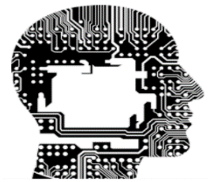
Université Constantine 2
جامعة قسنطينة 2

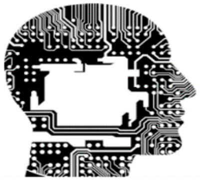


Foundation of Artificial Intelligence

CSPs

Dr. NECIBI Khaled
Faculté des nouvelles technologies
Khaled.necibi@univ-constantine2.dz

Université Constantine 2
جامعة قسنطينة 2



Foundation of Artificial Intelligence
Les Problèmes à Satisfaction de Contraintes

Dr. NECIBI Khaled
Faculté des nouvelles technologies
Khaled.necibi@univ-constantine2.dz

Etudiants concernés

Faculté/Institut	Département	Niveau	Spécialité
Nouvelles technologies	IFA	Master 01	SDIA

Objectif du cours

- Apprendre à formuler un problème à satisfaction de contraintes (CSP)
- Maitriser les différents algorithmes et heuristiques utilisés pour résoudre les CSPs

Objectif du cours

- Dans un problème de recherche standard, l'état représente une boite noire qui contient l'état but, la fonction successeur et la fonction heuristique
- La structure interne d'un état est spécifique à chaque problème
- Les problèmes à satisfaction de contraintes représente un type de problème où les <u>états</u> et le but peuvent être mis sous forme standard, structuré et simple
- Cette représentation permet de définir des <u>heuristiques</u> beaucoup plus générales applicables à tout sorte de CSP

Objectif du cours

- Un CSP est défini par :
  - Un ensemble de variables $X_1, X_2, \ldots, X_n$
  - Un ensemble de domaines D_i ou chaque D_i représente les valeurs possibles que peut prendre la variable X_i
- Un ensemble de contraintes $C_1, C_2, \ldots, C_p$ ou :
  - chaque contrainte C_i implique un sous-ensemble de variables
  - Chaque contrainte C_i spécifie les combinaisons de valeurs de ces variables qui soient admissibles
- Un état dans un CSP est défini par l'affectation ou l'attribution (assignment) de valeurs à certains, ou tout les variables : $\{X_i = v_i, X_j = v_j, \ldots\}$

Problèmes à Satisfaction de Contraintes

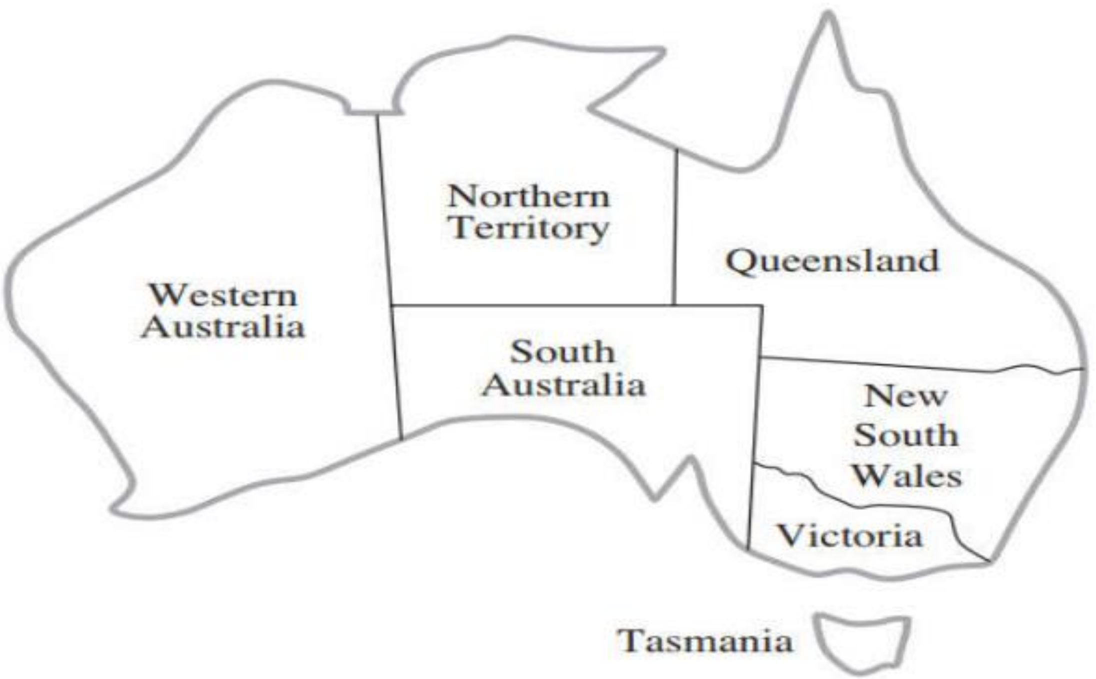
- Affectation consistante : une affectation qui ne viole pas les contraintes
- Affectation complète : affectation de tout les variables
- Affectation partielle : affectation de certaines variables seulement
- Affectation inconsistante : une affectation qui viole au moins une contrainte
- Une solution dans un CSP : est une affectation **complète** qui satisfait tout les contraintes (consistante)
- Exemple :
  - $X = \{X_1, X_2, X_3, X_4\}$
  - $D = \{D_1, D_2, D_3, D_4\}$ avec $D_1 = D_2 = D_3 = D_4 = \{0,2\}$
  - $C = \{X_1 = X_3, X_1 \neq X_2, X_1 + X_3 < X_2\}$

