
| **C1** | **C2** | **C3** |

**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 | A1 +A2 + A3 = 15 B1 + B2 + B3 = 15 C1 + C2 +C3 = 15  Somme des colonnes = 15 A1 + B1 + C1 = 15 A2 + B2 + C2 = 15 A3 + B3 + C3 = 15  Les 2 diagonales A1 + B2 + C3 = 15 A3 + B2 + C1 = 15  AllDiff(A1,A2….) A1 ≠ A2 ≠ A3 ≠ B1 ≠ B2 ≠B3 ≠C1 ≠C2 ≠C3 |
| 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 |

**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).

|  |  |  |
| --- | --- | --- |
| **A1** | **A2** | **A3** |
| **B1** | **B2** | **B3** |
| **C1** | **C2** | **C3** |

**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 a 1…**

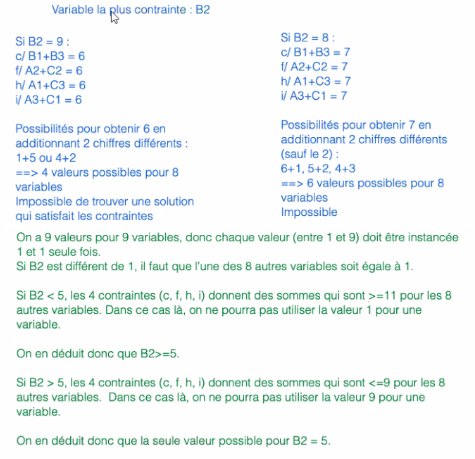
**A2 + C2 = 14   
B1 + B3 = 14  
A1 + C3 = 14**

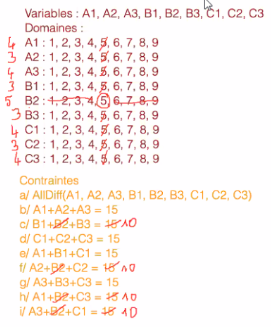
**A3 + C1 = 14**

**Comment je peux obtenir 14 en additionnent 2 chiffres diffèrent ?  
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 B2 = 2 ?**

**Le résonnement est le même sauf que les somme valle 14 et pas 13 :  
  
B1 + B3 = 13  
A2 + C2 = 13  
A1 + C3 = 13  
A3 + C1 = 13  
  
POSSIBLITES POUR obtenir 13 en additionnant 2 chiffres différents (sauf le 2) :  
7 + 6, 9 + 4, 8 + 5  
=> 6 valeurs possibles pour 8 variable => donc impossible.**



**Donc on met à jours nos contraintes et nos domaines :**

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

**Exercice 3**

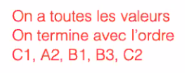
Un carré magique est une matrice 3 × 3   
dont chaque case contient un nombre différent compris entre 1 et 9,   
de façon à ce que la somme de chaque ligne,   
chaque colonne et chaque diagonale ait la même valeur (soit 15).

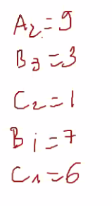
**Question 3**

. 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.

|  |  |  |
| --- | --- | --- |
| **2** | **9** | **4** |
| **7** | **5** | **3** |
| **6** | **1** | **8** |

|  |  |  |
| --- | --- | --- |
| **A1** | **A2** | **A3** |
| **B1** | **B2** | **B3** |
| **C1** | **C2** | **C3** |





MRV : tous  
degré : A3, C1, C3  
ALPHA : A3

A2 – Il reste ~~4,6~~,7,9  
A3 – 4,6  
B1 – ~~4,6~~,7,9  
B3 – 1,3,~~4,6~~  
C1 – 4,6  
C2 : 1,3,~~4,6~~

C1 + C2 = 7  
A3 + B3 = 7  
4 + 3 ; 6+1

MRV : C3

C3 = 8  
A2 + A3 = 13  
B1 + C1 = 13  
9+4 ;6+7

MRV : TOUTES LES VARIABLES  
DEGRE : A1, A3, C1, C3  
ALPHA : A1

MRV : B2

C3 = 9  
A2 + A3 = 14  
B1 + C1 = 14  
  
 8 + 6 = 14 -> 2 valeurs possibles pour 4 variables  
  
Echec, backtrack