- Exemple :
  - $X = \{X_1, X_2, X_3, X_4\}$
  - $D = \{D_1, D_2, D_3, D_4\}$ avec $D_1 = D_2 = D_3 = D_4 = \{0,2\}$
  - $C = \{X_1 = X_3, X_1 \neq X_2, X_1 + X_3 < X_2\}$
- $A = \{(X_2,0),(X_3,1)\}$ → affectation
- $A = \{(X_1,0),(X_2,0),(X_3,0),(X_4,0)\}$ → affectation totale
- $A = \{(X_1,0),(X_2,0)\}$ → affectation partielle
- $A = \{(X_1,0),(X_2,0)\}$ → affectation inconsistante
- $A = \{(X_3,0),(X_4,1)\}$ → affectation consistante
- $A = \{(X_1,0),(X_2,1),(X_3,0),(X_4,2)\}$ → solution

Problèmes à Satisfaction de Contraintes

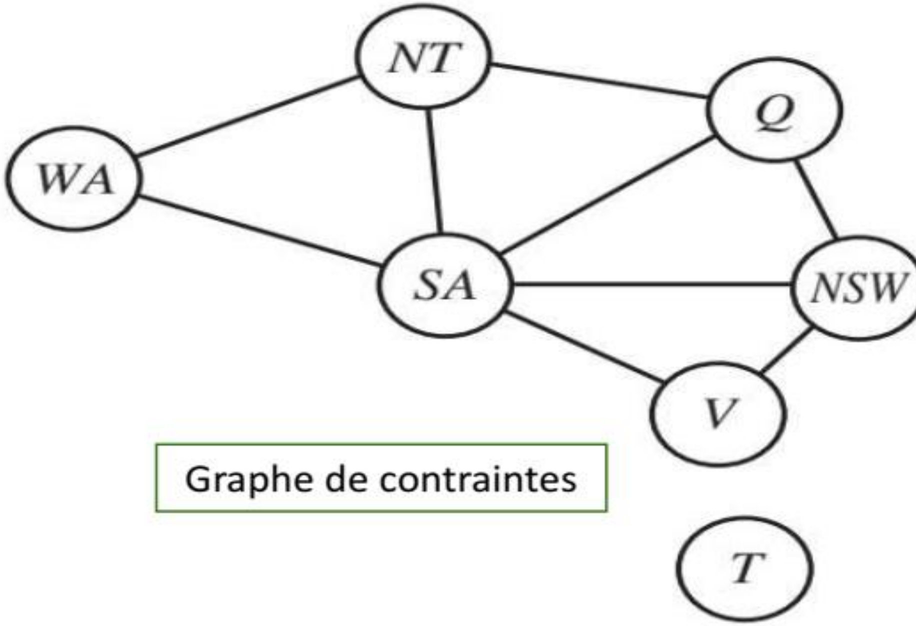
- Remarque
  - Certains CSPs nécessite une solution qui maximise une fonction objective
- Graphe de contraintes
  - Un CSP peut être visualisé par un graphe de contrainte
  - Chaque nœud dans le graphe correspond à une variable
  - Chaque arc dans le graphe correspond à une contrainte
- Avantages
  - La fonction successeur et l'état test de but peuvent être représentés d'une manière générique et applicable à tout les CSPs
  - Il est possible de développer des heuristiques plus générales
  - La structure du graphe de contraintes peut être exploitée afin de simplifier la résolution des CSPs

- Exemple : problème de coloration de carte
  - Considération la MAP de l'Australie
  - Le but est de colorier chaque région en <u>rouge</u>, <u>vert</u> et <u>bleu</u>
  - <u>Conditions</u> : aucune région ne peut avoir la même couleur qu'une de ses voisines



- Exemple : problème de coloration de carte
  - Pour formuler cet exemple comme un CSP on définit les variables qui correspond aux régions :
  - X = {WA, NT, Q, NSW, V, SA, T}
  - Le domaine de chaque variable est un ensemble :
  - D_i = {rouge, vert, bleu}
  - Étant donné qu'il y a neuf endroits ou les régions se touchent → il y a neuf contraintes :
  - C = {SA ≠ WA, SA ≠ NT, SA ≠ Q, SA ≠ NSW, SA ≠ V, WA ≠ NT, NT ≠ Q, Q ≠ NSW, NSW ≠ V}
  - On utilise l'abréviation : SA ≠ WA est une abréviation de ((SA, WA), SA ≠ WA) ou SA ≠ WA peut être explicité en :
  - {(rouge, vert),(rouge, bleu), (vert, rouge), (vert, bleu), (bleu, rouge), (bleu, vert)}

- Exemple : problème de coloration de carte
  - Il existe de nombreuses solutions à ce problèmes dont la suivante :
  - {WA = rouge, NT = vert, Q = rouge, NSW = vert, V = rouge, SA = bleu, T = rouge}
  - Il peut être utile de visualiser un CSP sous forme d'un graphe de contraintes



Graphe de contraintes

- Exemple de CSPs du monde réel
  - Problème d'affectation : qui enseigne, et dans quelle classe ?
  - Problème d'emploi du temps
  - Planification des transports
  - Problèmes de planification
  - Etc…

- Formulation de CSPs
- Formulation Incrémentale
  - État initial : affectation vide {}, tout les variables n'ont pas de valeurs
  - Fonction Successeur : chaque action permet d'affecter une valeur à une variable en satisfaisant les contraintes
  - Test de but : l'état courant est complet
  - Coût de chemin : un coût constant pour chaque affectation
- Question
  - Quelle est la profondeur de l'arbre de recherche dans ce cas ?
- Formulation Complète
  - Dans cette formulation chaque affectation est complète (affectation totale), <u>mais pas forcement consistante</u>

Problèmes à satisfaction de contraintes : Variétés

- Variétés des CSPs : Selon les variables
- Variables discrètes
  - Avec un ensemble de domaine fini. Exemple : Coloration de MAP, et des CSPs Boolean où les variable peuvent être soit vrai soit faux
  - Si d est la taille maximale de l'ensemble de domaine pour n'importe quelle variable, → le nombre total possible des affectations complètes est en $O(d^n)$
  - Avec un ensemble de domaines infini, exemple : ordonnancement industriel (Job Scheduling)

Problèmes à satisfaction de contraintes : Variétés

- Variétés des CSPs : Selon les contraintes
- Contraintes unaires
  - Implique une seule variable
  - Exemple : C1 ≠ green
- Contraintes binaires
  - Implique une paire de variables
  - Exemple : C1 ≠ C2
- Tout contrainte n-aire peut s'exprimer en termes de contraintes binaires
- Tout CSP peut se représenter comme un CSP binaire
- Un CSP binaire peut être représenté comme un graphe de contrainte

- Problème de coloration de cartes avec 03 régions et 03 couleurs différentes
- On peut utiliser la recherche dans un graphe avec la stratégie DFS et les paramètres suivants:
  - L'espace d'états peut avoir les propriétés suivantes
  - Un état est une assignation (affectation)
  - L'état initial : assignation vide {}
  - Fonction successeur : assigne une valeur à une variable non assignée, tout en respectant les contraintes
  - But : assignation complète et consistante
- Profondeur de l'arbre de recherche = n (n : nombre de variables)
- La profondeur de la solution est n

- Problème de coloration de cartes avec 03 régions et 03 couleurs différentes
- Le facteur de branchement à la racine est nd (d : cardinalité du domaine)
- Le facteur de branchement au niveau suivant est $(n - 1)d * nd$ et ainsi de suite pour les autres niveaux
- Cela donne $n!*d^n$ nœuds générés pour seulement d^n assignations ou affectations complètes
- Problème :
  - L'algorithme ignore la commutativité des transitions :
  - (WA = Rouge suivi de NT = Vert) est équivalent à (NT = Vert suivie de WA = Rouge)

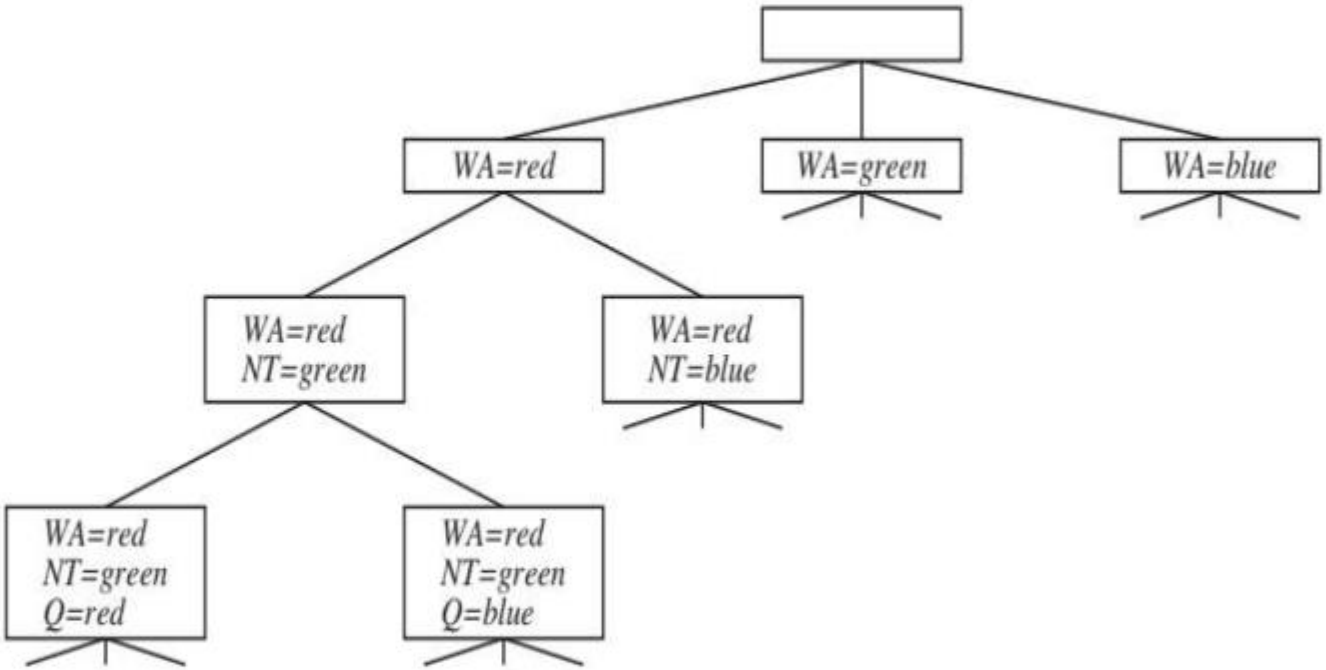
- Problème de coloration de cartes avec 03 régions et 03 couleurs différentes
- Commutativité
  - En CSP l'ordre de sélection des variables pour une affectation n'est pas significatif → plusieurs chemins sont équivalents
  - Dans tout les algorithmes CSP, à chaque nœud on ne considère qu'une seule variable pour l'affectation
  - L'ajout d'une affectation ne peut pas corriger la violation d'une contrainte

- Idée de base de Backtracking Search
- Backtracking Search utilise DFS comme stratégie de recherche
- Dès que DFS arrive à un échec sur une branche, l'algorithme Backtracking effectue un retour en arrière
- Essayer une autre valeur pour la variable
- Backtracking chronologique car la dernière décision est revisitée

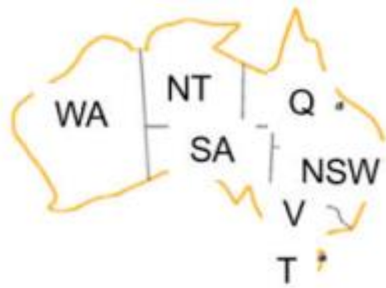
Backtracking Search

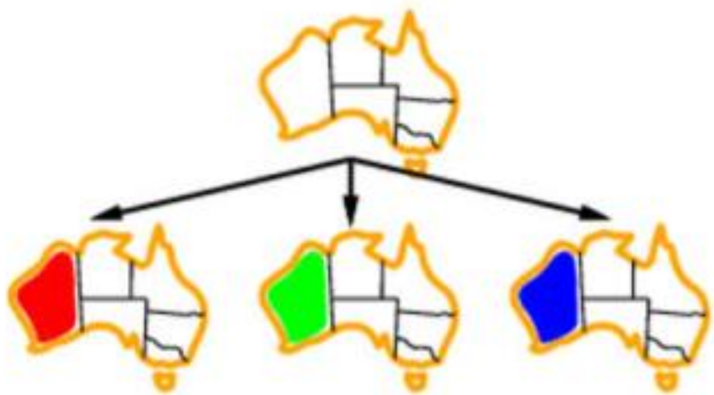
```
Function Backtracking-Search (csp) return solution ou échec
         return recursive-Backtracking ({}, csp);
End

Function recursive-Backtracking (assignment, csp) returns solution ou échec

        if (assignment est complète) then returns assignment;
        var ← Select-Unassigned-Variable(variable[csp], assignment, csp);

        for chaque valeur de Order-Domain-value(var, assignment, csp) do
                if (valeur est cohérente avec assignment) then
                ajouter {var = valeur} à assignment;
                result ← Recursive-Backtracking(assignment, csp);
                if (result ≠ échec) then return result;
                remove {var = valeur} from assignment;

        Fin

Return échec;
}
```

CSP : Recherche en mode Backtracking (Backtracking Search)

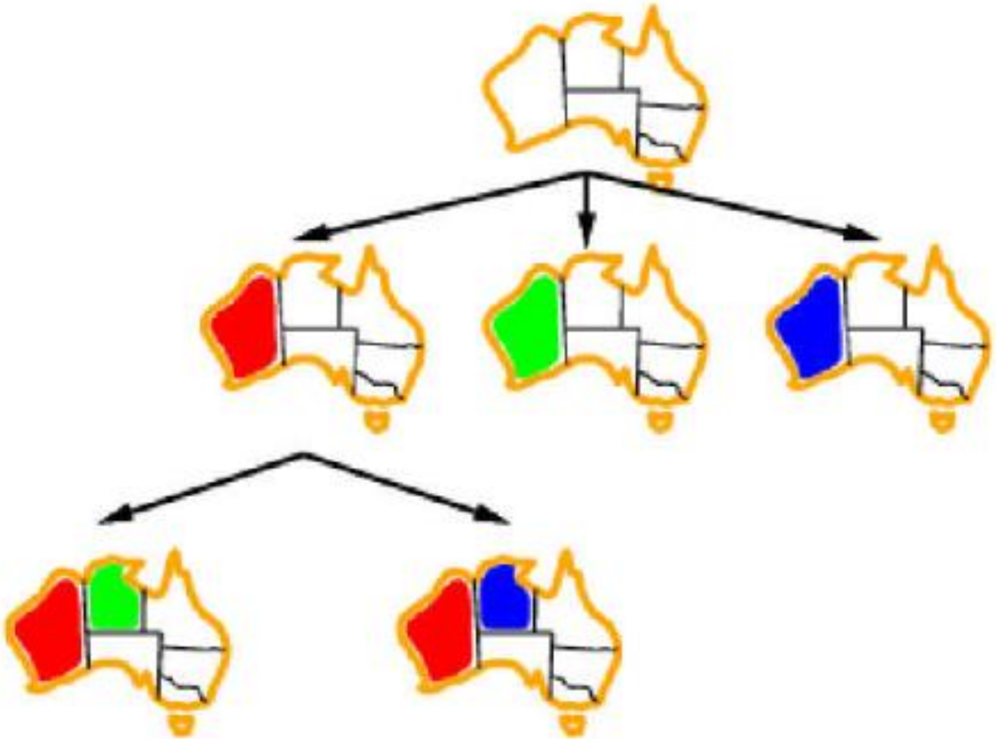


CSP : Recherche en mode Backtracking (Backtracking Search)

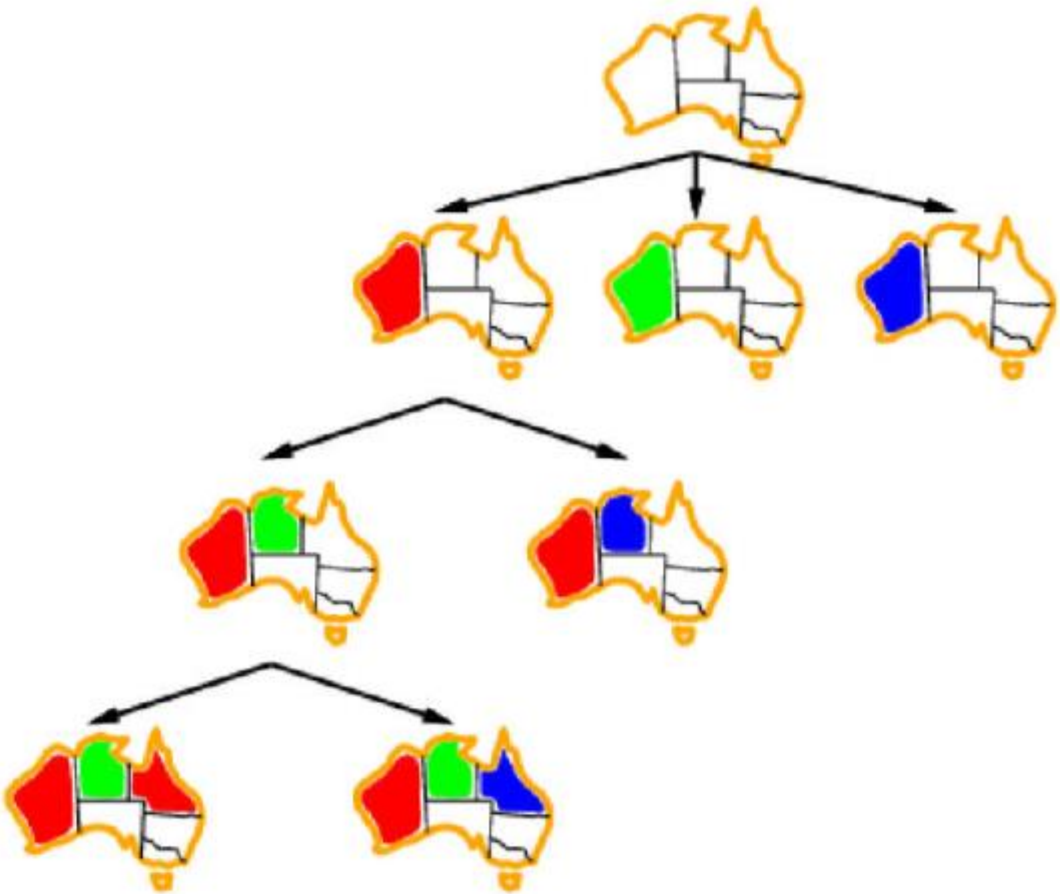




CSP : Recherche en mode Backtracking (Backtracking Search)



CSP : Recherche en mode Backtracking (Backtracking Search)



- Idée de base de Backtracking Search
- Représentation standard → état initial, fonction successeur
- `Select-Unassigned-Variable`, `Order-Domain-value` peuvent être utilisées pour implémenter des heuristiques générales
- Quand l'espace de recherche est très large, l'algorithme Backtracking Search n'est pas efficace
- Certaines améliorations sont possibles si on considère les points suivantes :

Heuristiques pour la résolution des CSPs

- Ordre des variables et des valeurs
- `var ← Select-Unassigned-Variable(variable[csp], assignment, csp);`
- Cette instruction permet de sélectionner une variable non traitée selon l'ordre établi dans `variable[csp]`
- Ceci conduit rarement à une recherche efficace
- Solution
  - Choisir la variable avec le nombre minimum de valeurs restantes
  - Minimum Remaining Value (MRV heuristic)
  - Connue aussi sous le nom de Most Constrained Variable

Heuristiques pour la résolution des CSPs

- Ordre des variables et des valeurs
- `var ← Select-Unassigned-Variable(variable[csp], assignment, csp);`
  - Dans quel ordre ces valeurs doivent être testées ?
- Remarque
  - Si une variable X à 0 valeurs restantes, l'heuristique MRV va sélectionner X
  - Un échec est immédiatement détecté
  - Gaspillage d'efforts pour tester une autre variable